FINAL REPORT

NASA RESEARCH GRANT NAG4-157

Period: August 27, 1998 — September 30, 2000

DEVELOPMENT OF AN UNSTRUCTURED MESH CODE FOR FLOWS ABOUT COMPLETE VEHICLES

submitted to:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
DRYDEN FLIGHT RESEARCH CENTER

Technical Officer: **Dr. K.K. Gupta**
NASA DFRC, PO Box 273, M.S. D-1013
Edwards, CA 93523-0273

Grant Negotiator: **S. Yee**
NASA DFRC, PO Box 273, M.S. D-1422
Edwards, CA 93523-0273

Principal Investigator: **Professor J. Peraire**
Massachusetts Institute of Technology, 37-451
Department of Aeronautics
Cambridge, MA 02139

1 Introduction

This report describes the research work undertaken at the Massachusetts Institute of Technology, under NASA Research Grant NAG4-157. The aim of this research is to identify effective algorithms and methodologies for the efficient and routine solution of flow simulations about complete vehicle configurations.

For over ten years we have received support from NASA to develop unstructured mesh methods for Computational Fluid Dynamics. As a result of this effort a methodology based on the use of unstructured adapted meshes of tetrahedra and finite volume flow solvers has been developed. A number of gridding algorithms, flow solvers, and adaptive strategies have been proposed. The most successful algorithms developed from the basis of the unstructured mesh system FELISA. The FELISA system has been extensively for the analysis of transonic and hypersonic flows about complete vehicle configurations. The system is highly automatic and allows for the rotine aerodynamic analysis of complex configurations starting from CAD data. The code has been parallelized and utilizes efficient solugion algorithms. For hypersonic flows, a version of the code which incorporates real gas effects, has been produced. The FELISA system is also a component of the STARS aeroservoelastic system developed at NASA Dryden. One of the latest developments before the start of this grant was to extend the system to include viscous effects. This required the development of viscous generators, capable of generating the anisotropic grids required to represent boundary layers, and viscous flow solvers. In figures 1 and 2, we show some sample hypersonic viscous computations using the developed viscous generators and solvers.

Although this initial results were encouraging it became apparent that in order to develop a fully functional capability for viscous flows, several advances in solution accuracy, robustness and efficiency were required. In this grant we set out to investigate some novel methodologies that could lead to the required improvements. In particular we focused on two fronts: finite element methods [1], [26] and iterative algebraic multigrid solution techiques [30].

Finite element algorithms have been enormously successful in the field of structural mechanics and in that field, they are the method of choice. They have a solid mathematical foundation and offer complete geometrical flexibility. Finite element methods utilize compact supports, which substantially aid the implict solution procedure, and, when properly formulated, can be extended to arbitrary orders of accuracy.

Multigrid techniques are well known for their efficiency in solving symmetric positive definite

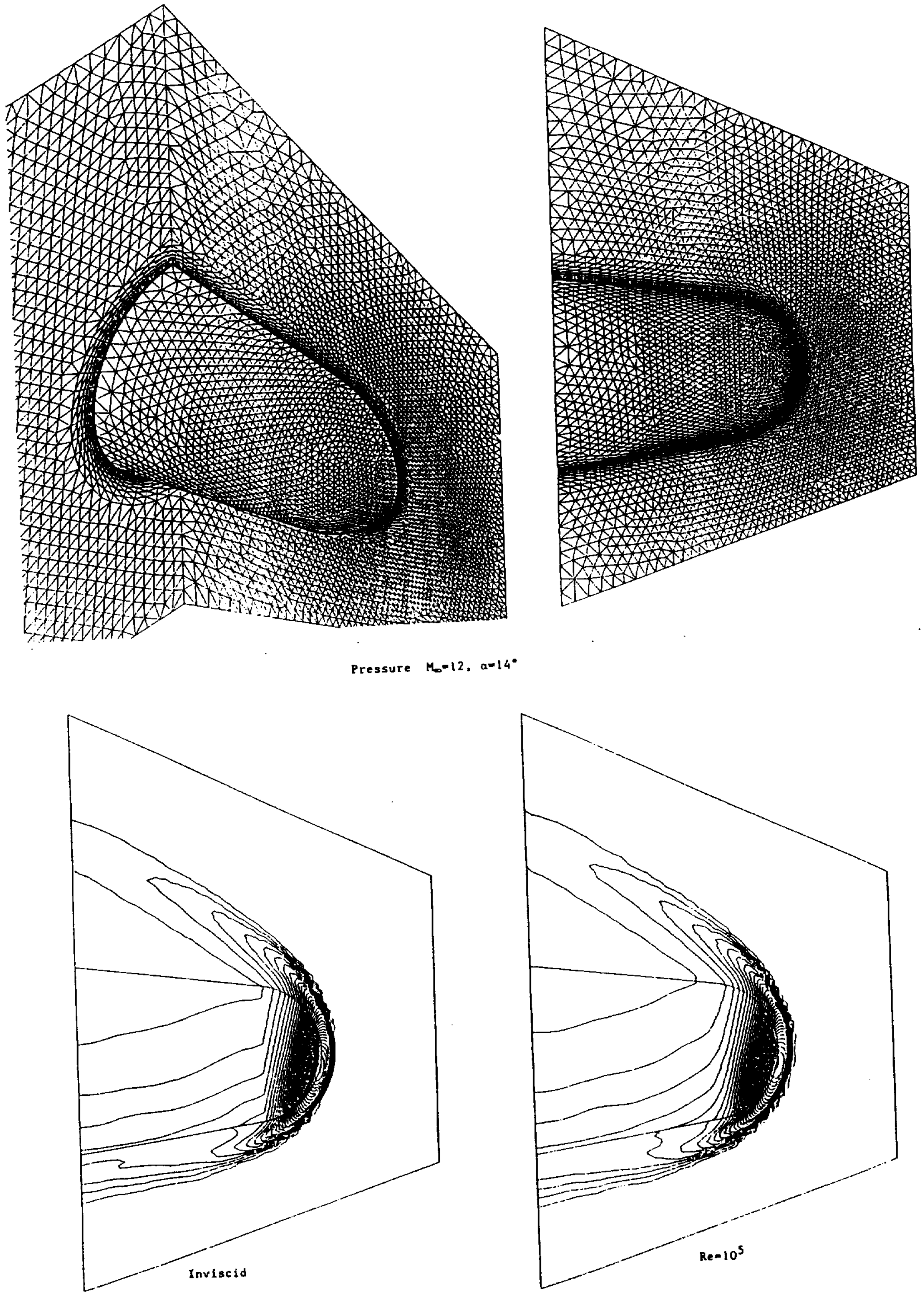


Figure 1: Mach 12 flow about a cone-sphere geometry. Meshes and computed inviscid and viscous pressure contour solutions on the same mesh

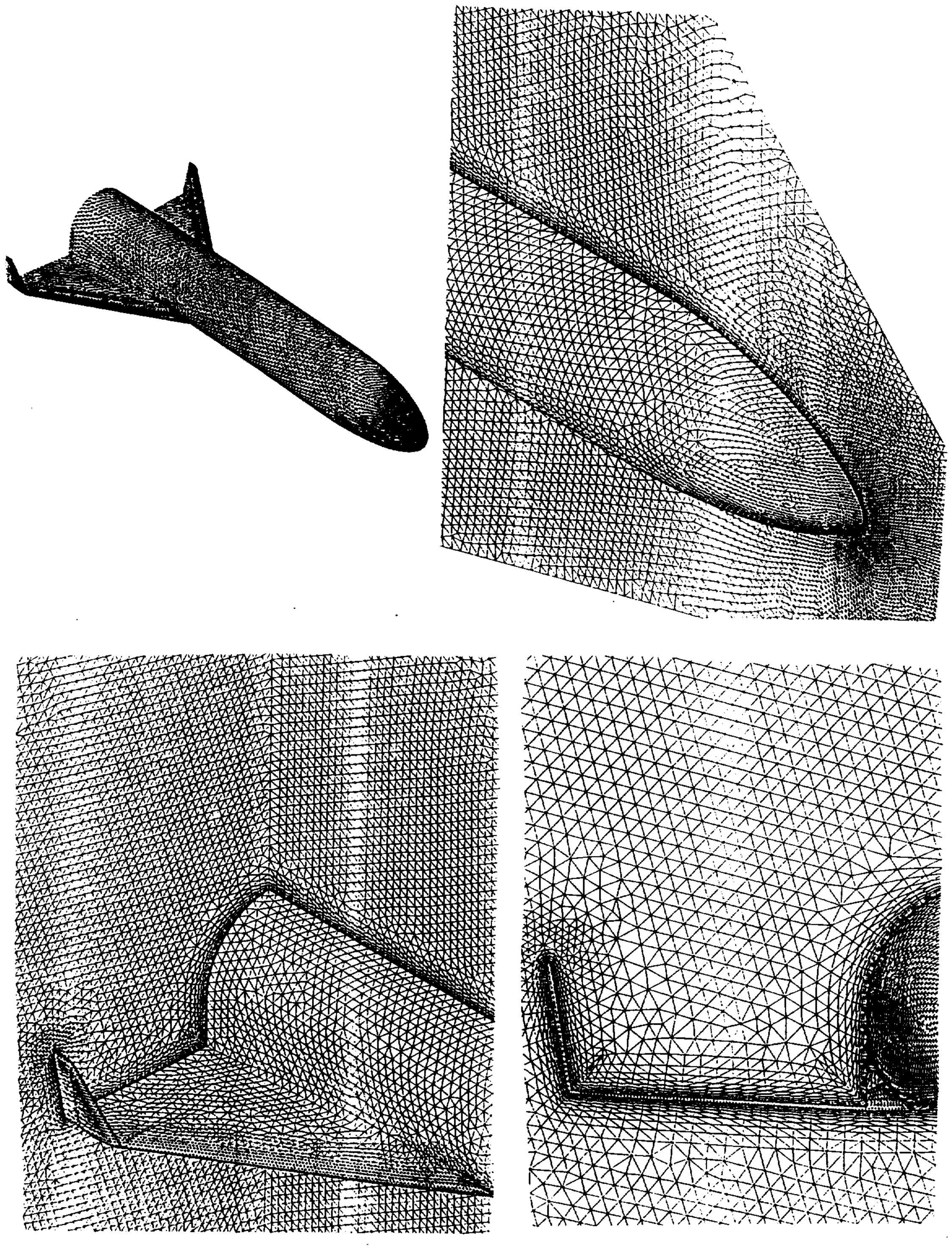
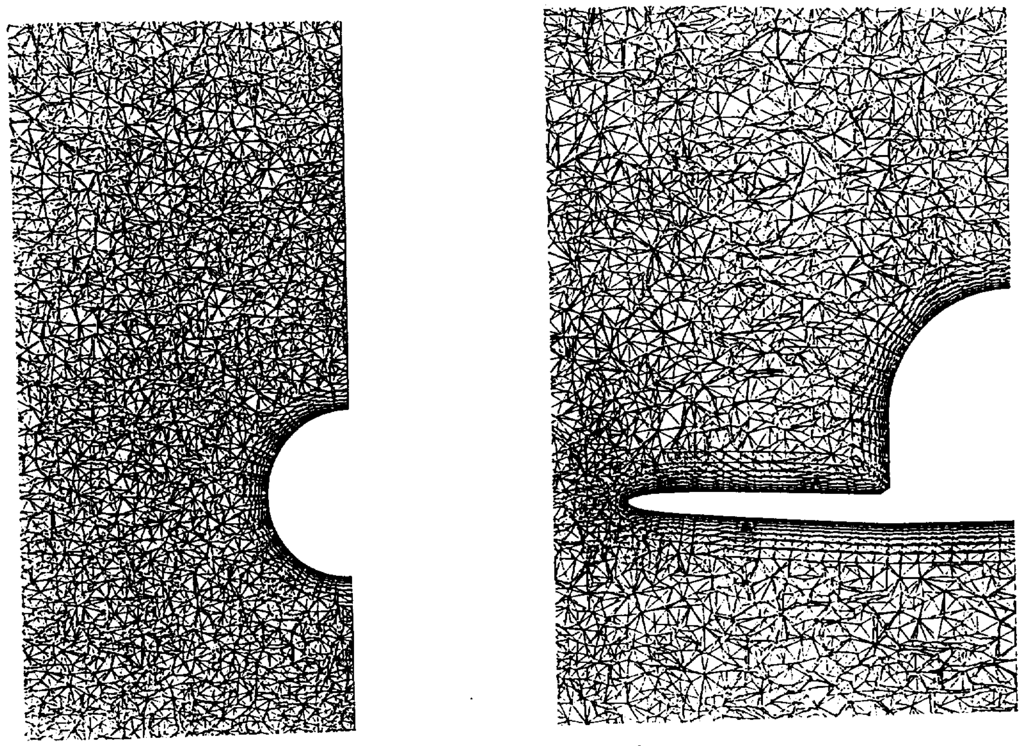
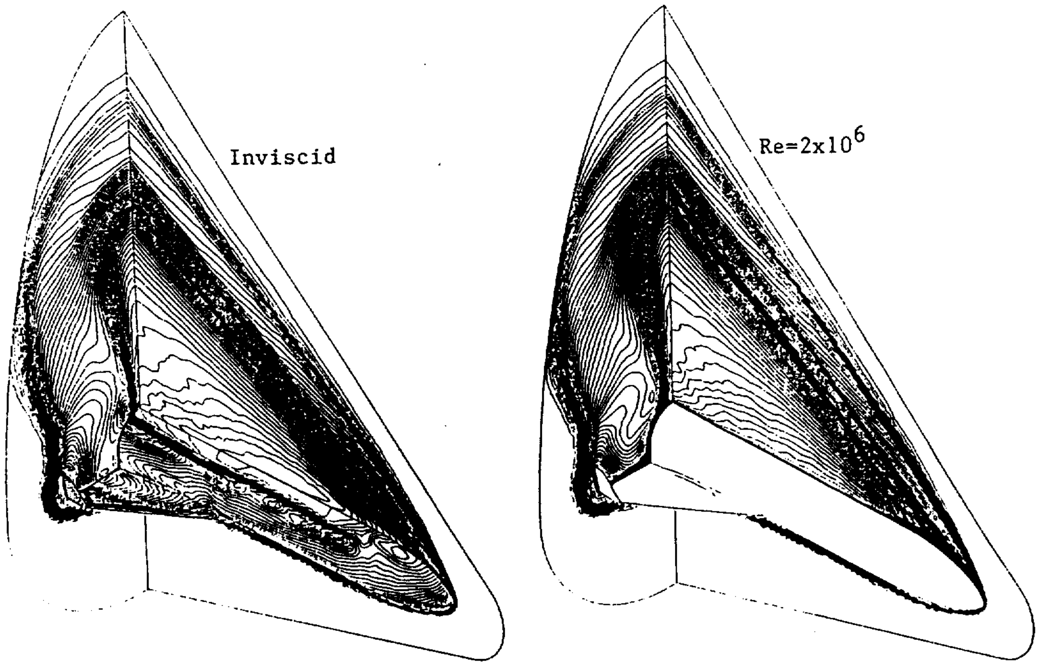


Figure 2: Mach 10.6 flow about a re-entry configuration. Meshes and Mach number contours for inviscid and viscous solutions on the same mesh



Mach $M_\infty=10.6$, $\alpha=14°$



problems. Unfortunately, the discretization of the Navier-Stokes equations generates systems of equations which are unsymmetric and much harder to solve.

The work reported below has been partially supported by this grant.

2 Finite Element Solution of the Compressible Flow equations

Our aim is to develop a robust and accurate finite element algorithm which is capable of solving flows ranging from low subsonic to transonic and hypersonic regimes. We regard the possibility of using higher order approximations as being highly attractive, specially for complex three dimensional flows which requiring high levels of resolution.

Here we present some of the accomplishments carried out during this grant period and which build upon existing finite element methods.

2.1 The solution of the Compressible Euler Equations at Low Mach Numbers

For low Mach numbers, the compressible Euler and Navier Stokes equations describe almost incompressible flow. This singular limit of the compressible flow equations is reasonably well understood. Indeed, under the assumption of isentropic flow, and some regularity conditions on the initial and boundary data, the solution of the compressible equations, in the zero Mach number limit, can be shown to satisfy the incompressible flow equations [16, 11]. From the computational point of view, accurate solutions of nearly incompressible flows are difficult to obtain. This is due to the very different magnitude of the wave speeds which are present in the system. Our interest is in the development of a compressible flow algorithm which is capable of solving flows ranging from almost incompressible to supersonic regimes. There are several reasons that justify the development of such algorithm. The first and most important is that in many situations, such as high angle of attack aerodynamics, large regions of very low Mach number coexist in the flow domain with regions where the flow is supersonic. Another more practical motivation is that an algorithm that can successfully handle free stream Mach numbers as low as 10^{-3} is well suited for a broad range of applications typically handled with incompressible formulations.

Over the last few years, stabilized finite element algorithms for the solution of the Euler and Navier-Stokes equations have gained increased popularity. Streamline-Upwind/Petrov-Galerkin algorithms (SUPG), and some of its relatives, have been presented and analyzed in numerous papers [14, 15, 20]. These algorithms, although not so widely used as their finite volume counterparts, possess many attractive features. In particular, they have a compact support and can be used with elements of arbitrary order, yielding solutions of increased accuracy whenever the solution is sufficiently smooth. In addition, there exists a rather well developed theory for linear problems which can be used to provide design criteria for the development of successful algorithms for non-linear equations.

One of the critical ingredients of SUPG algorithms is the construction of the stabilization matrix $\boldsymbol{\tau}$. Whilst the convergence analysis for the linear problem only dictates that this matrix has to be symmetric positive definite, scale appropriately with the local grid size, and have dimensions of time, much freedom is still available to fully determine it. Only under some very simplified cases (e.g. one dimensional flow) is the optimal choice of $\boldsymbol{\tau}$ unambiguous.

Classical compressible flow formulations, including the existing SUPG algorithms, fail to give adequate numerical solutions when the flow approaches incompressibility. Whenever the Mach number is progressively reduced, keeping the grid size fixed, a degradation in the solution accuracy is observed. This phenomenon has been studied extensively in the context of finite volume algorithms [23, 18, 6], and several explanations have been proposed. The most common argument is based on the mismatch which occurs, for low Mach numbers, between magnitude of the fluxes in the original equations and the corresponding terms in the numerically added artificial viscosity. Some local preconditioning strategies, aimed at modifying the numerical artificial viscosity to avoid this mismatch, have proven to be extremely successful [3, 24, 4, 25]. Accurate solutions for Mach numbers as low as 10^{-3} have been reported using such preconditioned algorithms. One of the main drawbacks of these preconditioners is their lack of robustness near stagnation points. The reason for this can be traced [5] to a lack of stabilization caused by the eigenvectors of the artificial dissipation matrix becoming nearly parallel.

The formulation of numerical algorithms for the compressible flow equations employing symmetrizing, or entropy, variables has been advocated by several authors e.g. [13, 2]. One of the claims often made is that discrete schemes formulated in entropy variables inherit global entropy stability properties of the original equations. Gustafsson [10] points out that, for a hyperbolic system of equations, the higher the degree of unsymmetry, as measured by the condition number of the transformation required to symmetrize the system, the lesser the well-posedness of the problem, in the sense that perturbations of the initial data influence the solution more. As it turns out, the compressible Euler equations formulated in terms of either primitive or conservative variables become increasingly unsymmetric when the reference Mach number goes to zero. From our perspective, one of the main attractive features derived from the use of entropy variables is that the dissipation operator remains symmetric. This means that a full set of orthogonal eigenvectors can always be found and therefore the stabilization terms remain effective throughout the computational doamin. In this respect, the combination of the preconditioning ideas presented in the finite volume context to deal with low Mach number flows combined with a formulation in entropy variables seems very appealing.

Here, we propose a specific construction of the stabilization matrix $\boldsymbol{\tau}$ for a finite element SUPG formulation of the steady state Euler equations. The development of such stabilization matrix incorporates low Mach number preconditioning concepts, previously developed in the finite volume context. The ideas presented here extend to the time dependent and the Navier-Stokes equations in a straightforward manner. The equations are first transformed into a new set of variables which, in the low Mach number limit, can be easily related to the incompressible velocity and pressure. A necessary condition for stability, when the Mach number tends to zero, is obtained by requiring that the asymptotic behavior of the stabilization terms matches that of the terms in the Euler equations. This requirement results in some additional constraints on $\boldsymbol{\tau}$ which are not typically satisfied by the classical choices of $\boldsymbol{\tau}$. The proposed algorithm combines the very attractive features of the stabilized finite element formulations with the ability to produce accurate answers over a very large range of Mach numbers.

We start with a brief description of the SUPG formulation for the Euler equations using entropy variables. For simplicity of presentation we will consider the two dimensional case, but results herein presented extend directly to three dimensions and are also applicable to the Navier-Stokes equations. Next, we discuss the low Mach number scaling arguments which lead to the proposed form of stabilization matrix $\boldsymbol{\tau}$ and outline our numerical implementation. Finally, numerical results using linear elements are presented for the flow over isolated cylinders and airfoils with free stream Mach numbers ranging from 10^{-3} to moderate subsonic values. In the next section, we present results for transonic flows using higher order elements.

2.2 Compressible flow governing equations

2.2.1 Conservation variables

We start from the time dependent two dimensional compressible Euler equations in conservation form

$$\mathbf{U}_{,t} + \mathbf{F}_{1,1} + \mathbf{F}_{2,2} = 0, \tag{1}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix}, \quad \mathbf{F}_1 = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ u_1(\rho E + p) \end{bmatrix}, \quad \mathbf{F}_2 = \begin{bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ u_2(\rho E + p) \end{bmatrix}.$$

In the above expressions, ρ is the density; $\mathbf{u} = [u_1, u_2]^T$ is the velocity vector; E is the specific total energy; p is the pressure; and the comma denotes partial differentiation (e.g. $\mathbf{U}_{,t} = \partial \mathbf{U}/\partial t$, the partial derivative with respect to time, $\mathbf{F}_{i,j} = \partial \mathbf{F}_i/\partial x_j$, the partial derivative with respect to the j-th spatial coordinate). The system of equations is closed once the pressure is related to the problem variables through the equation of state, $p = (\gamma - 1)\rho e$, where $e = E - |\mathbf{u}|^2/2$, is the internal energy. Here, γ is the ratio of specific heats which is assumed to be constant.

We will assume that all the above quantities have been non-dimensionalized using reference, or free stream, values for density ρ_*, velocity u_*, and length L. Thus, the dimensional variables, denoted with an overbar, are related to the non-dimensional variables introduced above as

$$\rho = \frac{\bar{\rho}}{\rho_*}, \quad u_i = \frac{\bar{u}_i}{u_*}, \; i = 1, 2, \quad p = \frac{\bar{p}}{\rho_* u_*^2}, \quad E = \frac{\bar{E}}{u_*^2}, \quad x_i = \frac{\bar{x}_i}{L}, \; i = 1, 2, \quad \text{and} \quad t = \frac{u_* \bar{t}}{L}.$$

Finally, we introduce the reference speed of sound c_*, and define, for later use, a reference Mach number as $\epsilon = u_*/c_*$.

We note that the equation system (1) can be written as

$$\mathbf{U}_{,t} + \mathbf{A}_1 \mathbf{U}_{,1} + \mathbf{A}_2 \mathbf{U}_{,2} = 0, \tag{2}$$

where the Jacobian matrices $\mathbf{A}_i = \mathbf{F}_{i,\mathbf{U}}$, $i = 1, 2$, are unsymmetric but have real eigenvalues and a complete set of eigenvectors.

2.2.2 Entropy variables

We seek a new set of variables $\mathbf{V}$, called entropy variables, such that the change $\mathbf{U} = \mathbf{U}(\mathbf{V})$ applied to (1) give the transformed system

$$\mathbf{U}(\mathbf{V})_{,t} + \mathbf{F}_1(\mathbf{V})_{,1} + \mathbf{F}_2(\mathbf{V})_{,2} = 0, \tag{3}$$

where $\mathbf{A}_0 = \mathbf{U}_{,\mathbf{V}}$ is symmetric positive definite, and $\tilde{\mathbf{A}}_i = \mathbf{A}_i\mathbf{A}_0 = \mathbf{F}_{i,\mathbf{V}}$, $i = 1, 2$, are symmetric.

Following [12], we introduce a scalar entropy function $H(\mathbf{U}) = -\rho g(s)$, where s is the non-dimensional entropy $s = \ln(p/\rho^\gamma)$. The required change of variables is obtained by taking

$$\mathbf{V} = H_{,\mathbf{U}}^T = \frac{g'}{e}\begin{bmatrix} e(\gamma - g/g') - |\mathbf{u}|^2/2 \\ u_1 \\ u_2 \\ -1 \end{bmatrix}. \tag{4}$$

The conditions $g' > 0$ and $g''/g' < \gamma^{-1}$, ensure that $H(\mathbf{U})$ is a convex function and therefore $\mathbf{A}_0^{-1} = \mathbf{V}_{,\mathbf{U}} = H_{,\mathbf{UU}}$, and $\mathbf{A}_0$, are symmetric positive definite. We consider in [1] the variables resulting from two particular choices of the function $g(s)$. The system (3), can thus be written in symmetric quasi-linear form as

$$\mathbf{A}_0\mathbf{V}_{,t} + \tilde{\mathbf{A}}_1\mathbf{V}_{,1} + \tilde{\mathbf{A}}_2\mathbf{V}_{,2} = 0. \tag{5}$$

Barth [2] noted that it is always possible to construct matrices $\tilde{\mathbf{R}}_i$, $i = 1, 2$ whose columns are the scaled right eigenvectors of $\mathbf{A}_i$, $i = 1, 2$ respectively, such that

$$\mathbf{A}_0 = \tilde{\mathbf{R}}_1\tilde{\mathbf{R}}_1^T = \tilde{\mathbf{R}}_2\tilde{\mathbf{R}}_2^T, \tag{6}$$

and

$$\mathbf{A}_i = \tilde{\mathbf{R}}_i\Lambda_i\tilde{\mathbf{R}}_i^{-1}, \quad \tilde{\mathbf{A}}_i = \tilde{\mathbf{R}}_i\Lambda_i\tilde{\mathbf{R}}_i^T, \quad \text{for} \quad i = 1, 2. \tag{7}$$

An explicit expression for $\tilde{\mathbf{R}}_i$, $i = 1, 2$, is given in [1].

2.3 Low Mach number limit

The low Mach number limit of the Euler equations is best studied by rewriting the equations in terms of the variables (p, u_1, u_2, s). This yields the following system

$$\begin{bmatrix} \frac{1}{\rho c}\frac{\partial p}{\partial t} \\ \frac{\partial u_1}{\partial t} \\ \frac{\partial u_2}{\partial t} \\ \frac{\partial s}{\partial t} \end{bmatrix} + \begin{bmatrix} u_1 & c & 0 & 0 \\ c & u_1 & 0 & 0 \\ 0 & 0 & u_1 & 0 \\ 0 & 0 & 0 & u_1 \end{bmatrix} \begin{bmatrix} \frac{1}{\rho c}\frac{\partial p}{\partial x_1} \\ \frac{\partial u_1}{\partial x_1} \\ \frac{\partial u_2}{\partial x_1} \\ \frac{\partial s}{\partial x_1} \end{bmatrix} + \begin{bmatrix} u_2 & 0 & c & 0 \\ 0 & u_2 & 0 & 0 \\ c & 0 & u_2 & 0 \\ 0 & 0 & 0 & u_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\rho c}\frac{\partial p}{\partial x_2} \\ \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_2} \\ \frac{\partial s}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{8}$$

In matrix form, this system can be written as

$$\mathbf{Z}_{,t} + \mathbf{C}_1 \mathbf{Z}_{,1} + \mathbf{C}_2 \mathbf{Z}_{,2} = 0, \tag{9}$$

where the matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ are symmetric and $d\mathbf{Z} = [d\phi, du_1, du_2, ds]^T$ with $d\phi = \frac{dp}{\rho c}$. If we assume that the flow is isentropic, which is certainly satisfied for low Mach numbers when the free stream is isentropic, we can express c and p as a function of ρ, i.e,

$$p = \frac{\rho^\gamma}{\gamma \epsilon^2} \qquad \text{and} \qquad c = \frac{\rho^{(\gamma-1)/2}}{\epsilon},$$

to obtain [9],

$$\phi = \frac{2}{(\gamma-1)\epsilon}(\rho^{(\gamma-1)/2} - 1) = \frac{2}{\gamma-1}(c - \frac{1}{\epsilon}).$$

In terms of ϕ, u_1 and u_2, the continuity and momentum isentropic compressible flow equations become

$$\begin{aligned} \phi_{,t} + u_1\phi_{,1} + \left(\frac{\gamma-1}{2}\phi + \frac{1}{\epsilon}\right) u_{1,1} + u_2\phi_{,2} + \left(\frac{\gamma-1}{2}\phi + \frac{1}{\epsilon}\right) u_{2,2} &= 0, \\ u_{1,t} + \left(\frac{\gamma-1}{2}\phi + \frac{1}{\epsilon}\right)\phi_{,1} + u_1 u_{1,1} + u_2 u_{1,2} &= 0, \\ u_{2,t} + u_1 u_{2,1} + \left(\frac{\gamma-1}{2}\phi + \frac{1}{\epsilon}\right)\phi_{,2} + u_2 u_{2,2} &= 0. \end{aligned} \tag{10}$$

Note that, in assuming isentropic flow, we have reduced the number of dependent variables by eliminating entropy from the system of governing equations. Equations (10), look now similar to the incompressible flow equations

$$\begin{aligned} \hat{u}_{1,1} + \hat{u}_{2,2} &= 0 \\ \hat{u}_{1,t} + \hat{p}_{,1} + \hat{u}_1\hat{u}_{1,1} + \hat{u}_2\hat{u}_{1,2} &= 0 \\ \hat{u}_{2,t} + \hat{u}_1\hat{u}_{2,1} + \hat{p}_{,2} + \hat{u}_2\hat{u}_{2,2} &= 0, \end{aligned} \tag{11}$$

where $\hat{\mathbf{u}} = [\hat{u}_1, \hat{u}_2]^T$, denotes the velocity and $\hat{p}$ the pressure. When $\epsilon \to 0$, the equivalence between (10) and (11), under some regularity assumptions on the initial and boundary data, can be rigorously shown [11]. In particular, we have that for $\epsilon \to 0$, $u_i \to \hat{u}_i$ and $((\gamma - 1)\phi/2 + 1/\epsilon)\, \phi_{,i} \to \hat{p}_{,i}$ for $i = 1, 2$. To obtain bounded derivatives when $\epsilon \to 0$, it follows that $\phi_{,i}$ and $u_{1,1} + u_{2,2}$ must be $O(\epsilon)$. We also note that the velocity components u_i and its spatial derivatives $u_{i,j}, i, j = 1, 2$ are $O(1)$. Finally, it can be shown that ϕ is $O(\epsilon)$ and that ϕ/ϵ is well defined in the limit $\epsilon \to 0$.

We now turn our attention to the steady state Euler equations for isentropic flow and derive some asymptotic estimates which are valid in the low Mach number limit. The equations for this reduced problem are,

$$\mathbf{C}_1^I \mathbf{Z}_{,1}^I + \mathbf{C}_2^I \mathbf{Z}_{,2}^I = 0 \tag{12}$$

where

$$\mathbf{Z}^I = \begin{bmatrix} \phi \\ u_1 \\ u_2 \end{bmatrix}, \qquad \mathbf{C}_1^I = \begin{bmatrix} u_1 & c & 0 \\ c & u_1 & 0 \\ 0 & 0 & u_1 \end{bmatrix}, \qquad \mathbf{C}_2^I = \begin{bmatrix} u_2 & 0 & c \\ 0 & u_2 & 0 \\ c & 0 & u_2 \end{bmatrix}.$$

Using the above estimates we have that, for $\epsilon \to 0$,

$$\mathbf{Z}_{,1}^I \sim \mathbf{Z}_{,2}^I \sim \begin{bmatrix} O(\epsilon) \\ O(1) \\ O(1) \end{bmatrix}, \ \mathbf{C}_1^I \sim \begin{bmatrix} O(1) & O(\epsilon^{-1}) & 0 \\ O(\epsilon^{-1}) & O(1) & 0 \\ 0 & 0 & O(1) \end{bmatrix}, \ \mathbf{C}_2^I \sim \begin{bmatrix} O(1) & 0 & O(\epsilon^{-1}) \\ 0 & O(1) & 0 \\ O(\epsilon^{-1}) & 0 & O(1) \end{bmatrix}. \tag{13}$$

Thus, for the individual terms of the isentropic Euler equations,

$$\mathbf{C}_1^I \mathbf{Z}_{,1}^I \sim \mathbf{C}_2^I \mathbf{Z}_{,2}^I \sim \begin{bmatrix} O(\epsilon^{-1}) \\ O(1) \\ O(1) \end{bmatrix} \quad \text{as } \epsilon \to 0. \tag{14}$$

2.4 Variational formulation for the steady state problem

We now consider the compressible steady problem in conservation form expressed in terms of symmetrizing variables. The conservative form of the equations is taken to be the starting point because we are ultimately interested in an algorithm that can be used over the whole range of speed regimes, including situations were the solution may contain discontinuitites. The problem is defined in a domain Ω with boundary Γ by

$$\mathbf{F}_1(\mathbf{V})_{,1} + \mathbf{F}_2(\mathbf{V})_{,2} = 0 \quad \text{in} \quad \Omega, \tag{15}$$

$$\tilde{\mathbf{A}}_n^- \mathbf{V} = \tilde{\mathbf{A}}_n^- \mathbf{g} \quad \text{on} \quad \Gamma \backslash \Gamma_a, \tag{16}$$

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on} \quad \Gamma_a. \tag{17}$$

For simplicity, the domain boundary is assumed to be made up of an impermeable solid wall Γ_a, and a computational far field boundary $\Gamma\backslash\Gamma_a$. In (16, 17), $\mathbf{n} = [n_1, n_2]^T$ is the outward unit normal vector to Γ, and $\tilde{\mathbf{A}}_n = \mathbf{A}_n \mathbf{A}_0$, $\mathbf{A}_n = \mathbf{A}_1 n_1 + \mathbf{A}_2 n_2$. Finally, $\tilde{\mathbf{A}}_n^- = \mathbf{A}_n^- \mathbf{A}_0$, and $\mathbf{A}_n^-$ denotes the negative definite part of $\mathbf{A}_n$.

Let the spatial domain Ω, be discretized into non-overlapping elements T_e, such that $\Omega = \bigcup T_e$, and $T_e \bigcap T_{e'} = \emptyset$, $e \neq e'$. We consider the space of functions $\mathcal{V}_h$, defined over the discretization and consisting of the continuous functions which are piecewise linear over each element

$$\mathcal{V}_h = \{\mathbf{W} \mid \mathbf{W} \in (C^0(\Omega))^4, \ \mathbf{W}|_{T_e} \in (\mathcal{P}_1(T_e))^4, \ \forall T_e \in \Omega\}.$$

The SUPG algorithm can then be written as: Find $\mathbf{V}_h \in \mathcal{V}^h$ such that for all $\mathbf{W} \in \mathcal{V}^h$,

$$B(\mathbf{V}_h, \mathbf{W})_{gal} + B(\mathbf{V}_h, \mathbf{W})_{supg} + B(\mathbf{V}_h, \mathbf{W})_{bc} = 0, \tag{18}$$

where the forms $B(\cdot,\cdot)_{gal}$, $B(\cdot,\cdot)_{supg}$ and $B(\cdot,\cdot)_{bc}$ account for the Galerkin, SUPG stabilization, and boundary condition terms respectively, and are defined as

$$B(\mathbf{V}, \mathbf{W})_{gal} = \int_\Omega (-\mathbf{W}_{,1} \cdot \mathbf{F}_1(\mathbf{V}) - \mathbf{W}_{,2} \cdot \mathbf{F}_2(\mathbf{V}))\, d\Omega, \tag{19}$$

$$B(\mathbf{V}, \mathbf{W})_{supg} = \int_\Omega (\tilde{\mathbf{A}}_1 \mathbf{W}_{,1} + \tilde{\mathbf{A}}_2 \mathbf{W}_{,2}) \cdot \boldsymbol{\tau}\, (\tilde{\mathbf{A}}_1 \mathbf{V}_{,1} + \tilde{\mathbf{A}}_2 \mathbf{V}_{,2})\, d\Omega, \tag{20}$$

and

$$B(\mathbf{V}, \mathbf{W})_{bc} = \int_{\Gamma_a} \mathbf{W} \cdot \mathbf{F}_w(\mathbf{V}; \mathbf{n})\, ds + \int_{\Gamma\backslash\Gamma_a} \mathbf{W} \cdot \mathbf{F}_{ff}(\mathbf{V}, \mathbf{g}; \mathbf{n})\, ds. \tag{21}$$

where $\boldsymbol{\tau}$ is the stabilization matrix. The numerical flux function on the impermeable wall boundary $\mathbf{F}_w$, is simply $[0, pn_1, pn_2, 0]^T$ while the numerical flux function on the far field boundary $\mathbf{F}_{ff}$, is defined by

$$\mathbf{F}_{ff}(\mathbf{V}_-, \mathbf{V}_+; \mathbf{n}) = \frac{1}{2}(\mathbf{F}_n(\mathbf{V}_-) + \mathbf{F}_n(\mathbf{V}_+)) - \frac{1}{2}|\mathbf{A}_n(\mathbf{V}_{Roe}(\mathbf{V}_-, \mathbf{V}_+))|(\mathbf{U}(\mathbf{V}_+) - \mathbf{U}(\mathbf{V}_-)).$$

Here, $|\mathbf{A}_n(\mathbf{V})| = \mathbf{A}_n^+(\mathbf{V}) - \mathbf{A}^-(\mathbf{V})$ is the absolute value of $\mathbf{A}_n$ evaluated at $\mathbf{V}$, and $\mathbf{V}_{Roe}(\mathbf{V}_+, \mathbf{V}_-)$, is the Roe average [19], between the states $\mathbf{V}^+$ and $\mathbf{V}^-$.

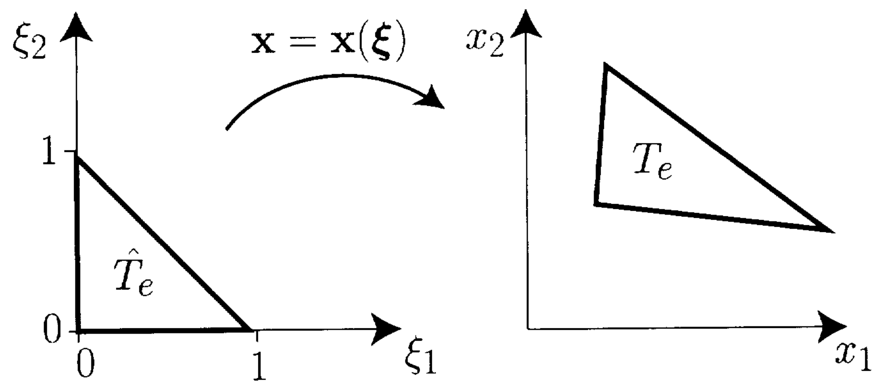


Figure 3: Mapping between the master triangle $\hat{T}_e$ and a typical element T_e.

2.4.1 Standard definitions for $\boldsymbol{\tau}$

In order to uniquely specify the SUPG algorithm (18), an appropriate expression for the stabilization matrix $\boldsymbol{\tau}$ needs to be given. We recall that an appropriate $\boldsymbol{\tau}$ should be symmetric, positive definite, have dimensions of time, and scale linearly with the element size [14].

For a triangle T_e, we introduce the mapping $\mathbf{x} = \mathbf{x}(\xi)$, between the master triangle $\hat{T}_e$, in the parametric space $\boldsymbol{\xi} = [\xi_1, \xi_2]^T$, and $T_e \in \Omega$ in the mapped space $\mathbf{x} = [x_1, x_2]^T$, as illustrated in figure 3. We define

$$\begin{aligned}
\mathbf{B}_1 &= (x_{2,\xi_2}\tilde{\mathbf{A}}_1 - x_{1,\xi_2}\tilde{\mathbf{A}}_2)/\mathbf{J},\\
\mathbf{B}_2 &= (x_{1,\xi_1}\tilde{\mathbf{A}}_2 - x_{2,\xi_1}\tilde{\mathbf{A}}_1)/\mathbf{J},\\
\mathbf{B}_3 &= ((x_{2,\xi_1} - x_{2,\xi_2})\tilde{\mathbf{A}}_1 + (x_{1,\xi_2} - x_{1,\xi_1})\tilde{\mathbf{A}}_2)/\mathbf{J},
\end{aligned}$$

and the Jacobian of the mapping, $\mathbf{J} = x_{1,\xi_1}x_{2,\xi_2} - x_{2,\xi_1}x_{1,\xi_2}$. The following definition for $\boldsymbol{\tau}$ can be shown to satisfy all of the above requirements

$$\boldsymbol{\tau} = \mathbf{A}_0{}^{-1}\left(|\mathbf{B}_1|^p + |\mathbf{B}_2|^p + |\mathbf{B}_3|^p\right)^{-1/p}. \tag{22}$$

The most common choice for p has been 2, which necessitates the evaluation of a matrix square root. This choice has been advocated by Shakib et al. [20], whereas Barth [2], has shown that the choice $p = 1$ is computationally advantageous and, in practice, gives very similar results.

These choices of stabilization matrix, work well provided that the flow Mach numberis not too low. For problems involving significant regions of low Mach number flow the solution degrades and becomes less and less accurate as the Mach number is decreased. In the following sections we will address this issue by first, identifying the source of the problem, and second, devising a new formulation for $\boldsymbol{\tau}$ which does not suffer from this drawback and which leads to a formulation which can handle very low Mach numbers accurately.

2.5 A stabilized formulation for isentropic flows

Turkel et al. [23] have employed a scaling analysis to determine the appropriate low Mach number behavior of local preconditioners. In their application, the local preconditioner modifies the upwind dissipation of the underlying finite volume or finite difference discretization. In this section, we will consider a stabilized SUPG formulation applied direclty to the isentropic equations (12). We will extend the analysis presented in [23] to our finite element formulation and, in particular, derive the appropriate scaling for the stabilization matrix in the limit of vanishing Mach number.

Introducing the finite element space,

$$\mathcal{V}_h^I = \{\mathbf{W}^I \mid \mathbf{W}^I \in (C^0(\Omega))^3,\ \mathbf{W}^I|_{T_e} \in (\mathcal{P}_1(T_e))^3,\ \forall T_e \in \Omega\}.$$

we consider the following SUPG formulation : Find $\mathbf{Z}_h^I \in \mathcal{V}_h^I$ such that for all $\mathbf{W}^I \in \mathcal{V}_h^I$,

$$B^I(\mathbf{Z}_h^I, \mathbf{W}^I)_{gal} + B^I(\mathbf{Z}_h^I, \mathbf{W}^I)_{supg} + B^I(\mathbf{Z}_h^I, \mathbf{W}^I)_{bc} = 0, \tag{23}$$

$$B^I(\mathbf{Z}^I, \mathbf{W}^I)_{gal} = \int_\Omega \left(\mathbf{W}^I \cdot (\mathbf{C}_1^I \mathbf{Z}_{,1}^I + \mathbf{C}_2^I \mathbf{Z}_{,2}^I)\right)\, d\Omega, \tag{24}$$

where,

$$B^I(\mathbf{Z}^I, \mathbf{W}^I)_{supg} = \int_\Omega (\mathbf{C}_1^I \mathbf{W}_{,1}^I + \mathbf{C}_2^I \mathbf{W}_{,2}^I) \cdot \boldsymbol{\tau}^I\ (\mathbf{C}_1^I \mathbf{Z}_{,1}^I + \mathbf{C}_2^I \mathbf{Z}_{,2}^I)\, d\Omega, \tag{25}$$

and $B^I(\cdot,\cdot)_{bc}$ is a term incorporating the desired boundary conditions. Note that here, the Galerkin term (24) has been integrated by parts, thus, $B^I(\cdot,\cdot)_{bc}$ also includes the additional boundary terms.

Our desire is to construct a stabilized finite element method which admits solutions, $\mathbf{Z}_h^I$, with the same low Mach number asymptotic behavior as the solutions of the isentropic Euler equations, $\mathbf{Z}^I$. In addition, we require that the stabilization operator (25) must scale like the Galerkin and boundary operators (24) as the Mach number is reduced. Specifically, these requirements imply that

$$\mathbf{C}_1^I \mathbf{Z}_{h,1}^I + \mathbf{C}_2^I \mathbf{Z}_{h,2}^I \sim \begin{bmatrix} O(\epsilon^{-1}) \\ O(1) \\ O(1) \end{bmatrix} \quad \text{for} \quad \epsilon \to 0, \tag{26}$$

and that,

$$\mathbf{C}_i^I \cdot \bar{\tau}^I \left(\mathbf{C}_1^I \mathbf{Z}_{h,1} + \mathbf{C}_2^I \mathbf{Z}_{h,2}\right) \sim \mathbf{C}_1^I \mathbf{Z}_{h,1}^I + \mathbf{C}_1^I \mathbf{Z}_{h,1}^I, \quad \text{for} \quad i = 1, 2, \quad \epsilon \to 0. \tag{27}$$

Expressions (26) and (27), provide an additional condition, on the asymptotic behavior of the components of the stabilization matrix $\bar{\tau}^I$ that should be satisfied for $\epsilon \to 0$. This condition can be written as,

$$\begin{bmatrix} O(1) & O(\epsilon^{-1}) & O(\epsilon^{-1}) \\ O(\epsilon^{-1}) & O(1) & 0 \\ O(\epsilon^{-1}) & 0 & O(1) \end{bmatrix} \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{12} & \tau_{22} & \tau_{23} \\ \tau_{13} & \tau_{23} & \tau_{33} \end{bmatrix} \begin{bmatrix} O(\epsilon^{-1}) \\ O(1) \\ O(1) \end{bmatrix} \sim \begin{bmatrix} O(\epsilon^{-1}) \\ O(1) \\ O(1) \end{bmatrix}. \tag{28}$$

2.5.1 Asymptotic behavior of τ using standard definitions

We consider now the standard definition of the stabilization matrix, given in (22), applied to our simplified problem (23), and show that it fails to satisfy condition (28). Specifically, for one dimensional problems, the standard stabilization (22) is, for $p = 1$,

$$\bar{\tau}_{1d}^I = \kappa \left|\mathbf{C}_{1d}^I\right|^{-1},$$

where κ, is a constant proportional to the element size. Based on the above assumed behavior of the solution, it can be shown that for $\epsilon \to 0$,

$$\bar{\tau}_{1d}^I \sim \begin{bmatrix} O(\epsilon) & O(\epsilon^2) \\ O(\epsilon^2) & O(\epsilon) \end{bmatrix}. \tag{29}$$

It can easily be verified that this stabilization matrix, does not satisfy the condition (28), which in the one dimensional case is

$$\begin{bmatrix} O(1) & O(\epsilon^{-1}) \\ O(\epsilon^{-1}) & O(1) \end{bmatrix} \begin{bmatrix} \bar{\tau}_{11} & \bar{\tau}_{12} \\ \bar{\tau}_{12} & \bar{\tau}_{22} \end{bmatrix} \begin{bmatrix} O(\epsilon^{-1}) \\ O(1) \end{bmatrix} \sim \begin{bmatrix} O(\epsilon^{-1}) \\ O(1) \end{bmatrix}. \tag{30}$$

Therefore, this form of $\bar{\tau}^I$, will fail to provide adequate stabilization when $\epsilon \to 0$. It can also be shown that, in the multi-dimensional case, the standard choice of stabilization matrix (22), does not have the appropriate asymptotic behavior for $\epsilon \to 0$, either.

When working directly with entropy, or conservative variables, the analysis is more complicated because the link between these variables and the incompressible velocity and pressure is less transparent. It is observed in practice that the standard stabilization scheme (22), when used with the equations written in terms of entropy variables, produce solutions which deteriorate severely when the Mach number is reduced. In [8], the use of the standard finite volume upwind scheme, using Roe's dissipation, is shown to give in the incompressible limit, solutions in which the pressure variations do not scale like the Mach number squared. This first order finite volume upwind scheme, can be thought of, at least in one dimension, as the result of using a stabilization matrix of the form (22), directly with conservative variables.

2.6 Alternative definition of $\boldsymbol{\tau}$ for low Mach number flow

Our objective is to derive a stabilization matrix $\boldsymbol{\tau}$, for the variational formulation in entropy variables (18). An approach which has been advocated in the design of low Mach preconditioners for finite volume schemes [23, 6], has been to derive an appropriate stabilization matrix $\bar{\boldsymbol{\tau}}$, for the system (8), and then transform it to conservative variables.

It is apparent that, even with the additional constraint given by condition (28), the stabilization matrix is not uniquely defined and some freedom is still available in constructing it. In addition to the requirements placed on the construction of the standard forms of $\bar{\boldsymbol{\tau}}$, i.e. (22), we place the following three constraints:

i) the entropy equation, which decouples from the other equations, is stabilized as an independent quantity convected with the velocity.

ii) the vorticity equation, which, in the incompressible limit, decouples from the other equations after taking the curl of the velocity evolution equations, is stabilized as an independent quantity convected with the velocity.

iii) the resulting algorithm has the correct low Mach number scaling as discussed in the previous section.

In order to incorporate the above conditions, we consider the Euler equation corresponding to the variational formulation (23), augmented with the entropy equation,

$$\mathbf{C}_1\mathbf{Z}_{,1} + \mathbf{C}_2\,\mathbf{Z}_{,2} = \left(\mathbf{C}_1\bar{\boldsymbol{\tau}}(\mathbf{C}_1\mathbf{Z}_{,1} + \mathbf{C}_2\mathbf{Z}_{,2})\right)_{,1} + \left(\mathbf{C}_2\bar{\boldsymbol{\tau}}(\mathbf{C}_1\mathbf{Z}_{,1} + \mathbf{C}_2\mathbf{Z}_{,2})\right)_{,2}. \tag{31}$$

We shall further assume that, only for the purpose of deriving $\boldsymbol{\tau}$, the matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ are locally constant and therefore the above system of equations can be treated as if it was linear.

Requirement *i*) forces the last row and column of $\bar{\boldsymbol{\tau}}$ to be zero, except for the diagonal entry which corresponds to a velocity time scale. Requirement *ii*), implies that the rest of the matrix must also be of diagonal form. The entries in the second and third rows must be equal and are also associated with a velocity time scale. Therefore, we find that $\bar{\tau}$, for the complete system must be of the form

$$\bar{\tau} = h_e \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & b & 0 \\ 0 & 0 & 0 & b \end{bmatrix}, \tag{32}$$

where $b = 1/|\mathbf{u}|$ and h_e is the size of the element. In order to satisfy the low Mach number scaling, it is clear that $b \sim O(1)$, and therefore a must be $O(\epsilon^2)$. The simplest choice, having the right dimensions, is $a = |\mathbf{u}|/c^2$. For very high speed flows, the stabilization should transition to a pure streamwise upwinding such that $a \to 1/|\mathbf{u}|$. In practice, the specific definition of $\bar{\tau}$ which we use is,

$$\bar{\tau} = \begin{bmatrix} \tau_a & 0 & 0 & 0 \\ 0 & \tau_c & 0 & 0 \\ 0 & 0 & \tau_c & 0 \\ 0 & 0 & 0 & \tau_c \end{bmatrix}, \tag{33}$$

where τ_c is the convective timescale and τ_a is the acoustic timescale. Specifically, for the convective timescale, we employed a form proposed in [17] for scalar convection,

$$\tau_c^{-1} = \sum_{i=1}^{3} \frac{|\mathbf{l}_i \cdot \bar{\mathbf{u}}|}{\mathbf{l}_i \cdot \mathbf{l}_i},$$

where $\bar{\mathbf{u}}$ is the average velocity in the element, and $\mathbf{l}_i$ is the vector between the nodes of side i of the element. We then define the acoustic timescale as,

$$\tau_a^{-1} = \tau_c^{-1} + \frac{c^2}{h_e|\bar{\mathbf{u}}|}.$$

Using these definitions, the acoustic timescale, τ_a, has the correct low Mach number behavior required by the asymptotic analysis. Also, τ_a behaves appropriately for large local Mach numbers where it returns to the convective timescale τ_c.

Finally, the required form of $\boldsymbol{\tau}$ can be obtained by transforming the system (31) to entropy variables. In principle, this could be accomplished by using the transformation matrix $\mathbf{S} = \mathbf{V}_{,\mathbf{Z}}$.

Inserting $d\mathbf{Z} = \mathbf{S}^{-1}d\mathbf{V}$ into (31) and multiplying through by $\mathbf{S}^{-T}$, gives,

$$\begin{aligned}\mathbf{S}^{-T}\mathbf{C}_1\mathbf{S}^{-1}\mathbf{V}_{,1} + \mathbf{S}^{-T}\mathbf{C}_2\mathbf{S}^{-1}\mathbf{V}_{,2} &= \left(\mathbf{S}^{-T}\mathbf{C}_1\mathbf{S}^{-1}\mathbf{S}\bar{\boldsymbol{\tau}}\mathbf{S}^{T}(\mathbf{S}^{-T}\mathbf{C}_1\mathbf{S}^{-1}\mathbf{V}_{,1} + \mathbf{S}^{-T}\mathbf{C}_2\mathbf{S}^{-1}\mathbf{V}_{,2})\right)_{,1} + \\ &\quad \left(\mathbf{S}^{-T}\mathbf{C}_2\mathbf{S}^{-1}\mathbf{S}\bar{\boldsymbol{\tau}}\mathbf{S}^{T}(\mathbf{S}^{-T}\mathbf{C}_1\mathbf{S}^{-1}\mathbf{V}_{,1} + \mathbf{S}^{-T}\mathbf{C}_2\mathbf{S}^{-1}\mathbf{V}_{,2})\right)_{,2}. \end{aligned} \tag{34}$$

Where, the transformed matrix $\boldsymbol{\tau}$, can be readily identified as

$$\boldsymbol{\tau} = \mathbf{S}\,\bar{\boldsymbol{\tau}}\mathbf{S}^{T}. \tag{35}$$

We note that the transformed matrices $\mathbf{S}^{-T}\mathbf{C}_i\mathbf{S}^{-1}, i = 1, 2$, are not equal to $\tilde{\mathbf{A}}_i, i = 1, 2$ and indeed $\mathbf{S}^{-T}\mathbf{S}^{-1}$ is not be equal to $\mathbf{A}_0$. This is due to the fact that the entropy equation in (8), completely decouples from the rest of the system. Therefore, one can multiply the first three components of $d\mathbf{Z}$, i.e. $dp/\rho c, du_1, du_2$, by a scalar factor, and the fourth component of $d\mathbf{Z}$, i.e. ds, by different scalar factor without changing the jacobian matrices $\mathbf{C}_i, i = 1, 2$, or $\bar{\boldsymbol{\tau}}$, in (31). It is not hard to find the scalar factors that one should use to define modified $d\mathbf{Z}$ variables so that the resulting $\mathbf{S}$ matrix would give $\mathbf{S}^{-T}\mathbf{C}_i\mathbf{S}^{-1} = \tilde{\mathbf{A}}_i, i = 1, 2$. An alternative procedure for evaluating $\mathbf{S}$, which avoids the use of modified variables is given in [1]. Once the appropriate transformation matrix has been evaluated, $\boldsymbol{\tau}$ is found using expression (35).

2.7 Numerical results

In this section we present some numerical results that illustrate the performance of the proposed algorithm. For all the examples, we have solved problem (18), which is expressed in terms of entropy variables. The non-linear set of equations resulting from the discretization of (18) is solved by a Newton-Raphson iteration using exact linearization. For some of the simulations however, it was necessary to damp the Newton-Raphson iteration during the first few iterations. The solution of the non-symmetric non-linear system of equations required for each iteration was solved using an enhanced BiCGstab(2) algorithm [7, 21] together with a block ILU(k) preconditioning, with k either 0 or 1. For all the examples we have considered the two choices of entropy variables given in [1], and we have not found any appreciable differences in the computed results. We have followed [2], and employed a linear representation of the solution over each triangular element, but have used a quadratic mapping to more accurately represent the geometry. We have found that using quadratic interpolation of the geometry greatly improves the quality of the solution and only incurs a minimal incremental cost.

2.7.1 Example 1: Flow over cylinder

In this example, we consider the flow about a cylinder and perform several numerical tests. Because of the symmetry of the problem at the low Mach numbers considered here, only half of the domain is considered for the solution. We have used a sequence of three meshes of triangular elements containing 1271, 4941, and 19481 nodes, respectively, to discretize to computational domain. The medium and fine meshes are obtained by uniformly subdividing the coarse mesh. Figure 4 shows a detail of the medium size mesh near the cylinder.

In the first test, we consider the SUPG algorithm (18), with the standard choice of $\boldsymbol{\tau}$ given by (22) for $p = 1$, and solve for the flow about the cylinder on the coarse mesh for free stream Mach numbers of 0.38, 0.1 and 0.01. Figure 5 shows the pressure contours for the three solutions. The degradation in the solution accuracy when the Mach number is reduced is clearly apparent. We could not obtain any numerical solutions below a free stream Mach number of 0.01, and for this Mach number, the iteration would only converge if a damped Newton-Raphson iteration was used. For comparison purposes, we show in figure 6 the pressure contours obtained, for the same flow conditions and mesh, when the proposed form of $\boldsymbol{\tau}$, given in (35), is used. No qualitative degradation of the solution is observed and in fact, we observe that, the solutions for Mach numbers of 0.1 and 0.01 look almost identical, as expected.

In the second test, we perform a mesh convergence analysis at various Mach numbers. Figure (7) shows the Mach number contours computed on the coarse, medium and fine mesh for a free stream Mach number of 0.38. Analogous results, but now for a Mach number of 0.001 are shown in figure (8). The qualitative behavior of the solution, at both Mach numbers, does not present any anomalies. In figure (9), a plot of the solution error norm versus grid size is shown. Here, solutions at Mach numbers of 0.38, 0.01, and 0.001 are compared and second order convergence is observed in all cases, showing no deterioration in the convergence rate as the Mach number is reduced. The norm of the solution error considered was $\left[\int_\Omega (\mathbf{V} - \mathbf{V}^h)\mathbf{A}_0(\mathbf{V} - \mathbf{V}^h) d\Omega\right]^{1/2}$, where $\mathbf{V}$, is a reference solution computed using a quadratic approximation on a highly refined grid of 77361 nodes [26].

2.7.2 Example 2: Flow over an airfoil

In this example, the proposed scheme was used to simulate the flow over NACA 0012 airfoil at Mach numbers of 0.01 and 0.6, and at an angle of attack of 2°. An unstructured triangular mesh of 19948 nodes was used for the simulation. Figure 10 shows a coarser mesh from which the

19948 node mesh can be obtained by uniformly subdividing each element into four elements. The computed Mach number and pressure contours for the two flow conditions are shown in figures 11 and 12, respectively. The qualitative behavior of the solution looks excellent, and again, no solution accuracy or numerical anomalies are observed when very low Mach number flows are considered.

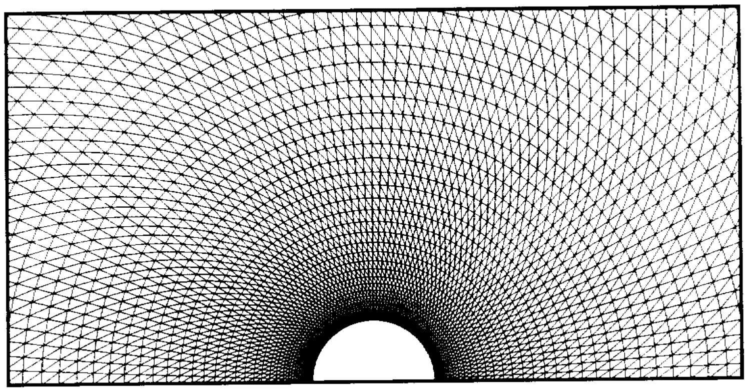
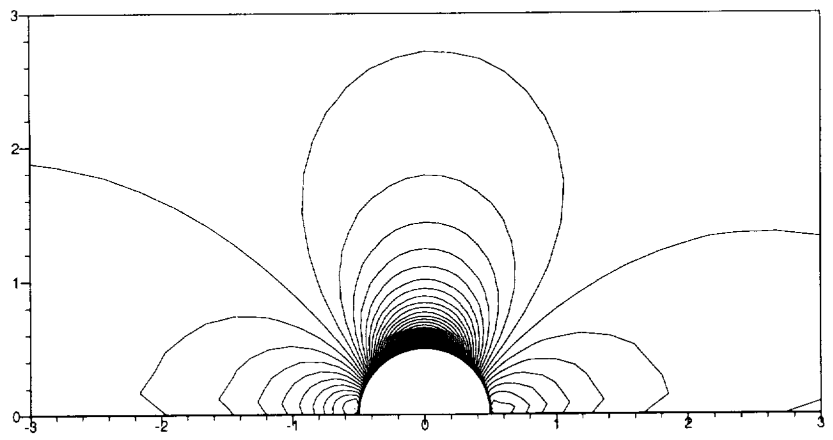


Figure 4: Detail of the medium size mesh used for computations

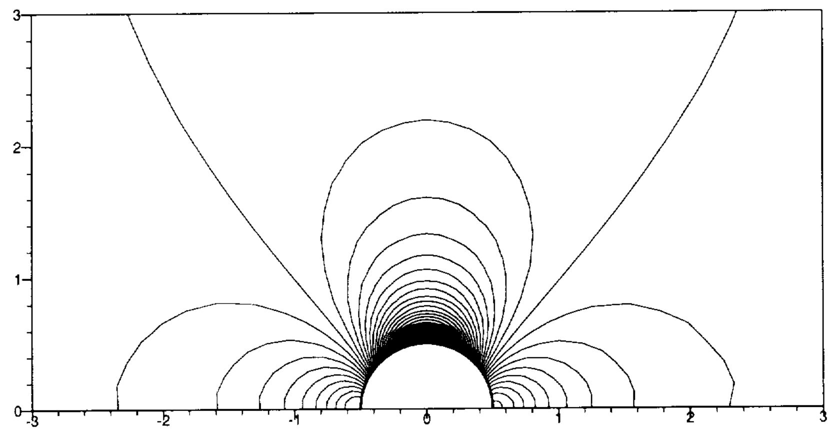
3 High Order Finite Element Discretization of the Compressible Euler and Navier-Stokes Equations

This section centers upon a high-order accurate, stabilized, finite element method for the numerical solution of the compressible Euler and Navier-Stokes equations. The SUPG finite element method for compressible flow simulations was initially developed and analyzed by Hughes *et al.* [14, 15, 13, 20] and has since gained significant popularity. Its relation to multidimensional upwinding was elucidated in [28, 29] and higher order implementations for inviscid flows were presented in [2]. In [1], the SUPG algorithm was extentded to cover the simulation of near-incompressible flows by employing a stabilization matrix which exhibits proper scaling over the entire range of Mach numbers. Here, we focus on the higher order implementation of the algorithm developed in [1] for invicid and viscous flows.

a.



b.



c.

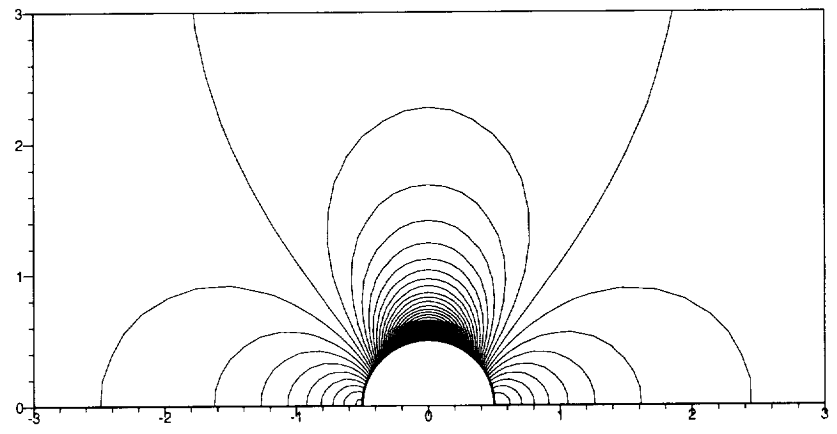


Figure 5: Static pressure contours computed on the coarse mesh using the SUPG algorithm with the standard choice of stabilization matrix given by (22), for a free stream Mach number of a) 0.01, b) 0.1, and c) 0.38.

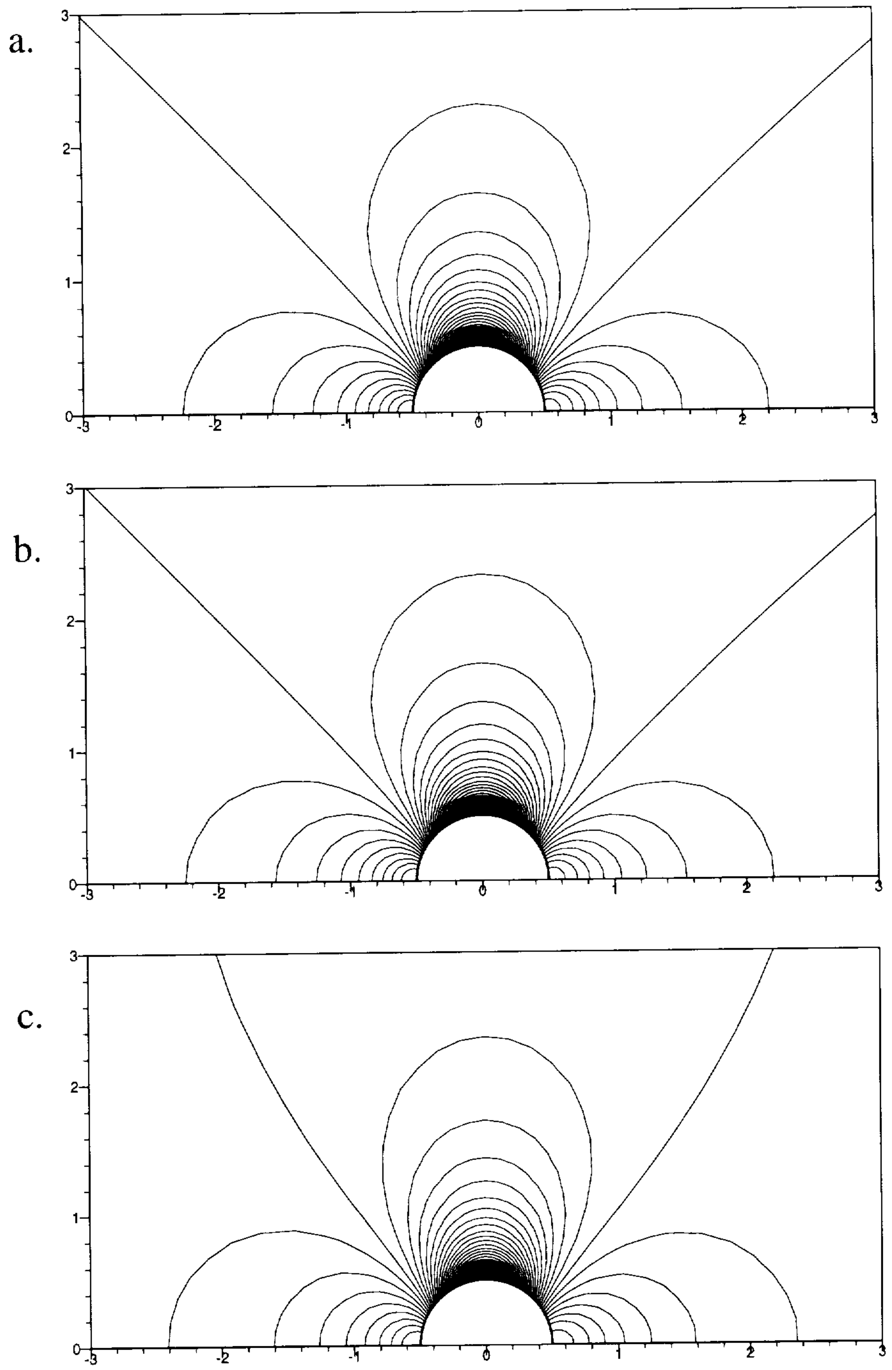


Figure 6: Static pressure contours computed on the coarse mesh using the SUPG algorithm with the proposed choice of stabilization matrix given by (35), for a free stream Mach number of a) 0.01, b) 0.1, and c) 0.38.

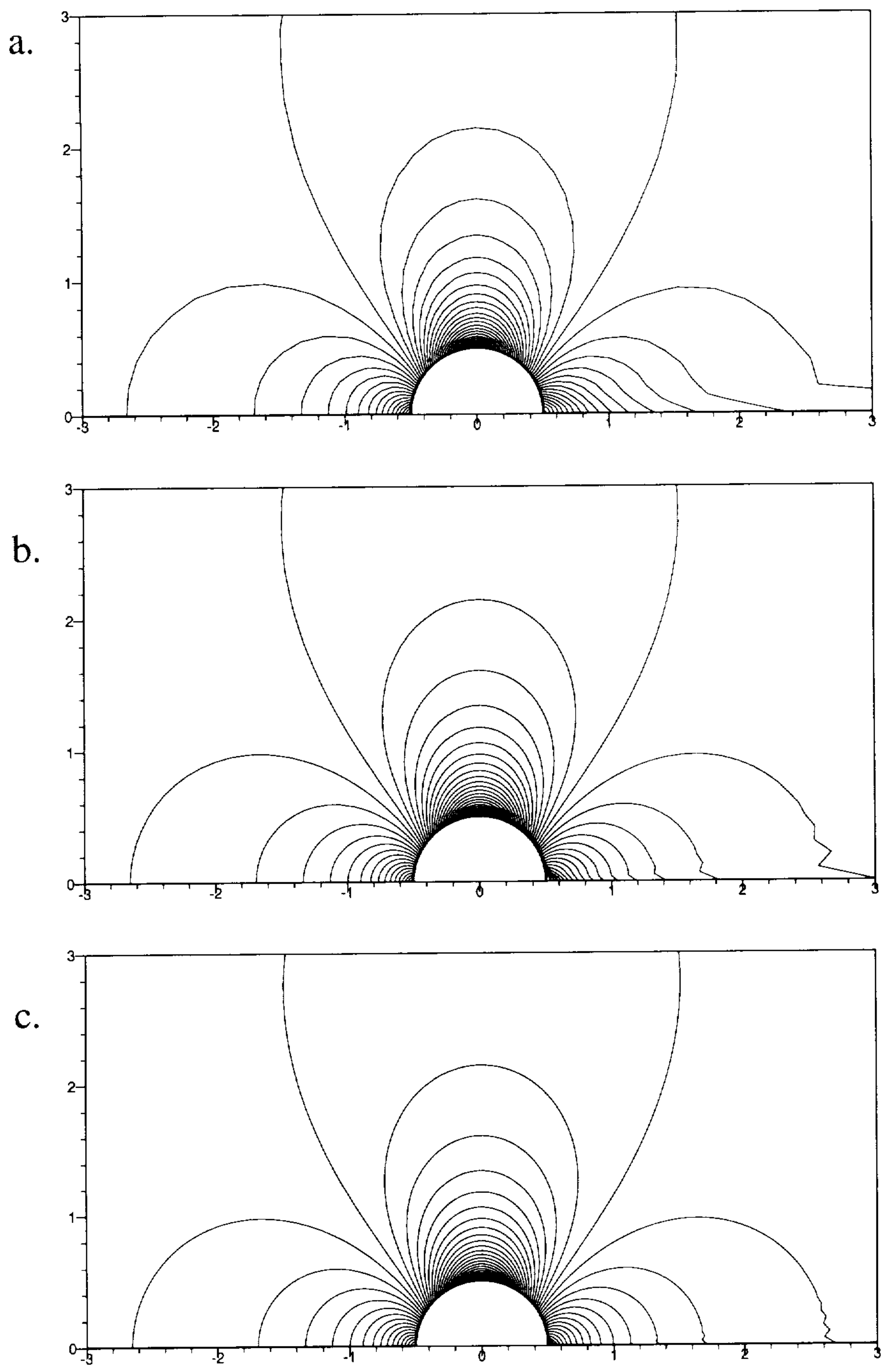


Figure 7: Mach contours for a free stream Mach number of 0.38 computed on the : a) coarse, b) medium, and c) fine meshes.

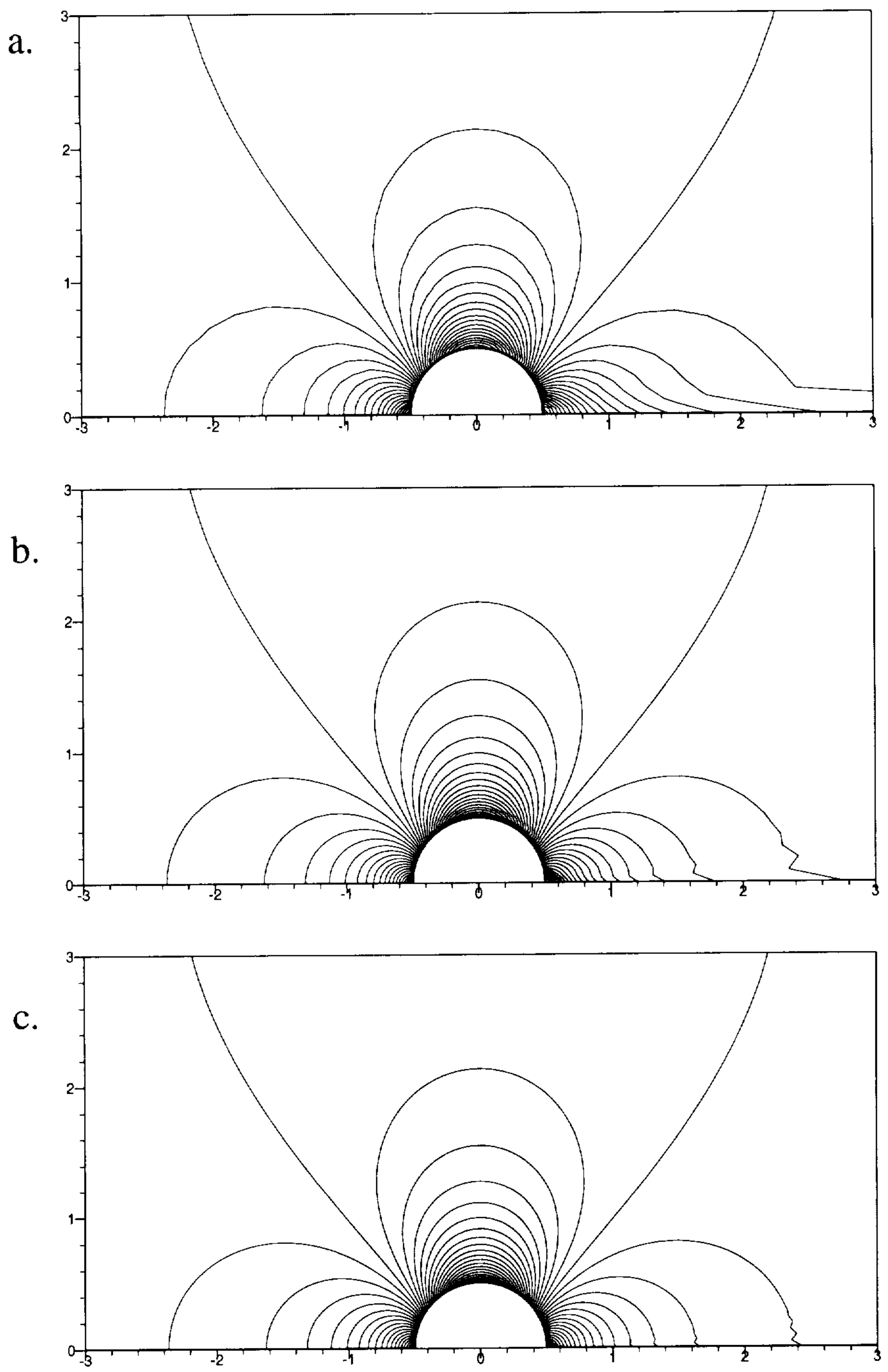


Figure 8: Mach contours for a free stream Mach number of 0.001 computed on the : a) coarse, b) medium, and c) fine meshes.

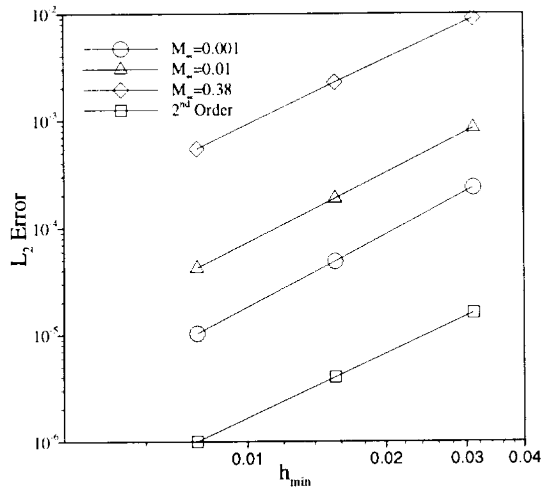


Figure 9: Grid convergence plot showing the solution error norm versus the grid size parameter using the proposed algorithm and for free stream Mach numbers of 0.001, 0.01 and 0.38.

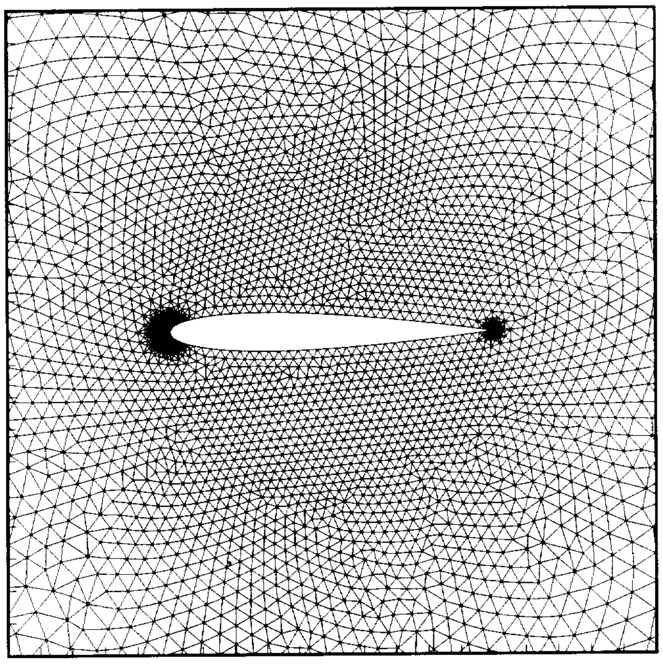
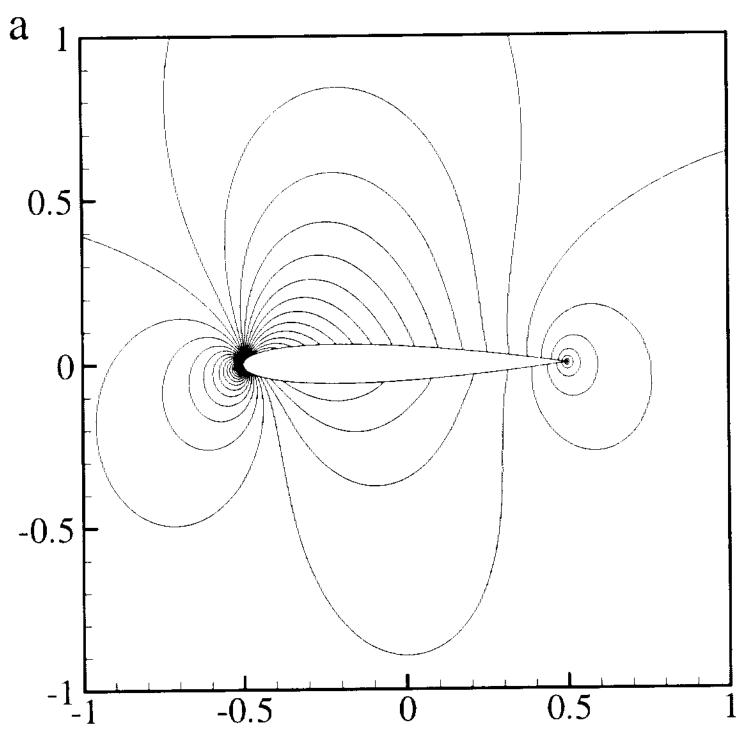


Figure 10: Detail of the unstructured triangular the mesh used for the airfoil computations



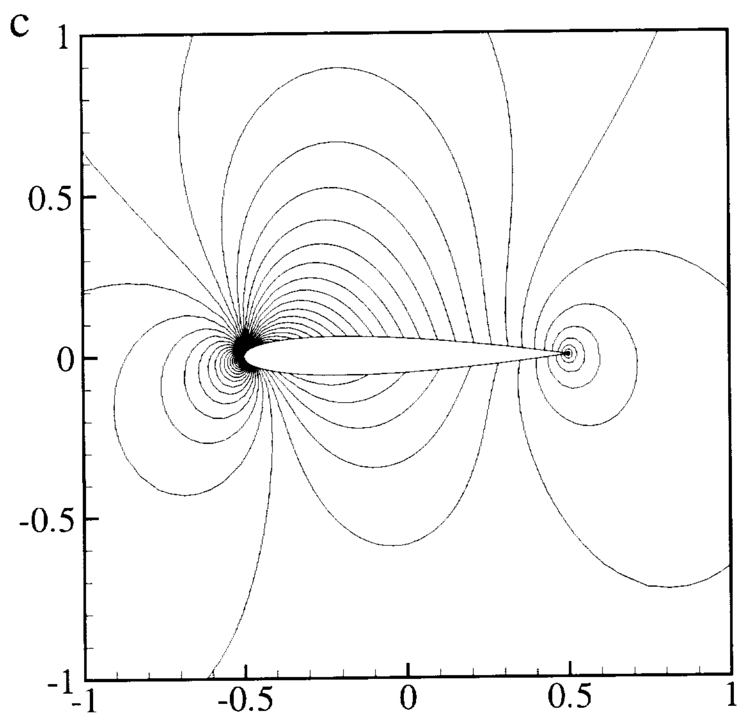
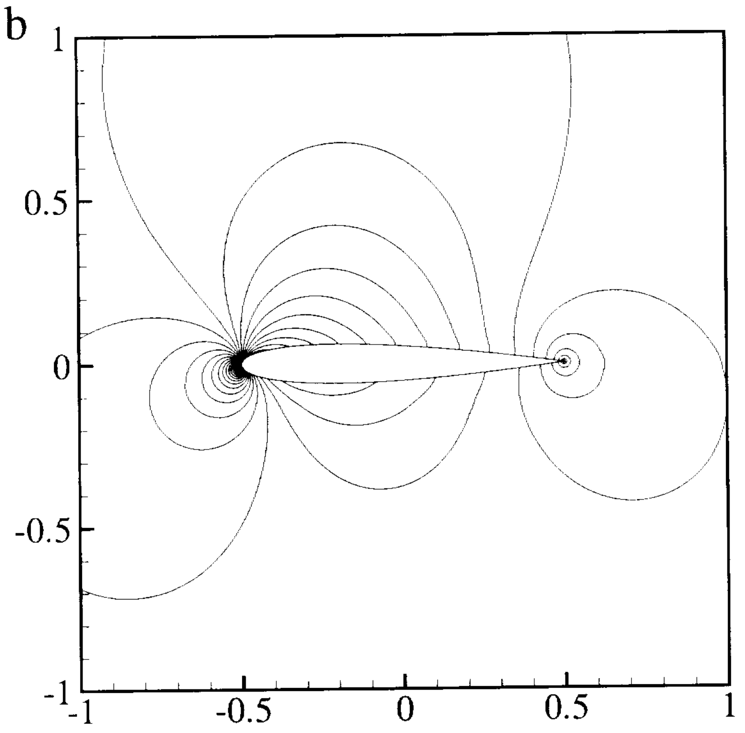


Figure 11: Mach number contours for the solution of the flow over NACA 0012 at an angle of attack of 2° for a free stream Mach number of : a) 0.01, and b) 0.6.



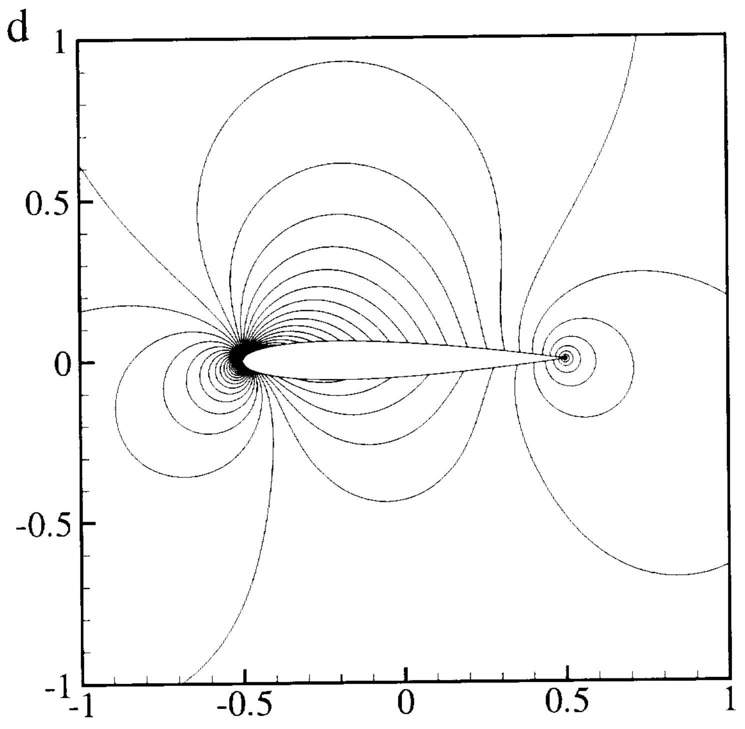


Figure 12: Static pressure contours for the solution of the flow over NACA 0012 at an angle of attack of 2° for a free stream Mach number of : a) 0.01, and b) 0.6.

3.1 Compressible flow governing equations

We start from the time dependent two dimensional compressible Euler equations in conservation form

$$\mathbf{U}_{,t} + (\mathbf{F} - \mathbf{F}^v)_{1,1} + (\mathbf{F} - \mathbf{F}^v)_{2,2} = 0, \tag{36}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix}, \quad \mathbf{F}_1 = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ u_1(\rho E + p) \end{bmatrix}, \quad \mathbf{F}_2 = \begin{bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ u_2(\rho E + p) \end{bmatrix}.$$

and

$$\mathbf{F}_1^v = \begin{bmatrix} 0 \\ \tau_{11} \\ \tau_{12} \\ u_1\tau_{11} + u_2\tau_{12} + q_1 \end{bmatrix}, \quad \mathbf{F}_2^v = \begin{bmatrix} 0 \\ \tau_{21} \\ \tau_{22} \\ u_1\tau_{21} + u_2\tau_{22} + q_2 \end{bmatrix}.$$

In the above expressions, ρ is the density; $\mathbf{u} = [u_1, u_2]^T$ is the velocity vector; E is the specific total energy; p is the pressure; and the comma denotes partial differentiation (e.g. $\mathbf{U}_{,t} = \partial\mathbf{U}/\partial t$, the partial derivative with respect to time, $\mathbf{F}_{i,j} = \partial\mathbf{F}_i/\partial x_j$, the partial derivative with respect to the j-th spatial coordinate). The system of equations is closed once the pressure is related to the problem variables through the equation of state, $p = (\gamma - 1)\rho e$, where $e = E - |\mathbf{u}|^2/2$, is the internal energy. Here, γ is the ratio of specific heats and μ is the absolute viscosity, both of which are assumed to be constant. Following the usual assumptions:

$$\tau_{11} = 2\frac{\mu}{Re}\frac{\partial u_1}{\partial x_1} - \frac{2}{3}\frac{\mu}{Re}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right)$$

$$\tau_{12} = \tau_{21} = \frac{\mu}{Re}\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right)$$

$$\tau_{22} = 2\frac{\mu}{Re}\frac{\partial u_2}{\partial x_2} - \frac{2}{3}\frac{\mu}{Re}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right)$$

and

$$q_1 = -\frac{\mu}{RePr}\frac{\partial T}{\partial x_1}, \quad q_2 = -\frac{\mu}{RePr}\frac{\partial T}{\partial x_2}.$$

We will assume that all the above quantities have been non-dimensionalized using reference, or free stream, values for density ρ_*, velocity u_*, and length L. Thus, the dimensional variables, denoted with an overbar, are related to the non-dimensional variables introduced above as

$$\rho = \frac{\bar{\rho}}{\rho_*}, \quad u_i = \frac{\bar{u}_i}{u_*}, \; i = 1, 2, \quad p = \frac{\bar{p}}{\rho_* u_*^2}, \quad E = \frac{\bar{E}}{u_*^2}, \quad \mu = \frac{\bar{\mu}}{\mu_*}, \quad x_i = \frac{\bar{x}_i}{L}, \; i = 1, 2, \quad \text{and} \quad t = \frac{u_* \bar{t}}{L}.$$

We note that the equation system (36) can be written as

$$\mathbf{U}_{,t} + \mathbf{A}_1 \mathbf{U}_{,1} + \mathbf{A}_2 \mathbf{U}_{,2} = (\mathbf{K}_{11} \mathbf{U}_{,1})_{,1} + (\mathbf{K}_{12} \mathbf{U}_{,2})_{,1} + (\mathbf{K}_{21} \mathbf{U}_{,1})_{,2} + (\mathbf{K}_{22} \mathbf{U}_{,2})_{,2}, \tag{37}$$

where the Jacobian matrices $\mathbf{A}_i = \mathbf{F}_{i,\mathbf{U}}$, $i = 1, 2$, are unsymmetric but have real eigenvalues and a complete set of eigenvectors. $\mathbf{K}_{ij} = \mathbf{F}^v_{i,\mathbf{U}_{,j}}$ are the viscous flux jacobians. The above equation may be symmetrized through a change of variables, for details, we refer the reader to [13, 12].

3.2 Variational formulation for the steady state problem

We now consider the compressible steady problem in conservation form expressed in terms of symmetrizing variables. The conservative form of the equations is taken to be the starting point because we are ultimately interested in an algorithm that can be used over the whole range of speed regimes, including situations were the solution may contain discontinuitites. The problem is defined in a domain Ω with boundary Γ by

$$(\mathbf{F} + \mathbf{F}^v)_1 (\mathbf{V})_{,1} + (\mathbf{F} + \mathbf{F}^v)_2 (\mathbf{V})_{,2} = 0 \quad \text{in} \quad \Omega, \tag{38}$$

$$\tilde{\mathbf{A}}_n^- \mathbf{V} = \tilde{\mathbf{A}}_n^- \mathbf{g} \quad \text{on} \quad \Gamma \backslash \Gamma_a, \tag{39}$$

$$\mathbf{F}^v \cdot \mathbf{n} = \mathbf{f} \quad \text{on} \quad \Gamma_a \tag{40}$$

For simplicity, the domain boundary is assumed to be made up of a solid wall Γ_a, and a computational far field boundary $\Gamma \backslash \Gamma_a$. In (39, 40), $\mathbf{n} = [n_1, n_2]^T$ is the outward unit normal vector to Γ, and $\tilde{\mathbf{A}}_n = \mathbf{A}_n \mathbf{A}_0$, $\mathbf{A}_n = \mathbf{A}_1 n_1 + \mathbf{A}_2 n_2$. Finally, $\tilde{\mathbf{A}}_n^- = \mathbf{A}_n^- \mathbf{A}_0$, and $\mathbf{A}_n^-$ denotes the negative definite part of $\mathbf{A}_n$.

Let the spatial domain Ω, be discretized into non-overlapping elements T_e, such that $\Omega = \bigcup T_e$, and $T_e \bigcap T_{e'} = \emptyset$, $e \neq e'$. We consider the space of functions $\mathcal{V}_h$, defined over the discretization and consisting of the continuous functions which are piecewise linear over each element

$$\mathcal{V}_h = \{ \mathbf{W} \mid \mathbf{W} \in (C^0(\Omega))^4, \; \mathbf{W}|_{T_e} \in (\mathcal{P}_k(T_e))^4, \; \forall T_e \in \Omega \}.$$

The SUPG algorithm can then be written as: Find $\mathbf{V}_h \in \mathcal{V}^h$ such that for all $\mathbf{W} \in \mathcal{V}^h$,

$$B(\mathbf{V}_h, \mathbf{W})_{gal} + B(\mathbf{V}_h, \mathbf{W})_{supg} + B(\mathbf{V}_h, \mathbf{W})_{bc} = 0, \tag{41}$$

where the forms $B(\cdot,\cdot)_{gal}$, $B(\cdot,\cdot)_{supg}$ and $B(\cdot,\cdot)_{bc}$ account for the Galerkin, SUPG stabilization, and boundary condition terms respectively, and are defined as

$$B(\mathbf{V}, \mathbf{W})_{gal} = \int_\Omega (-\mathbf{W}_{,1} \cdot (\mathbf{F} - \mathbf{F}^v)_1(\mathbf{V}) - \mathbf{W}_{,2} \cdot (\mathbf{F} - \mathbf{F}^v)_2(\mathbf{V}))\, d\Omega, \tag{42}$$

$$B(\mathbf{V}, \mathbf{W})_{supg} = \int_\Omega (\tilde{\mathbf{A}}_1 \mathbf{W}_{,1} + \tilde{\mathbf{A}}_2 \mathbf{W}_{,2}) \cdot \boldsymbol{\tau}\, (\tilde{\mathbf{A}}_1 \mathbf{V}_{,1} + \tilde{\mathbf{A}}_2 \mathbf{V}_{,2} - (\mathbf{K}_{11}\mathbf{V}_{,1})_{,1} - (\mathbf{K}_{12}\mathbf{V}_{,2})_{,1} - (\mathbf{K}_{21}\mathbf{V}_{,1})_{,2} - (\mathbf{K}_{22}\mathbf{V}_{,2})_{,2})\, d\Omega, \tag{43}$$

and

$$B(\mathbf{V}, \mathbf{W})_{bc} = \int_{\Gamma \backslash \Gamma_a} \mathbf{W} \cdot (\mathbf{F}_{ff} + \mathbf{F}^v)(\mathbf{V}, \mathbf{g}; \mathbf{n})\, ds. + \int_{\Gamma_a} \mathbf{W} \cdot \mathbf{F}^v(\mathbf{V}, \mathbf{f}; \mathbf{n})\, ds. \tag{44}$$

where $\boldsymbol{\tau}$ is the stabilization matrix. The numerical flux function on the far field boundary $\mathbf{F}_{ff}$, is defined by

$$\mathbf{F}_{ff}(\mathbf{V}_-, \mathbf{V}_+; \mathbf{n}) = \frac{1}{2}(\mathbf{F}_n(\mathbf{V}_-) + \mathbf{F}_n(\mathbf{V}_+)) - \frac{1}{2}|\mathbf{A}_n(\mathbf{V}^*(\mathbf{V}_-, \mathbf{V}_+))|(\mathbf{V}_+ - \mathbf{V}_-).$$

Here, $|\mathbf{A}_n(\mathbf{V})| = \mathbf{A}_n^+(\mathbf{V}) - \mathbf{A}^-(\mathbf{V})$ is the absolute value of $\mathbf{A}_n$ evaluated at $\mathbf{V}^*$, and $\mathbf{V}^*(\mathbf{V}_+, \mathbf{V}_-)$, is the an average between the states $\mathbf{V}^+$ and $\mathbf{V}^-$. The average state, $\mathbf{V}^*$, is chosen to ensure the global stability of the algorithm [2]. The acquisition of $\mathbf{V}^*$ requires iteration and in practice an arithmetic average may be used. The Roe flux [19] was used in all the numerical simulations presented herein. For inviscid compuations, the viscous terms in the expressions above would of course, vanish. For viscous simulations, Dirichlet boundary conditions may replace portions of the boundary integral.

3.3 Numerical results

In this section we present some numerical results that illustrate the performance of the proposed algorithm. Test problems were solved employing both linear and quadratic element approximations. For comparative purposes, the meshes used for all linear element approximations were obtained by subdividing each element of the corresponding quadratic element mesh into four linear elements. In this way, comparisons between P_1 and P_2 solutions involving the same number of nodes can be made. h_c thus represents the distance between two nodes in the meshes used in the numerical simulations presented herein.

3.3.1 Example 1: Rinleb flow

In this example, we consider a ringleb test case (an exact solution of the Euler equations, [27]). The error is computed in the L_2 entropy norm [1]. Both P_1 and P_2 element approximation achieved their respective optimal convergence rate of $O(h^2)$ and $O(h^3)$ respectively, as can be seen in figure 13.

3.3.2 Example 2: Flow over an airfoil

In this example, the proposed scheme was used to simulate the flow over NACA 0012 airfoil at a Mach number of 0.6, and at an angle of attack of 2°. In figure 14, the L_2 entropy deviation for both P_1 and P_2 simulations are presented. The quadratic element approximation results in a much lower level of entropy error than its linear element counterpart. The geometric singularity at the trailing edge of the airfoil requires a much finer discretization around that point relative to the rest of the mesh for both the linear and quadratic element approximations to achieve their optimal convegence rate. This is particularly important for the P_2 approximation since the error away from the geometric singularity vanishes far quicker, rendering the trailing edge error as the dominant source of error for the numerical approximation. Figure 15, shows the computed Mach contours for the p_1 and P_2 simulations.

3.3.3 Example 3: Flow over flat plate

In this example, we consider flow over a flat plate of unit length. The computational domain is $[-1.5, 1] \times [0, 1]$, with the leading edge of the plate at $(0, 0)$. The free stream Mach number is 0.5. The Reynolds number is raised from 8000 to 64000 in successive simulations while keeping the mesh unchanged. The results in the form of boundary layer thickness, $\delta_{99}(x = L)$, are plotted in figure 16. The quadratic element approximation yields results very close to that of the Blasius solution while the linear element approximation shows increasing error with rising Reynolds number. Further numerical tests have shown that it is possible to resolve the Blasius boundary layer with only two elements when quadratic element approximation is used.

3.3.4 Example 4: Transonic computations

Here we show the application of the algorithm proposed using P_2 interpolations to the inviscid computation of the transonic flow over a NACA0012 airfoil at a mach number of 0.8 and and angle of attack of 1.2^o degrees 17. The shock capturing term employed is that reported in [2]. This results are only preliminary but illustrate the ability of the high order finite element algorithm to capture shocks.

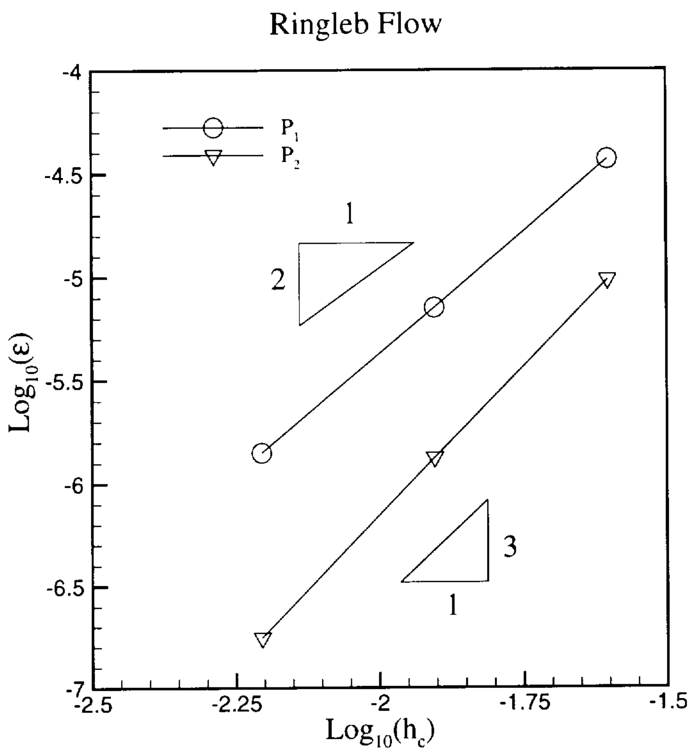
4 An Algebraic Multigrid Method for Convection-Difsusion Flows

Rapid advances in unstructured mesh methods for computational fluid dynamics (CFD) have been made in recent years and, for the computation of inviscid flows, have achieved a considerable level of maturity. Viscous flow technology is also rapidly developing and the use of unstructured grids has been indispensable. Unstructured meshes offer a practical means for computation and have the advantages that they provide both flexible approximations of the domain geometry and easy adaptation/refinement of the mesh.

Accurate and efficient solutions to the compressible Navier-Stokes equations, especially in the turbulent high Reynolds number limit, remains one of the most challenging problems due to the myriad of associated length scales required to properly resolve flow features. This is especially true in the boundary layer regions where severe grid anisotropy is required. Discretization of the partial differential equations on the mesh gives rise to a large linear system of equations. For 3D problems, these large discrete problems often cannot be solved using direct solution methods. As a result, iterative solution methods based on Krylov subspace methods and/or multilevel methods, which include multigrid and domain decomposition methods, are attractive. Multilevel methods can often provide mesh independent convergence rates [31] and offer good scaling of the compute time as well as data storage requirements. In the typical context, multilevel methods are not used as solvers but as preconditioners for Krylov subspace iterative solvers. This provides a powerful and flexible framework for computation.

There are two major problems associated with AMG for the solution of convection diffusion flows. The first is the definition of accurate coarse spaces in the multilevel construction. This is required to properly capture the behaviour of the discretized equations on these coarse spaces. The second problem is the behaviour of the smoothing operators with high Reynolds number and anisotropic effects. These two effects are typically the leading causes of convergence rate deterioration.

In this section, we present a multigrid methodology for the solution of convection-diffusion based problems, especially in the finite element context. The target application for this algorithm is high



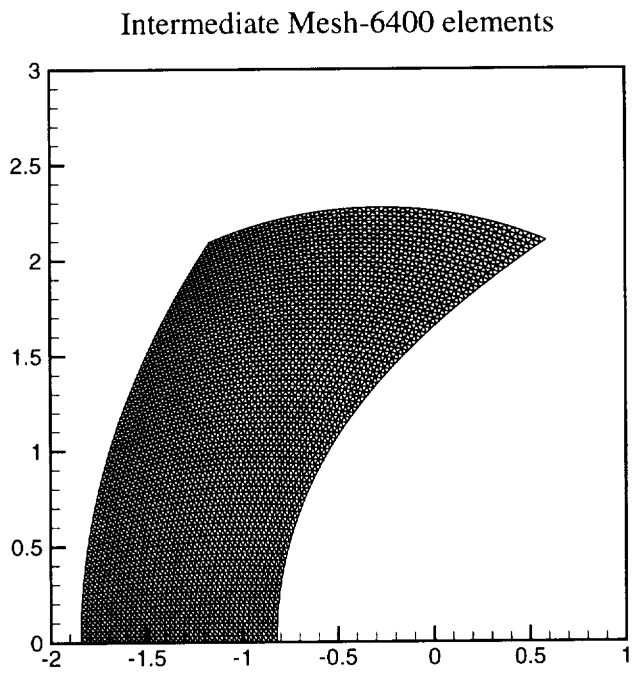
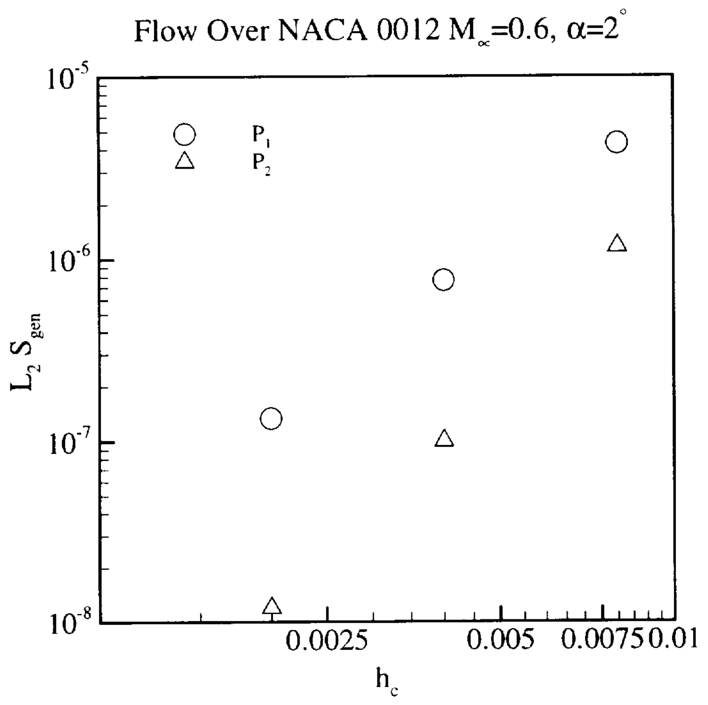


Figure 13: Top: L_2 entropy norm error of ringleb flow solution, Bottom: Computational mesh.



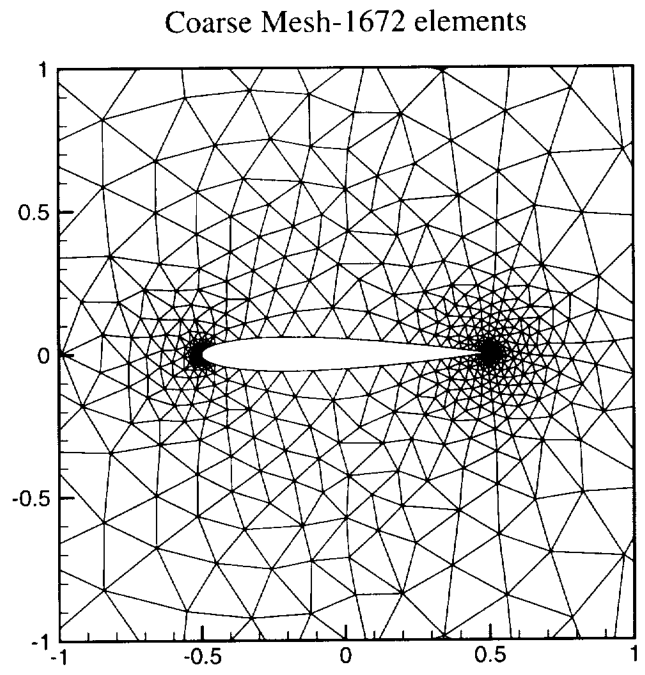
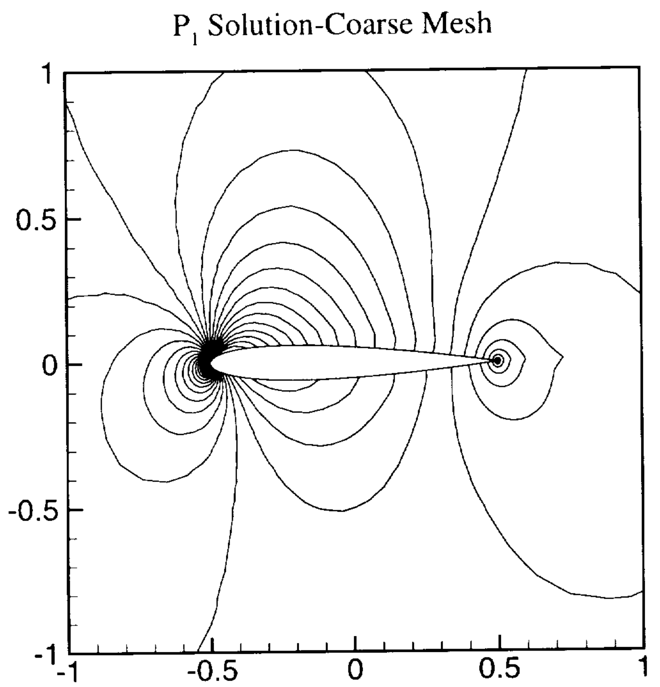


Figure 14: Top: L_2 entropy error of flow over NACA 0012 airfoil, Bottom: Computational mesh.



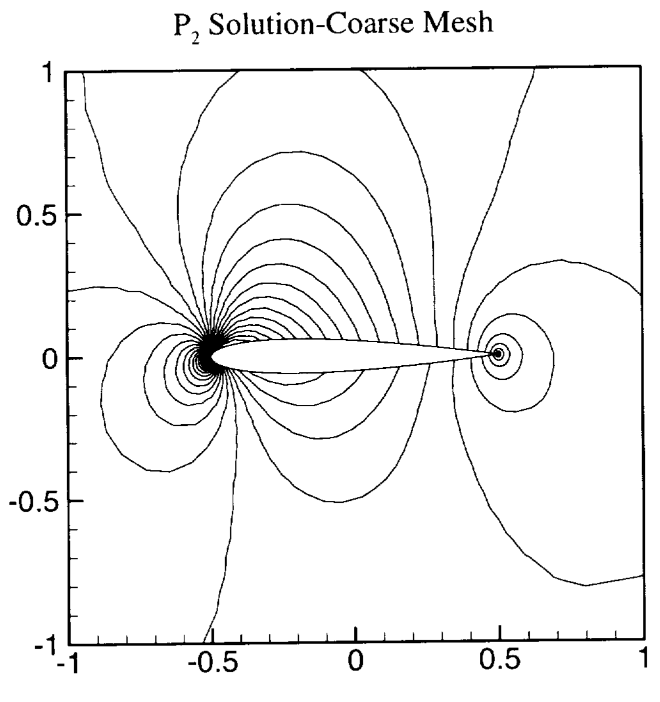
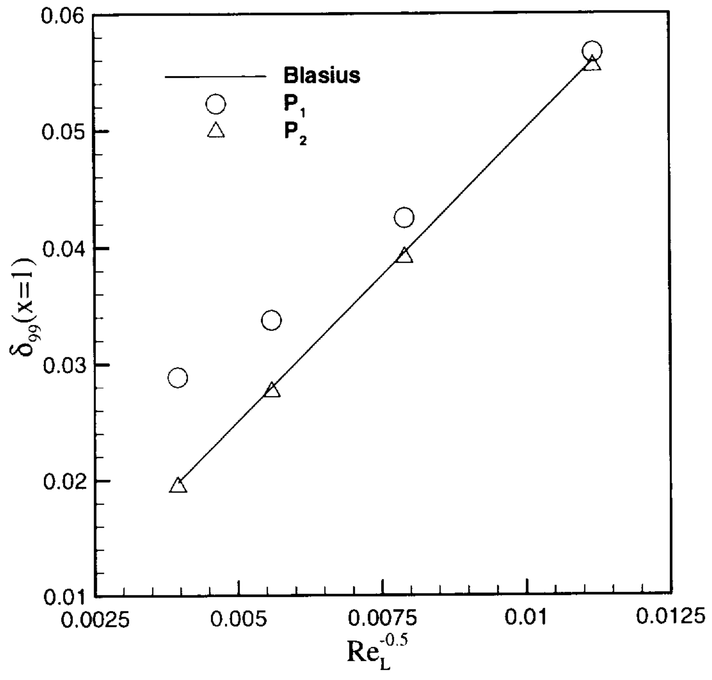


Figure 15: Mach contour of flow over NACA 0012 airfoil, $M_\infty = 0.6$, $\alpha = 2^\circ$



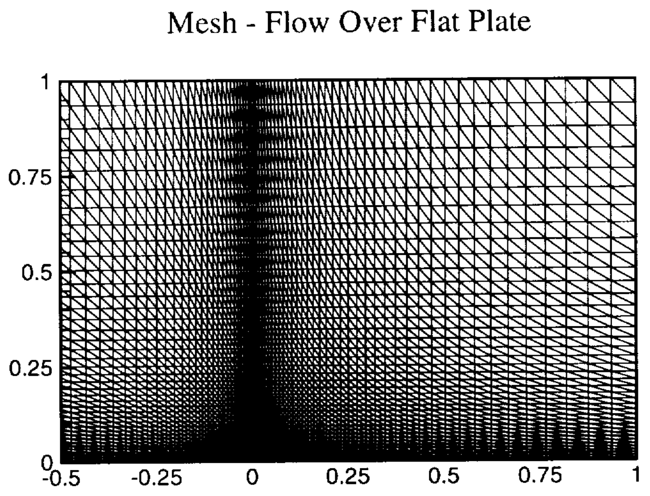
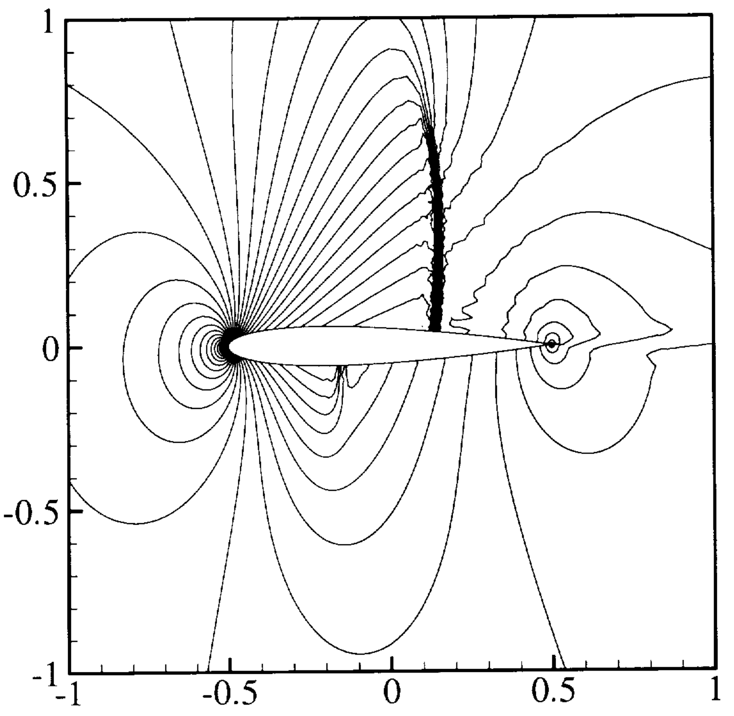


Figure 16: Top: Computed value of boundary layer thickness at $x = 1$, Bottom: Computational mesh.



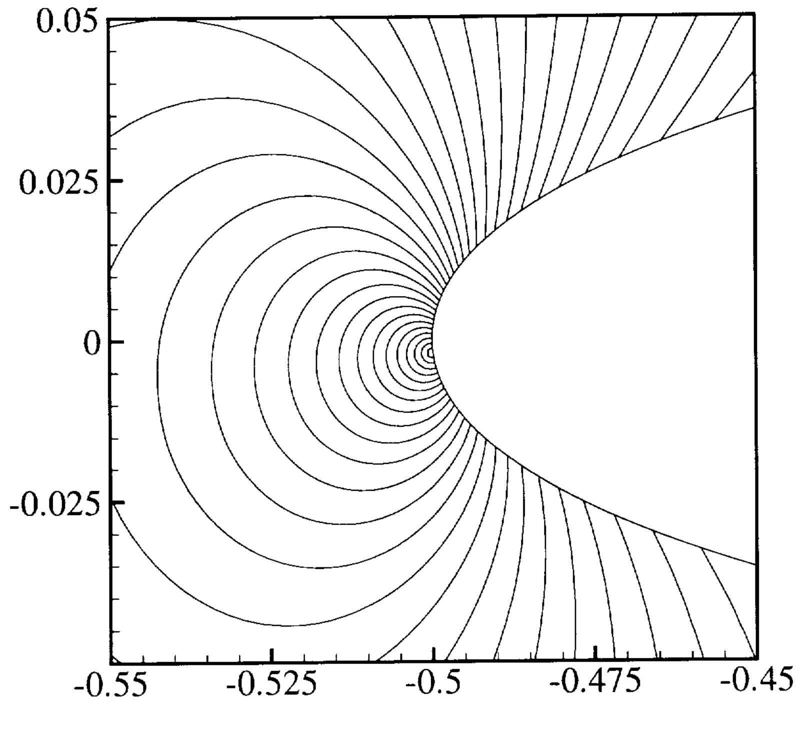


Figure 17: Computed Mach contours for the flow about a NACA0012 airfoil at $M = 0.8$ and $\alpha = 1.25^o$ value of boundary layer thickness at $x = 1$, .

Reynolds number Navier-Stokes flows.

4.1 Model problem

The model problem considered is the linear convection diffusion problem

$$\begin{aligned} \mathbf{U}\cdot\nabla\phi &= \nu\nabla^2\phi + f && \text{in } \Omega, && (45)\\ \phi &= \alpha_D && \text{on } \partial\Omega_D, && (46)\\ \frac{\partial\phi}{\partial\eta} &= \alpha_N && \text{on } \partial\Omega_N, && (47) \end{aligned}$$

where Ω is a bounded domain in $I\!R^n$ $(n = 1,2,3)$ with boundary $\partial\Omega$, and $\mathbf{U} = (U_1, \ldots, U_n)$ is an incompressible prescribed velocity field. The finite element discretization is based on the stabilized SUPG formulation analogous to that described in the previous sections. Introducing the variational form of Eq. 45, the problem reduces to finding $\phi \in \mathrm{H}_0^1(\Omega;\partial\Omega_\mathrm{D})$ such that

$$a(\phi, v) = F(v) \qquad (48)$$

where

$$\begin{aligned} a(\phi, v) &= \int_\Omega \tilde{v}(\mathbf{U}\cdot\nabla\phi)\, d\Omega + \int_\Omega \nu\nabla\phi\nabla\tilde{v}\, d\Omega && (49)\\ F(v) &= \int_\Omega f\tilde{v}\, \mathbf{dx} && (50)\\ \tilde{v} &= v + \tau\mathbf{U}\cdot\nabla v && (51) \end{aligned}$$

$\mathrm{H}_0^1(\Omega;\partial\Omega_\mathrm{D})$ is the Sobolev space which contains functions that vanish on $\partial\Omega_D$ with square integrable first derivatives, $\tilde{v}$ is the stabilized SUPG test function and τ is the SUPG stabilization parameter. The discretization of the problem is done by covering Ω with non-overlapping finite elements through a triangulation and defining standard linear basis functions over these elements. The discrete problem now reduces to:

Find $\phi_h \in V_h$ such that

$$a(\phi_h, v_h) = F(v_h) \qquad \forall v_h \in V_h \qquad (52)$$

where V_h is the finite dimensional subspace of $\mathrm{H}_0^1(\Omega;\partial\Omega_\mathrm{D})$ consisting of continuous functions which are linear over the elements. This results in a system of linear equations

$$\mathbf{A}\phi = \mathrm{b} \qquad (53)$$

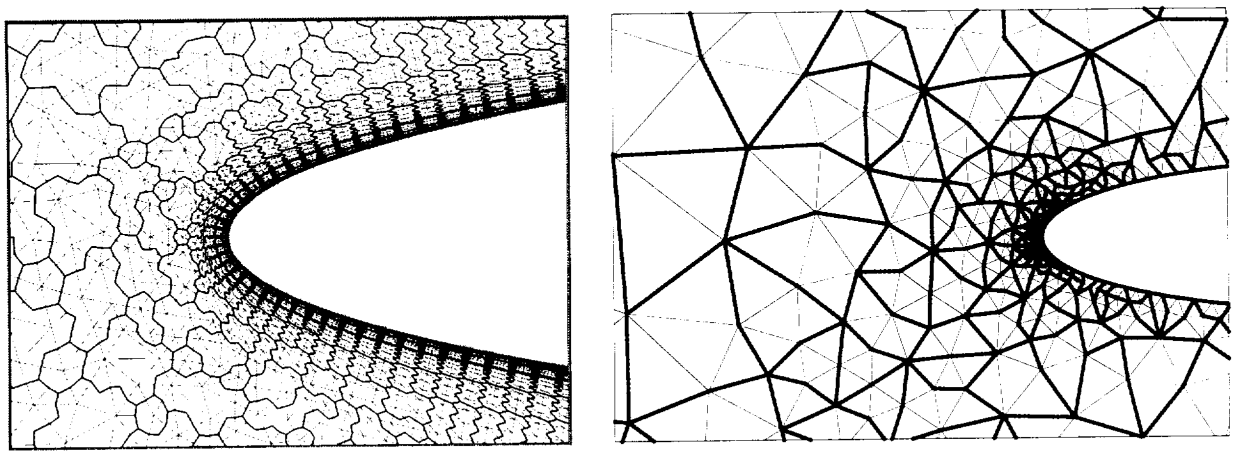
which need to be solved for the discrete solution ϕ_h.

In order to obtain accurate results which provide detailed resolution of the flow field, a well resolved mesh is required which gives rise to a large linear system. In general, iterative methods give very good performance when the underlying matrix A is symmetric and positive definite (SPD). However, the resulting matrices from the discretization of convection-diffusion problems are not SPD. This is usually due to the anisotropic stiffness introduced in the system of equations through the grid and the convective nature of the system.

Here, the development of a multilevel methodology based on algebraic multigrid is discussed. Multigrid has shown great promise in the solution of linear algebraic systems and can be shown to have mesh independent convergence for a wide range of problems. Iterative schemes remove certain types of errors, typically high frequency (*rough*) errors but are unable to damp out the low frequency (*smooth*) components. Multigrid may be used in conjunction with these iterative schemes to form a powerful solver by removing these smooth error components. Representation of these smooth errors on the multigrid coarse spaces means that they appear rough on these spaces where they may be damped out effectively.

The construction of these coarse spaces may be done in several ways and the most obvious one is to simply retriangulate the domain with a larger mesh spacing. This however is a very expensive procedure especially for meshes required for Navier-Stokes simulations which can have very complex geometries. These methods are termed Geometric Multigrid (GM) and they make full use of geometry. Another way is by nodal decimation which involves selection of a vertex subset and retriangulation. The selection process is typically based on fine grid geometry and depends on some pattern in the fine grid [32]. Depending on the pattern, different coarsenings arise. Calculations in the inviscid regions of the mesh use a full coarsening technique which gives a 4:1 reduction in 2D, an example of which is given in Fig. 18(a). However, alleviation of the stiffness due to stretched grids in viscous flow calculations requires semi-coarsening [33] which gives a smaller reduction. We refer to [34] for other references on this.

In contrast to geometric multigrid, another promising avenue is Algebraic Multigrid (AMG) which uses an algebraic definition for the coarse spaces by agglomeration of the finite element subspace on the fine grid [35]. A purely algebraic definition allows for automatic construction of the coarse spaces and does not require geometric information. However, the smoother and the coarsening algorithms need to be carefully matched. The agglomeration technique is typically nodal [36, 37, 38, 39] which results in the Additive Correction Multigrid (ACM) method. However,



(a) Vertex Agglomeration (b) Element Agglomeration

Figure 18: Agglomeration Types

another effective method is through elemental agglomeration which involves the agglomeration of neighbouring elements into macroelements as shown in Fig. 18(b). Hence, the coarse space elements are not standard elements and as such, the coarse space meshes are not proper meshes. Appropriate basis functions as well as transfer operators need to defined. Perhaps the most recent development in this area is by Chan *et al* [34, 31] and the results have been shown to be promising. This coarsening technique is based on the underlying graph of the fine grid and does not involve geometry. The technique produces a set of node-nested coarse spaces which can be retriangulated based on fixed patterns in the agglomerated macroelement. This method offers great potential since the proposed interpolation operators are based on integers and leads to savings in storage and CPU time. Also, the algorithm recovers the natural structure of the coarse grids if the fine grid is regular. However, since the algorithm is purely topology-based, it does not distinguish between anisotropic and isotropic mesh regions which may lead to decreased convergence of the multigrid procedure. We propose a new and simple technique for defining coarse spaces which are properly nested in both the elemental and nodal sense. This method represents a hybrid between geometric and algebraic multigrid.

4.2 Multigrid Principles

4.2.1 Multigrid Requirements

In order to solve the linear system Eq. 53 using multigrid, the definition of the coarse spaces as well as the representation of the error components on these coarse spaces must be defined. Let $\{\Psi_k : \quad (\mathrm{k} = 0, \ldots, \mathrm{m})\}$ represent the hierarchy of finite dimensional coarse spaces along with the associated coarse grids. Also, let $\{\mathbf{A_k} : \quad (\mathrm{k} = 0, \ldots, \mathrm{m})\}$ be the approximations of $\mathbf{A}$ on these subspaces such that $\mathbf{A_0} = \mathbf{A}$. In order to represent the error in one space on the next coarse space, we require a transfer operator called the restriction

$$\mathbf{R}_k : \quad \Psi_k \rightarrow \Psi_{k+1} \tag{54}$$

which acts as a mapping between these spaces through a reduction in the space dimension. Also, error correction on the fine space from the coarse space requires the transfer operator called the prolongation

$$\mathbf{P}_k : \quad \Psi_k \rightarrow \Psi_{k-1} \tag{55}$$

which acts as a reverse mapping. To complete the picture, we require a smoothing operator $\mathbf{S}_k$ which acts to reduce the rough error components on each subspace Ψ_k. These smoothers may be different on each grid but are typically chosen to be the same e.g Gauß-Seidel . The generalized multigrid cycle now reduces to

1. Perform ν_1 pre-smoothing sweeps on the fine grid.

$$\phi_k^p = \phi_k^{p-1} + \mathbf{S}_k(b_k - \mathbf{A}_k\phi_k^{p-1}) \quad (\mathrm{p} = 1, \ldots, \nu_1)$$

2. Restrict the equation residual from the fine grid to the coarse grid.

$$\begin{aligned} \phi_{k,pre} &= \phi_k^{\nu_1} \\ b_{k+1} &= \mathbf{R}_k(b_k - \mathbf{A}_k\phi_{k,pre}) \end{aligned}$$

3. Solve on coarse grid and compute the coarse grid correction.

$$\phi_{k+1} = \mathbf{A}_{k+1}^{-1} b_{k+1}$$

4. Prolong the correction back to the fine grid from the coarse grid.

$$\phi_{\mathrm{k,corrected}} = \phi_{k,pre} + \mathbf{P}_k\phi_{k+1}$$

5. Perform ν_2 post-smoothing sweeps on the fine grid.

$$\begin{aligned}\phi_k^0 &= \phi_{k,corrected}\\ \phi_k^p &= \phi_k^{p-1} + \mathbf{S}_k(b_k - \mathbf{A}_k\phi_k^{p-1}) \quad (\mathrm{p} = 1,\ldots,\nu_2)\end{aligned}$$

Depending on the scheduling of operations between the spaces, we end up with different flavors of multigrid cycles such as the V-cycle, W-cycle and F-cycle ([40]). In the algebraic multigrid context, the coarse grid approximations $\mathbf{A}_k$ to $\mathbf{A}$ are defined using the formula

$$\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{A}_k\mathbf{P}_k \tag{56}$$

In our implementation of multigrid, the coarsening procedure terminates when the coarse grid operator $\mathbf{A}_k$ is small enough to solve exactly.

4.2.2 Galerkin Form for Transfer Operators

While the restriction and prolongation operators are independent, a Galerkin form of these operators requires that

$$\mathbf{R}_k = \mathbf{P}_k^T$$

Given the fact that a correction from the coarse space is required to remove the smooth error components from the fine space, it is natural to seek the best possible correction. Let ϕ_{k+1} represent the correction from the coarse grid and ϕ_k the current solution on the fine grid. The error in the solution after correction is thus

$$e_k = \phi_k + \mathbf{P}_k\phi_{k+1} - \mathbf{A}_k^{-1}b_k$$

Let us measure the error in the A-norm, $\|\cdot\|_{\mathbf{A}}$ and minimize the error:

$$\begin{aligned}\min_{\phi_{k+1}} \mathrm{F}(\phi_{k+1}) &= \min_{\phi_{k+1}} \|(\phi_k + \mathbf{P}_k\phi_{k+1}) - \mathbf{A}_k^{-1}b_k\|_{\mathbf{A}} \qquad (57)\\ &= \min_{\phi_{k+1}}(\phi_k + \mathbf{P}_k\phi_{k+1} - \mathbf{A}_k^{-1}b_k)^T\mathbf{A}_k(\phi_k + \mathbf{P}_k\phi_{k+1} - \mathbf{A}_k^{-1}b_k)\end{aligned}$$

Differentiation of the quadratic form with respect to ϕ_{k+1} gives

$$\mathbf{P}_k^T(\mathbf{A}_k + \mathbf{A}_k^T)(\phi_k + \mathbf{P}_k\phi_{k+1} - \mathbf{A}_k^{-1}b_k) = 0 \tag{58}$$

For a symmetric matrix $\mathbf{A}_k$, we may easily solve for ϕ_{k+1} and obtain

$$\phi_{k+1} = (\mathbf{P}_k^T\mathbf{A}_k\mathbf{P}_k)^{-1}\mathbf{P}_k^T(b_k - \mathbf{A}_k\phi_k) \tag{59}$$

Further examination of the Hessian shows that this is a minimum. Comparing this formula to the $\mathbf{A}_k$ in Eq. 56, we find that

$$\mathbf{R}_k = \mathbf{P}_k^T \tag{60}$$

where the restriction is now defined as the formal adjoint of the prolongation:

$$\mathbf{R}_k = \mathbf{P}_k^* \tag{61}$$

The same argument cannot be made for a non-symmetric matrix since the use of symmetry in the proof may not be used any longer. However, it can be seen that even in this case, Eq. 59 corresponds to a stationary point but no information may be gleaned from the Hessian. However, if we measure the error in the residual in the L2-norm and follow a similar proof, we again come to the same result as Eq. 60.

4.2.3 Fixed Point Iterative Methods

Let us consider a splitting of the matrix $\mathbf{A}$ in the following form:

$$\mathbf{A}_k = \mathbf{M}_k - \mathbf{N}_k \tag{62}$$

where $\mathbf{M}_k$ is non-singular. The basic idea behind preconditioning is to obtain a matrix $\mathbf{M}_k$ such that $\mathbf{M}_k \approx \mathbf{A}_k$ and inversion of $\mathbf{M}_k$ is much less expensive than $\mathbf{A}_k$. A basic iterative method is defined as the following linear fixed-point iteration:

$$\begin{aligned} \phi_k^{i+1} &= \mathbf{M}_k^{-1}\mathbf{N}_k\phi_k^i + \mathbf{M}_k^{-1}b_k \\ &= \phi_k^i + \mathbf{M}_k^{-1}(b_k - \mathbf{A}_k\phi_k^i) \end{aligned} \tag{63}$$

The matrix $\mathbf{M}_k$ is known as the preconditioning matrix and matrix $\mathbf{S}_k = \mathbf{M}_k^{-1}\mathbf{N}_k = \mathbf{I} - \mathbf{M}_k^{-1}\mathbf{A}_k$ is called the iteration matrix or smoother. Damping may also be taken into account by defining:

$$\phi_k^{i+\frac{1}{2}} = \mathbf{S}_k\phi_k^i + \mathbf{M}_k^{-1}b_k \tag{64}$$

$$\phi_k^{i+1} = \omega\phi_k^{i+\frac{1}{2}} + (1-\omega)\phi_k^i \tag{65}$$

$$= \mathbf{S}_k^*\phi_k^i + \omega\mathbf{M}_k^{-1}b_k \tag{66}$$

where

$$\mathbf{S}_k^* = \omega\mathbf{S}_k + (1-\omega)\mathbf{I} \tag{67}$$

For a given coarse space, let the exact solution be ϕ_k and the error in the solution be

$$e^i_{\phi_k} = \phi^i_k - \phi_k \tag{68}$$

This error is controlled by $\mathbf{S}_k$ in the following fashion:

$$\begin{aligned} e^{i+1}_{\phi_k} &= \phi^{i+1}_k - \phi_k \\ &= \mathbf{S}_k\phi^i_k + \mathbf{M}_k^{-1}b_k - \mathbf{I}\phi_k \\ &= \mathbf{S}_k\phi^i_k + \mathbf{M}_k^{-1}\mathbf{A}_k\phi_k - \mathbf{I}\phi_k \\ &= \mathbf{S}_k\phi^i_k - \mathbf{S}_k\phi_k \\ &= \mathbf{S}_k e^i_{\phi_k} \end{aligned} \tag{69}$$

The convergence property of the iterative method (63) can be summed up in the well known result:

Theorem 1 *Convergence of (63) for any initial guess u^0 is equivalent to*

$$\rho(\mathbf{S}^*_k) < 1 \tag{70}$$

4.2.4 Multigrid as a Fixed Point Method

Multigrid may also be thought of as a fixed point method and this can be shown fairly easily for the V-cycle multigrid cycle. We consider the general V(ν_1, ν_2) cycle for the two-level method but simplify it by assuming that we have only one pre-smoothing and one post-smoothing i.e a V(1,1) cycle. Let $\mathbf{A}$ represent the fine grid matrix and $\bar{\mathbf{A}}$ represent the coarse grid matrix. For an initial guess $\phi^{(0)} = 0$:

1. Pre-smoothing: $\phi^{(1)} = \mathbf{S}^T b$
2. Coarse grid correction:

   (a) Restrict residual:

   $$q^{(0)} = \mathbf{R}(\mathbf{I} - \mathbf{A}\mathbf{S}^T)b$$

   (b) Coarse grid solve:

   $$q^{(1)} = \bar{\mathbf{A}}^{-1}\mathbf{R}(\mathbf{I} - \mathbf{A}\mathbf{S}^T)b$$

(c) Fine grid correction:

$$\begin{aligned}\phi^{(2)} &= \phi^{(1)} + \mathbf{R}^T q^{(1)} \\ &= [\mathbf{S}^T + \mathbf{R}^T \bar{\mathbf{A}}_1^{-1} \mathbf{R}(\mathbf{I} - \mathbf{A}\mathbf{S}^T)]b\end{aligned}$$

3. Post-smoothing:

$$\begin{aligned}\mathbf{M}^{-1}b &= \phi^{(2)} + \mathbf{S}(b - \mathbf{A}\phi^{(2)}) \\ &= [\mathbf{S} + \mathbf{S}^T - \mathbf{S}\mathbf{A}\mathbf{S}^T + (\mathbf{I} - \mathbf{S}\mathbf{A})\mathbf{R}^T \bar{\mathbf{A}}^{-1} \mathbf{R}(\mathbf{I} - \mathbf{A}\mathbf{S}^T)]b\end{aligned}$$

The multigrid iteration matix $\mathbf{S}_{multigrid}$ now takes the form:

$$\begin{aligned}\mathbf{S}_{multigrid} &= \mathbf{I} - \mathbf{M}^{-1}\mathbf{A} \\ &= (\mathbf{I} - \mathbf{S}\mathbf{A})(\mathbf{I} - \mathbf{R}^T \bar{\mathbf{A}}^{-1} \mathbf{R}\mathbf{A})(\mathbf{I} - \mathbf{S}^T\mathbf{A})\end{aligned}$$

For the extension to multiple levels and variable number of pre- and post-smoothing sweeps, we refer to [31].

4.2.5 Convergence Conditions

In order to obtain mesh independent convergence properties, certain conditions must be met by the transfer operators. The definition of the subspace Ψ_k is obtained by interpolation in ψ_{k-1}, and according to the analysis in [31], these subspaces must satisfy stability and approximation properties to ensure convergence of Eq. 63. These properties are:

$$\|\mathbf{R}_k u\|_{1,\Omega} \leq C\|u\|_{1,\Omega}, \quad \text{(stability)} \tag{71}$$

$$\|\mathbf{R}_k u - u\|_{0,\Omega} \leq Ch\|u\|_{1,\Omega} \quad \text{(approximation)} \quad \forall u \in \mathrm{H}^1(\Omega) \tag{72}$$

and as noted, special care must be taken in defining $\mathbf{R}_k$. Another condition as outlined in [40] is

$$m_P + m_R > 2m \tag{73}$$

where the orders m_P, m_R of $\mathbf{P}_k$ and $\mathbf{R}_k$ are defined as the order (degree plus one) of the polynomials that are interpolated exactly by $\mathbf{P}_k$ and $\mathbf{R}_k$, and 2m is the order of the governing partial differential equation.

The use of nodal agglomeration in ACM to construct the subspaces results in the definition of the restriction as an injection operator. This has several associated problems. The first is that in 3D, this operator violates the stability property (71) ([31]). Secondly, both m_P and m_R are unity and for the Laplacian operator (2m = 2), the accuracy condition (73) is violated ([40]).

4.3 Agglomerated Coarse Space

In this section, we now describe the construction of the coarse spaces as well as coarse space basis functions and multigrid operators. Our goals in the agglomeration and associated construction of the coarse space basis functions are:

1. The coarse grid matrix $\mathbf{A}_k$ defined on the coarse grid should be a good approximation to the fine grid matrix $\mathbf{A}_0$.
2. The coarse grid should have a reduction of anisotropic effects from previous coarse spaces.

The proposed algorithm is based on the fusion of coarse space elements into macroelements with subsequent definitions of the coarse grid topology and basis functions. This method is applied recursively to generate the hierarchy of coarse spaces. One essential difference between this method and that proposed by Chan *et al* is that the coarse mesh elements are not converted into standard elements by a retriangulation but are generalized polygons formed by the agglomerated fine mesh elements. This is especially attractive in 3D because of the complicated rules which may be involved for the retriangulation method described in [34]. Although the support for the basis functions defined on these macroelements is larger than standard triangular elements, a well designed agglomeration should relieve some of this. This algorithm also has the feature that the resulting coarse grid topology is both node-nested and element-nested.

4.3.1 Coarse Space Topology

The coarse grid topology is constructed by partitioning of the elements into macroelement groups as shown in Fig. 19. A macroedge is defined to be the ordered collection of fine grid edges which are shared by two neighboring macroelements. To complete the definition of the coarse grid graph, the coarse nodes are chosen to be the fine grid nodes where three or more macroedges meet. This is the reverse of what is described by Chan *et al* where the coarse grid points are first defined and then the macroelements are chosen.

Two different element partitioning algorithms have been developed and both are similar in the sense that they are based on elemental accretion across edges using some measure of the coupling strength of the vertices making up that edge. They however differ in that one tries to alleviate grid anisotropy while the other typically introduces grid anisotropy in regions of stiff matrix coefficients. The second algorithm is based on a semi-coarsening algorithm using the subspace matrix as the

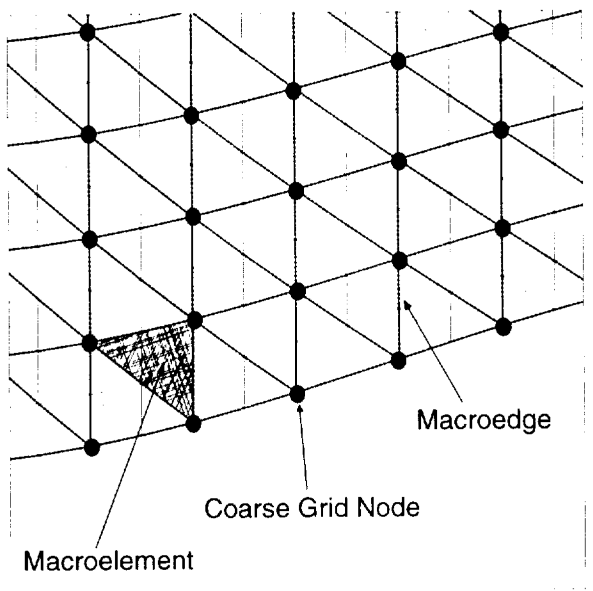


Figure 19: Coarse Space Topology

control parameter while the first is a reduced-geometry based algorithm which attempts to alleviate stiffness in regions of highly stretched grids.

Matrix Based Agglomeration

The matrix based algorithm has its origins in the pure AMG implementation of the ACM algorithm. However, modifications are required in order to apply this to element agglomeration as opposed to vertex agglomeration which it is was originally designed for ([37]. The basic idea is that strong matrix coupling between vertices in the graph of the mesh typically corresponds to a weak coupling between the vertices in the dual graph. The strength of the coupling is typically based on the stencil coefficients of the matrix. We will describe later, the implemented definition of the vertex coupling strength in relation to the development of the implicit line solver. Since the dual graph vertices correspond to the elements, agglomeration of elements across edges with weakly coupled vertices is equivalent to directional agglomeration of strongly coupled vertices.

Using this principle, we may now develop an algorithm which directionally clusters elements in order to relieve stiffness in the next coarse space. The accretion is performed with a Breadth First Search (BFS)/ Greedy algorithm which maintains a queue of elements sorted according to the relative coupling strengths of the vertices on their bounding edges. We now present the algorithm in detail:

Algorithm 1 *(Matrix Based Macroelement Construction)*

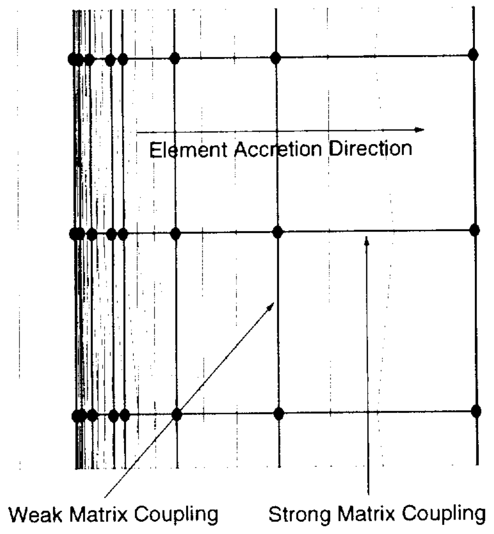


Figure 20: Matrix Based Agglomeration in Boundary Layer

Step 0: *Consider the graph of the mesh: $G = (V,E)$ and calculate the coupling strengths for the edges E in the graph.*

Step 1: *Seed element procurement. Obtain a seed element to initialize the BFS algorithm. If there is no seed element in the queue, choose any suitable element which does not belong to a macroelement group.*

Step 2: *Perform accretion around the seed element by recursively considering the neighbouring elements.*

Step 3: *Repeat* **Step 2** *until the macroelement has desired number of elements or heap contains no more elements.*

Step 4: *If the number of elements agglomerated is less than a specified fraction of the desired number of elements (usually $\frac{1}{2}$), these agglomerated elements are ungrouped and the original seed element is marked.*

Step 5: *Repeat* **Step 1** *until all elements either belong a macroelement or has been marked as an invalid seed.*

In **Step 2** above, a heap is maintained whose members contains a key pair consisting of element and face identifiers. Initially, seed element is placed on the heap. The head of the seed is then popped for the current seed element and added to the macroelement list. For every neighbor of the popped element which does not belong to a macroelement, the connection strength of the separating edge is checked. If the edge is considered to be weak, the neighboring element is inserted into the macroelement. For every newly inserted element, the neighbors which do belong to any macroelement are considered and the corresponding element/edge key identifier pair is inserted into the the heap according to the edge connection strength.

After the algorithm terminates, post-processing is necessary to deal with exceptions which may arise. These are described below:

(1) Elements not in any macroelement: Elements which are marked as invalid seeds and have not been absorbed into a macroelement will end up as lone elements. These elements are merged with the neighboring element/macroelement that shares the edge with the weakest connectivity.

(2) Insufficient number of elements: After exception 1 above, a macroelement may end up with an insufficient number of agglomerated elements. The macroelement is divided up amongst neighboring macroelements by erosion of the boundary elements until there are no more elements. The decision as to where an element goes is also based on the edge connectivity.

(3) Insufficient number of coarse nodes: A macroelement may also end up with two or less coarse grid nodes which presents a problem in the construction of the transfer operators. These macroelements are also divided up amongst neighboring macroelements.

This algorithm is closely tied to the line solver and can be particularly effective for equations with strongly preferential directions. Extension to 3D would be straightforward if a suitable "face" connectivity can be defined. If the connection strength for all edges is defined to be a constant, then an isotropic agglomeration algorithm is recovered. Unfortunately, since there will be no preferential direction, there is no real control in the regularity of the coarse grid and self similar meshes cannot be obtained for structured, topologically rectangular meshes.

Geometry Based Agglomeration

The geometry based algorithm is based on the idea of removing grid anisotropy as well as maintaining isotropy in the isotropic regions of the mesh. This is related to the work done by Mavriplis [33, 41] except that it is applied to elements as opposed to vertices. The proposed algorithm makes use of the edge lengths only and this represents a reduced geometry method. The reduced geometry for the lower level coarse spaces is defined entirely in terms of the fine grid i.e the macroedge lengths are simply the sum of the edge lengths of the constituting fine grid edges. The decision to agglomerate two neighboring elements is based on a geometry based connectivity concept which we term *macroelement skew.*

Definition 1 *For a general polygon, the polygon skew is a measure of anisotropy and is defined as the area of the n-gon divided by the area of a perfect n-gon with the same perimeter.*

In the extreme cases, this is zero for colinear polygon vertices and unity for a perfect n-gon. It can be seen that this readily extends to 3D. The macroelemental areas for the coarse spaces are also easy to compute as they are simply sums of the agglomerated element areas. This makes it easy to apply the algorithm recursively once the required geometry variables have been computed on the finest mesh. In order to complete the operators required for this algorithm, we need to define the edge connection strength which we term *edge skew.*

Definition 2 *For an element which borders a macroelement/element on a given edge, edge skew is defined as the macroelement skew of the macroelement which would be created if the element is merged with the macroelement/element across that edge.*

We now present the algorithm in detail:

Algorithm 2 *(Geometry Based Macroelement Construction)*

Step 0: *Consider the graph of the mesh: $G = (V,E)$ and calculate the edge length for the edges E in the graph.*

Step 1: *Initialize seed queue.*

Step 2: *Seed element procurement. Obtain a seed element to initialize the algorithm. If there is no seed element in the queue, choose any suitable element which does not belong to a macroelement group.*

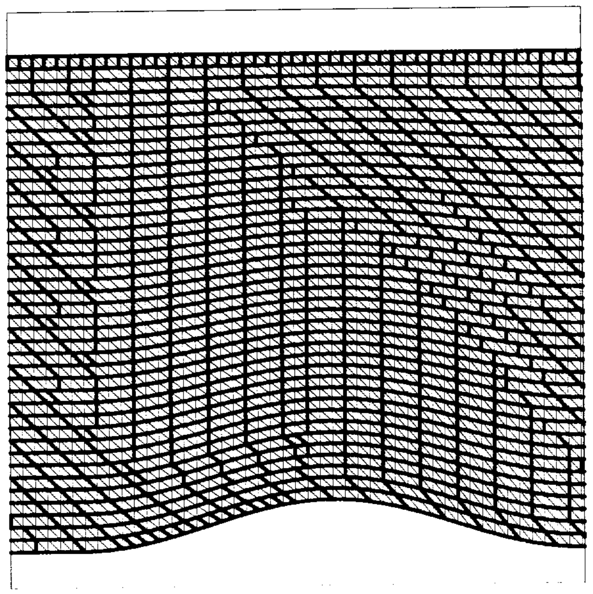
Step 3: *Perform accretion around the seed element. Place seed element in macroelement and for every neighboring element, compute the edge skew. Every neighboring element which has an edge skew larger than some specified fraction (typically 0.75) of the average edge skew and not in a macroelement is placed in the macroelement.*

Step 4: *Enqueue seed elements. New seed elements are placed in the queue to continue the algorithm. These are chosen to be elements which share a vertex but no edges with the macroelement. In 3D, this would extend to elements which share a vertex and/or an edge but no faces with the macroelement.*

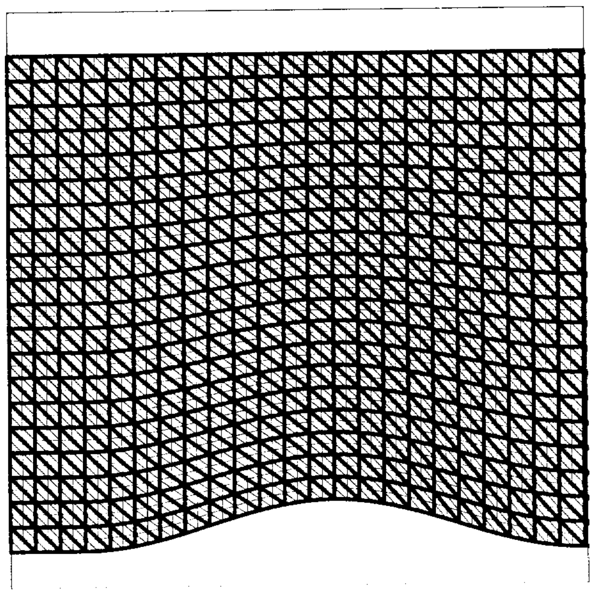
Step 5: *Repeat* **Step 1** *until all elements either belong a macroelement or there are no more seed elements.*

A queue of seed elements needs to be maintained and hence in **Step 1** above, this is initialized with one element. This initial choice can be very important especially in cases of unstructured meshes generated from structured data. In such a case, the agglomeration pattern may radically depart from a 4:1 agglomeration in 2D since the accretion algorithm will not properly identify potential elements. In this case, simply pick an element with no domain boundary edges.

After the algorithm terminates, post-processing is necessary to deal with "sliver" elements which may not have been picked up by the algorithm. A determination of which macroelement to merge these elements with is made a-priori based on edge skew. In the case where the lengths and areas are equal, the algorithm degenerates to a 4:1 isotropic agglomeration in 2D and fully recovers the natural coarse structure for a regular grid.



(a) Matrix Based Agglomeration



(b) Geometry Based Agglomeration

Figure 21: Resultant Agglomerations Based on Different Agglomeration Algorithms

Fig. 21 shows the difference between the two algorithms in the case of a cosine Ni Bump with 9600 elements and 4961 vertices. The underlying geometry is generated from structured data and the geometry based algorithm nicely recovers the natural coarse grid.

4.3.2 Coarse Space Basis Functions

The transfer operators may now be defined based on the constructed macroelements. The GCA formulation is in effect so it is sufficient to construct the prolongation operator only. We would like the basis functions to at least satisfy the stability (71) and approximation (72) conditions, preserve the constant function, and behave like standard interpolants i.e

$$\Phi_i(\mathbf{x}_j) = \begin{cases} 1 & \text{if i = j} \\ 0 & \text{if i} \neq \text{j} \end{cases} \tag{74}$$

The construction of the proposed basis functions makes use of topology and reduced geometry if provided. If the geometry is not given, then the elements are assumed to be isotropic. We now define the basis functions using graph distance interpolation on both the boundary and interior. If the geometry is available, this is used in combination to form a more accurate interpolant. This is an improvement over the interpolation proposed by Chan *et al* which makes use of graph distance interpolation on the boundary and constant interpolation over the interior. This algorithm leads to a quasi-linear interpolant as shown in Fig. 22. The detailed algorithm is given below:

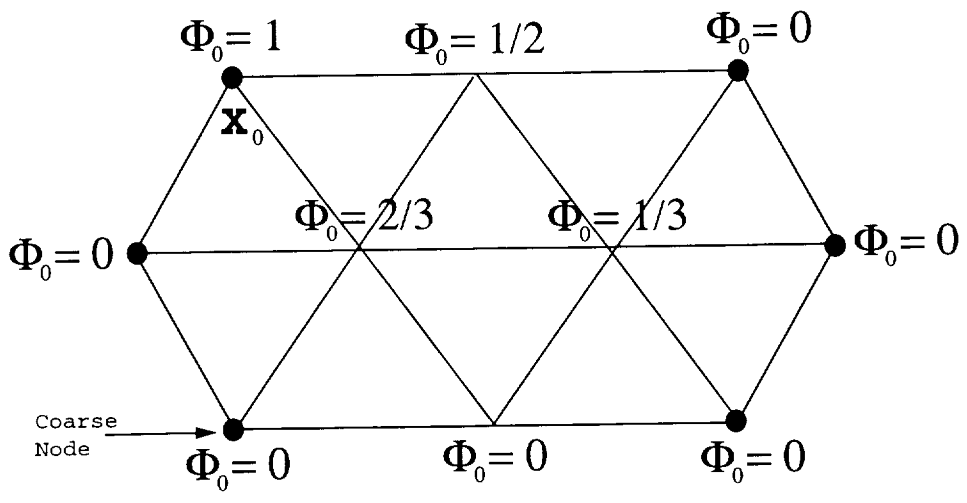


Figure 22: Coarse Space Basis Function Based on Graph Distance

Algorithm 3 *(Basis Function Construction)*

Step 1: *For each macroelement, create a local subgraph. In the process, create an ordering of the boundary edges such that the boundary can be traversed.*

Step 2: *Extract the list of interior vertices. Extract the ordered list of coarse grid vertices by traversing the boundary edges.*

Step 3: *For all fine grid edge vertices which lie between consecutive coarse grid nodes, construct length weighted interpolation data. The macroedge length is also computed simultaneously.*

Step 4: *Interior vertex interpolation. For each coarse grid node in the macroelement, a BFS iteration on the local subgraph is performed with the coarse grid node as a seed. Both the level set as well the distance from the coarse node is recorded for all interior nodes in the subgraph during the process. The graph distance of each macroelement fine grid node from the macroelement coarse grid nodes is then set. For each fine grid node, these distances are then weighted to sum to unity.*

4.3.3 Scaling Issues

The success of the multigrid methods depends heavily on how good of an approximation the coarse space matrices $\mathbf{A}_k$ are to $\mathbf{A}$. In the GCA formulation, special care must be taken to ensure that that these approximations are accurate enough. The construction of the prolongation operator is typically not a problem. However, the definition of the restriction operator needs to be modified slightly. Let us choose the restriction operator to be

$$\mathbf{R}_k = \sigma \mathbf{P}_k^* \tag{75}$$

where $\mathbf{P}_k^*$ is the formal adjoint of the prolongation and σ is a suitable scaling factor. The scaling of $\mathbf{R}_k$ is determined by the role of $\mathbf{R}_k$. If $\mathbf{R}_k$ is to be used to construct coarse grid representations of ϕ_k (i.e $\mathbf{R}_k\phi_k$), then $\sum_j \mathbf{R}_k(i,j) = 1$. However, if R is to be used to transfer residuals to the coarse grid, then the correct value of the scaling depends on the scaling of the fine grid and coarse grid problems. This implies that the coarse grid problem should be consistent with the differential problem in the same way as the fine grid problem. This is the basic problem with vertex agglomeration. Let H represent a characteristic mesh size on the coarse grid and h represent a characteristic length on the fine grid. Finite volume and finite element schemes in 2D lead to a scaling rule which says that $\sum_j \mathbf{R}_k(i,j) = (\frac{H}{h})^2$ which can be viewed as a ratio of the area associated with a coarse grid node to the area associated with the counterpart fine grid node.

The basis function construction algorithm thus needs to be modified to take this into account. The associated area for the nodes on both the fine and coarse grids is computed by looping through the elements on a grid and sending elemental area contributions (element area divided by the number of element vertices) to these vertices. The diagonal scaling matrix σ is now computed. However, we would like to maintain the GCA formulation, so the system is symmetrized by defining

$$\hat{\mathbf{P}} = \mathbf{P}\sigma^{\frac{1}{2}} \tag{76}$$

$$\hat{\mathbf{R}} = \sigma^{\frac{1}{2}}\mathbf{P}^T \tag{77}$$

$$= \hat{\mathbf{P}}^T \tag{78}$$

This formulation has the nice feature that the eigenstructure is preserved for an SPD matrix. The coarse grid equations are now constructed using the GCA approximation and the multigrid operation continues with this new definition for the transfer operators.

4.4 Implicit Line Smoother

In the context of our multigrid formulation, we would like to be able to solve the high Reynolds number Navier-Stokes equations which is a parabolic system characterized by both advective and acoustic modes. We would like to decouple the two modes in the following way. Multigrid methods are very effective at damping out elliptic error modes such that the choice for the smoother must be that it can handle the advective error components. Following this reasoning, we have opted to use an implicit Gauß-Seidel line relaxation scheme where the lines are constructed to follow characteristic directions. The use of a line relaxation scheme leads to a natural splitting of the matrix into tridiagonal submatrices which may be solved by any of the myriad tridiagonal matrix solvers. The general rule behind the line relaxation is that points which are strongly coupled should be updated simultaneously. This leads to the description of how the implicit lines are constructed.

4.5 Implicit Line Construction

The implicit line construction process is based on the philosophy of linking strongly coupled nodes. In order to reduce the amount of work in the line smoother, minimal overlap between the lines is allowed. To properly describe the algorithm, we need to define two terms:

1. **Coupling measure:** The *coupling measure* between two nodes gives a local quantification of the connectivity between these nodes. Typically, this is based on the matrix stencil connecting these points. Ideally, the measure of the coupling between the nodes should be based

on a discretization of a scalar convection-diffusion equation but other measures such as an approximation to the flow velocity or streamlines may be used. This becomes even more complicated in the case of block systems of equations such as arise in discretizations of the Navier-Stokes equations. It may however be possible to use entropy variables to form an approximation of the scalar convection equation.

2. **Coupling degree:** The *coupling degree* between two nodes gives a quantification of the connectivity between these nodes as compared to other connected nodes. This is based on the coupling measure earlier defined. In the current framework, the degree of coupling between two nodes V1 and V2 is determined by first computing the coupling measure of the nodes to all surrounding nodes and this measure may be freely defined. For each node, the maximum value is taken to be the reference value for that node. From the point of view of V1, the coupling between nodes V1 and V2 will be considered strong if the connectivity measure between these nodes is larger than a threshold value. This threshold value is defined to be a fraction of the V1 reference value. Two nodes are linked up in the line if the degree of coupling between them is stonger than any other connection. In advection dominated flows, strong coupling tends to be one-sided which is why a two way consideration (i.e from both points of view of V1 and V2) is necessary.

Line construction is done in a two pass process. The first pass involves the construction of individual lines while the second pass involves merging lines to reduce the line count and improve convergence. The construction of a single line begins by choosing a seed node. All nodes which do not exist in a line are placed in a queue in no particular order and the seed node is chosen from this queue. The chosen seed is usually not an extremety of the line based on a straightforward implementation of the algorithm. Hence, we need to introduce the concept of forward and backward mode line construction.

4.5.1 Forward Mode Line Construction

Forward mode line construction involves stepping through a line by starting at a given node and simply choosing the next node with the best strong connection which is not in the current line. The best connection, however, may be a node which already exists in another line. Hence, in an effort to minimize overlap, the next best node which has a strong connection and has not been assigned to a line is chosen. If no such node exists, then the originally selected node is chosen.

If the number of times a node which has already been assigned to a line is chosen reaches some predetermined limit, then the line is terminated and the overlap nodes are removed. Also, if the next chosen node is an extremety of another existing line, then the current line is simply merged with that line. Flows with recirculation regions or circular flows give rise to cyclic couplings, so this case is specially handled if the chosen node turns out to be the head of the current line.

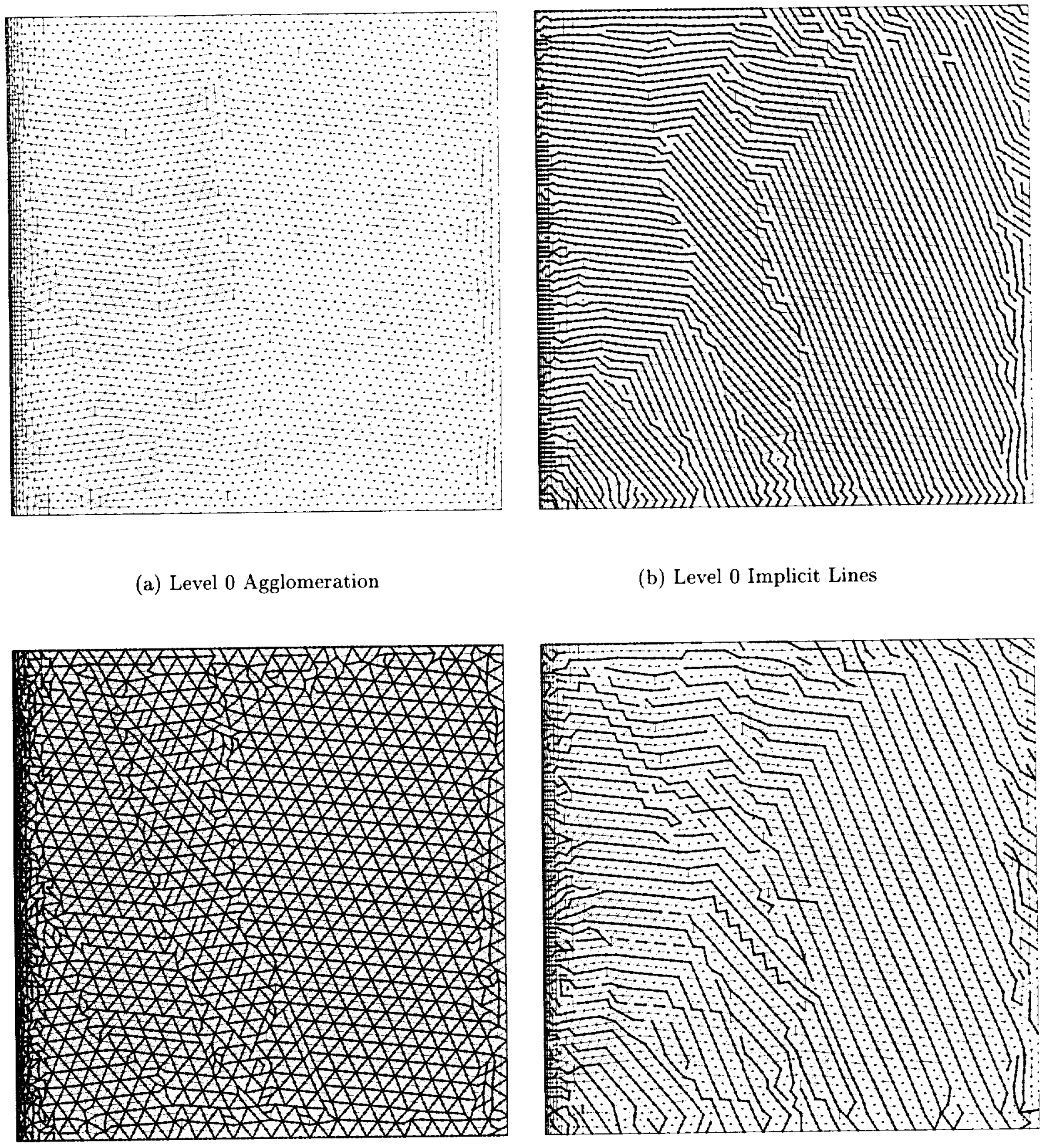
4.5.2 Backward Mode Line Construction

Backward mode line construction is akin to the reverse of the forward mode. For a given node, all the adjoining nodes are scanned and a single step of the forward mode algorithm is performed on these connected nodes. If any of these connected nodes would have chosen the starting node as the next node in the line, then this node is chosen as the next node. In the event that multiple nodes would have chosen this node, then the one with the strongest coupling is chosen. As in the forward mode, if the chosen node is an extremety of an existing line, then the current line is merged with that line.

4.5.3 Line Processing

The line is constructed by performing backward and forward mode construction from the seed point. After the two halves of the line have been constructed, they are merged together into a single line. The post-processing pass is performed once all the nodes have been assigned to reduce the line count. This is done by checking the extremeties of every line and testing to see if the node on the extremety has a strong connection to a node on the extremety of another line. The connection threshold for each node is reduced by a factor (typically between 0.5 and 0.75) to allow more lines to be considered.

Fig. 23 shows a 2-level example of the implicit line construction on the grids. The agglomeration shown is the geometry based algorithm.



(a) Level 0 Agglomeration

(b) Level 0 Implicit Lines

(c) Level 1 Agglomeration

(d) Level 1 Implicit Lines

Figure 23: Multilevel Agglomeration and Implicit Lines

4.6 Results

The performance of the algorithm for different flow regimes and characteristics is now presented. As mentioned by Chan ([31]), the performance of many numerical techniques for elliptic problems are not robust for convection dominated flows. Two particular kinds of flows which are known to cause divergence for many techniques are boundary layer flows and recirculation flows.

4.6.1 Boundary Layer Flow

We consider the linear convection diffusion equation (Eq. 45) over a square domain $\Omega =]0,1[^2$ and prescribed velocity field $\mathbf{U}$ = (-y,x). The forcing function f is set to zero and the dirichlet boundary on the inflow and left wall (x = 0) is

$$\alpha_D = \begin{cases} 5.0(x-0.2), & \text{for } 0.2 < x \leq 0.4, \quad y = 0 \\ 1, & \text{for } 0.4 < x \leq 0.6, \quad y = 0 \\ 1-5(x-0.6), & \text{for } 0.6 < x \leq 0.8, \quad y = 0 \\ 0, & \text{otherwise} \quad . \end{cases}$$

This particular set of conditions is chosen to simulate a boundary layer flow with the nominal Reynolds number

$$\begin{aligned} Re &= \frac{U_h \cdot l_h}{\nu} \\ &= \frac{1}{\nu} \end{aligned} \tag{79}$$

The discretized domain is adapted on the dirichlet boundary to capture the boundary layer as shown in Fig. 23(a). All presented results are based on a V(1,1) multigrid cycle with no FMG. The agglomeration technique is the geometry based algorithm and the solver is terminated when the RMS absolute error in the residual is less than 10^{-13}. The relaxation factor ω chosen for all the test cases was 0.95. The behaviour of the algorithms for a variety of parameter states is now presented:

Multigrid Level Dependency:

The dependence of the convergence rate on the number of coarse spaces is shown in Fig. 24. The fine mesh has 60399 vertices and 119714 elements and a total of 6 coarse spaces were constructed. In the asymptotic limit, the convergence rate is the same for all the curves and beyond the two-grid case, the curves fall unto the same line.

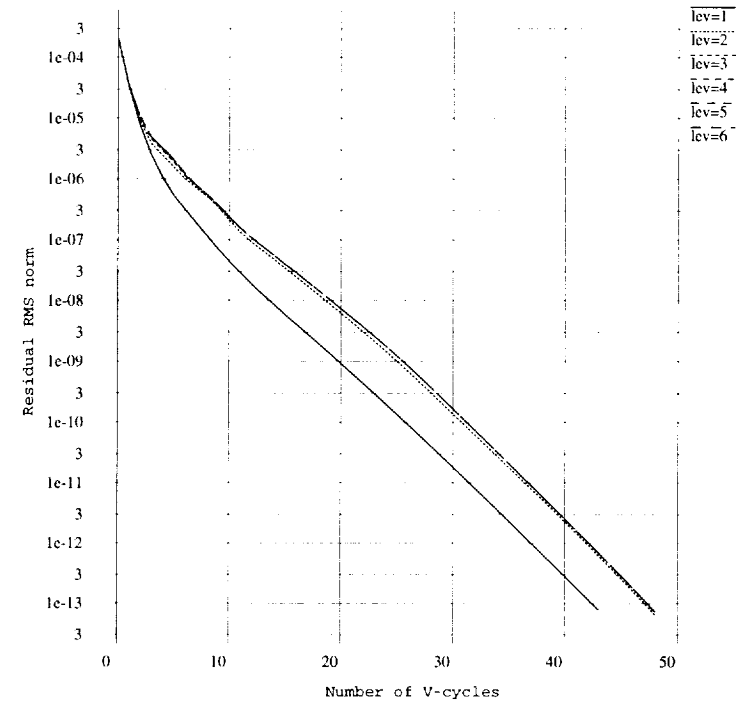


Figure 24: Multigrid Level Dependency for Boundary Layer Problem: 60399 points; Re = 1.0e6

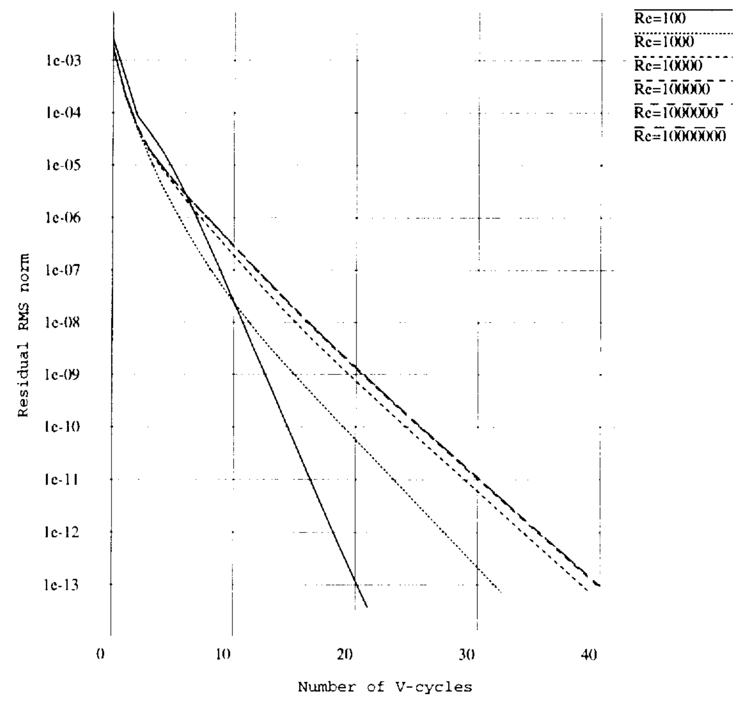


Figure 25: Reynolds Number Dependency for Boundary Layer Problem: # of points = 3849

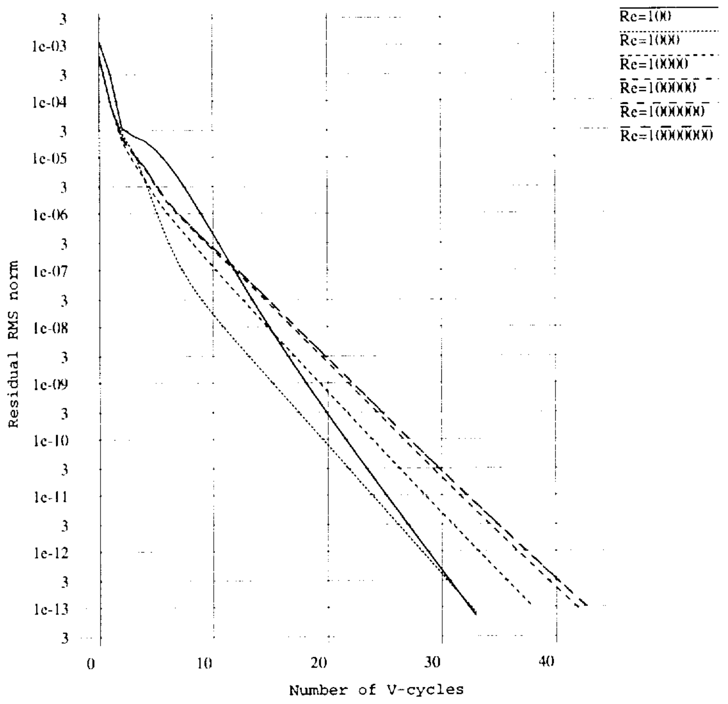


Figure 26: Reynolds Number Dependency for Boundary Layer Problem: # of points = 15763

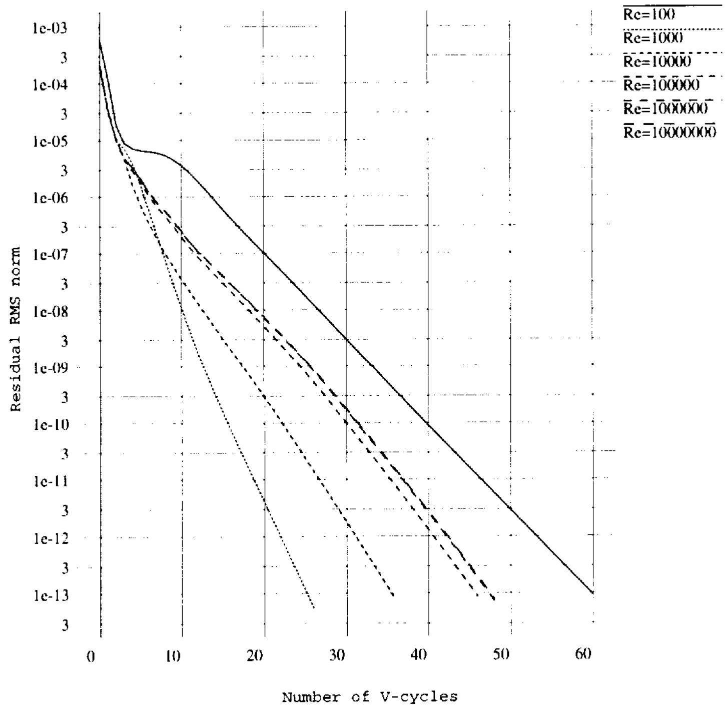
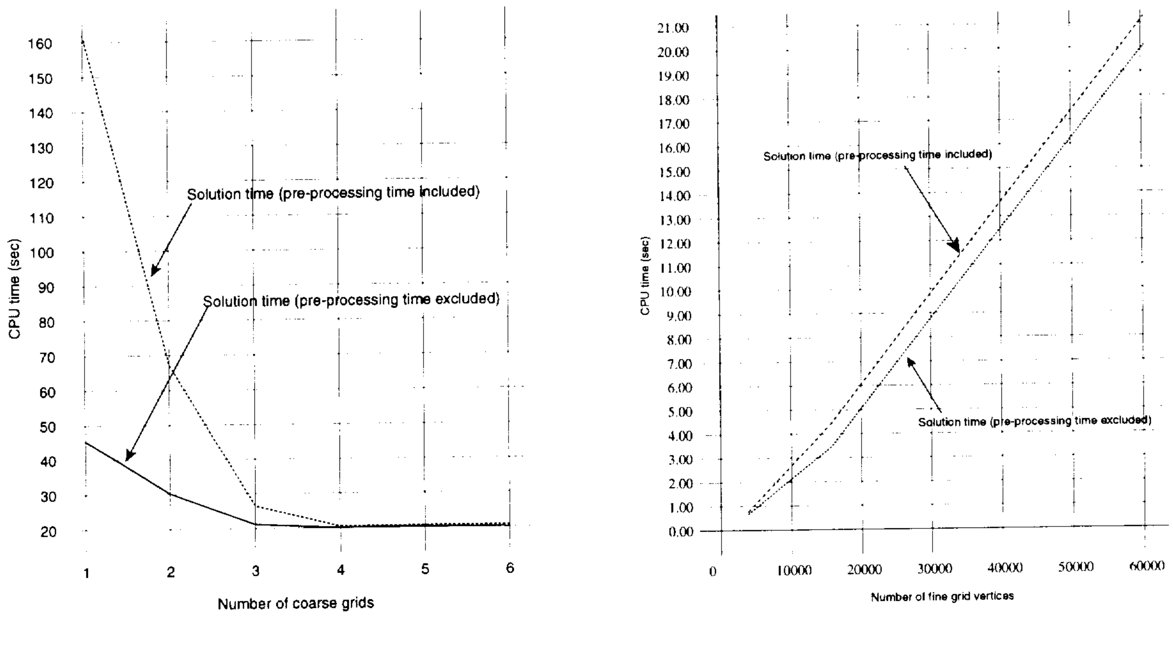


Figure 27: Reynolds Number Dependency for Boundary Layer Problem: # of points = 60399

Reynold Number Dependency:

The dependence of the convergence rate on the Reynolds number is shown in Fig. 25, Fig. 26 and Fig. 27 for a range of Reynolds numbers from 10^2 to 10^7. Figures 25 to 27 were generated on a sequence of fine meshes with 3849, 15763 and 60399 vertices respectively which represents an approximate halving of the mesh spacing. In all cases, we find a similar asymptotic convergence rate. Even more important is the fact that the algorithm works well for such a wide range of Reynolds numbers while maintaining a fairly constant bound on the number of iterations required for convergence.

A noticeable trend can be observed with the $Re = 100$ case, which is the increasingly pronounced stall in the residual after a few iterations followed by convergence. The problem is fairly elliptic such that well defined characteristic lines are not easily identifiable. As a result, the line solver does not facilitate proper propagation of information in the characteristic directions. For such a low Re problem, the use of a point implicit Gauß-Seidel smoother would be a better choice.



(a) # of points = 60399 (b) # of coarse grids = 4

Figure 28: Timing Information: Re=1.0e6

Fig. 28(a) shows timing information for the algorithm on the boundary layer problem. The Reynolds number is 1 million and the fine mesh has 60399 points. The plot shows the CPU time required for the solution versus the number of multigrid levels with and without the preprocessing time included. This preprocessing time includes the time required to construct the coarse spaces and implicit lines as well as factorization of the coarsest grid matrix. Fig. 28(b) shows the CPU time required for the solution versus the number of fine grid vertices for the three meshes mentioned above with the number of coarse meshes fixed at 4.

4.6.2 Recirculation Flow

We also consider the linear convection diffusion equation (Eq. 45) over a square domain $\Omega =]-1,1[^2$ (Fig. 29) and prescribed velocity field $\mathbf{U}$ = (-y,x). Neumann boundary conditions $\frac{\partial \phi}{\partial \eta} = 0$ are imposed on the domain boundaries and the flow field is initialized with random values ranging from -1 to 1 with a mean of zero. The nominal Reynolds number is also defined as

$$\begin{aligned} Re &= \frac{U_h \cdot l_h}{\nu} \\ &= \frac{1}{\nu} \end{aligned}$$

The discretized fine mesh has 1990 points and an example of a solution converged to an RMS residual tolerance of 10^{-13} is shown in Fig. 30 for a Reynolds number of 1 million. The range of ϕ shown is between -1.17×10^{-7} and 7.94×10^{-9}.

The results in Fig. 31 show the dependence of the convergence rate for a number of Reynolds numbers. There is significant deviation of the convergence rate as the Reynolds number increases. It can be seen that for the higher Reynolds number cases, the algorithm has increasing trouble in damping out certain modes. This is due to the nature of the implicit lines in the context of an unstructured grid. Due to the unstructured nature of the grid, these lines do not wrap around on themselves (Fig. 32) and as such, the system of equations for the lines are not periodic. This means that information cannot be propagated properly along the characteristic lines with resulting deterioration in convergence rate. However, there is a reduction of six orders of magnitude in the residual before this effect becomes noticeable, so it is still possible to use the algorithm in the role of a preconditioner to Krylov subspace solvers.

References

[1] J. Wong, D. Darmofal and J. Peraire, *The solution of the compressible euler equations at low Mach numbers using a stabilized finite element algorithm*, to appear in Comp. Meth. in Appl.

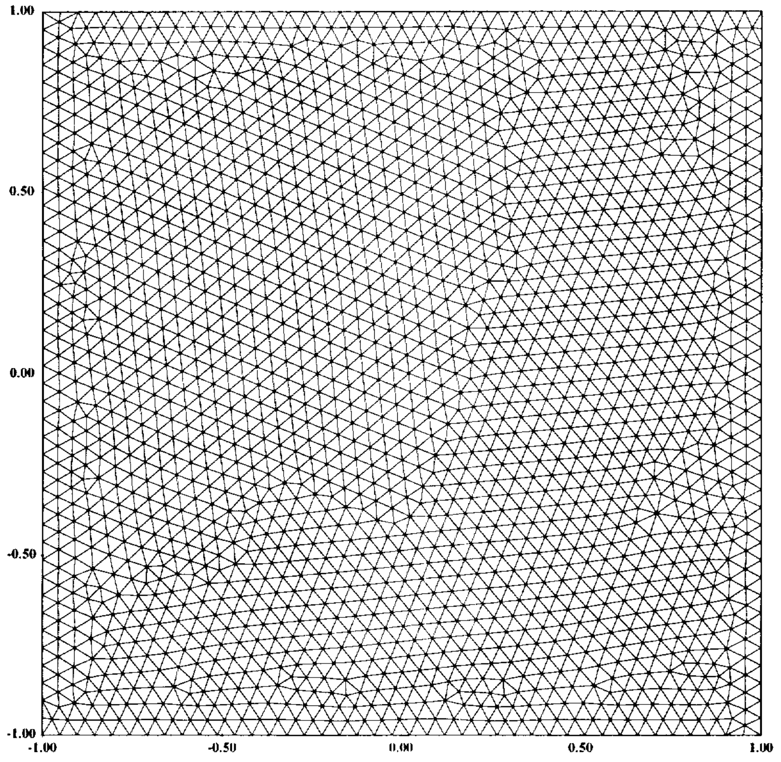


Figure 29: Circular Convection Grid: # of points = 1990

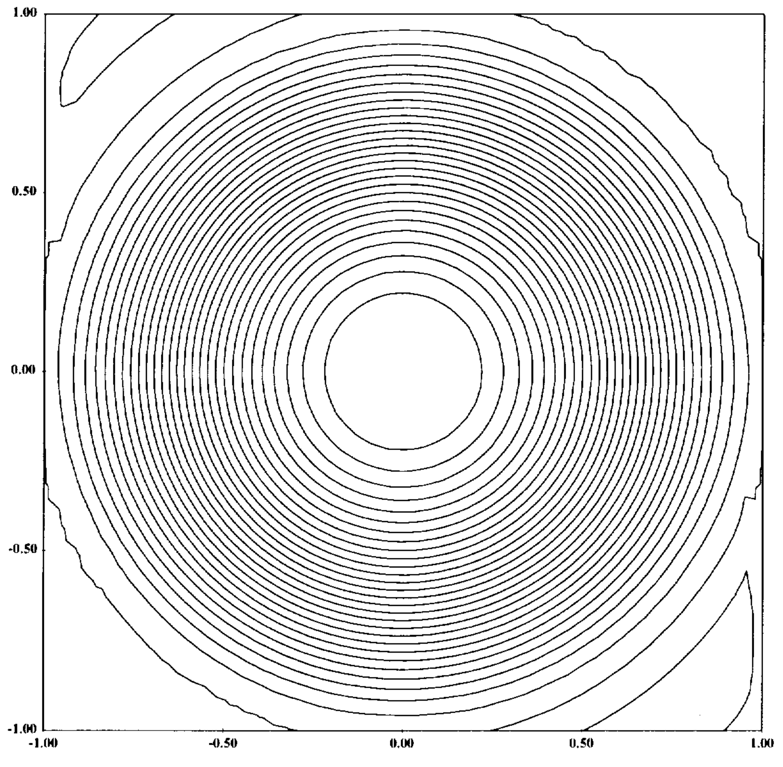


Figure 30: Circular Convection Solution Contours: # of points = 1990, Re=1.0e6

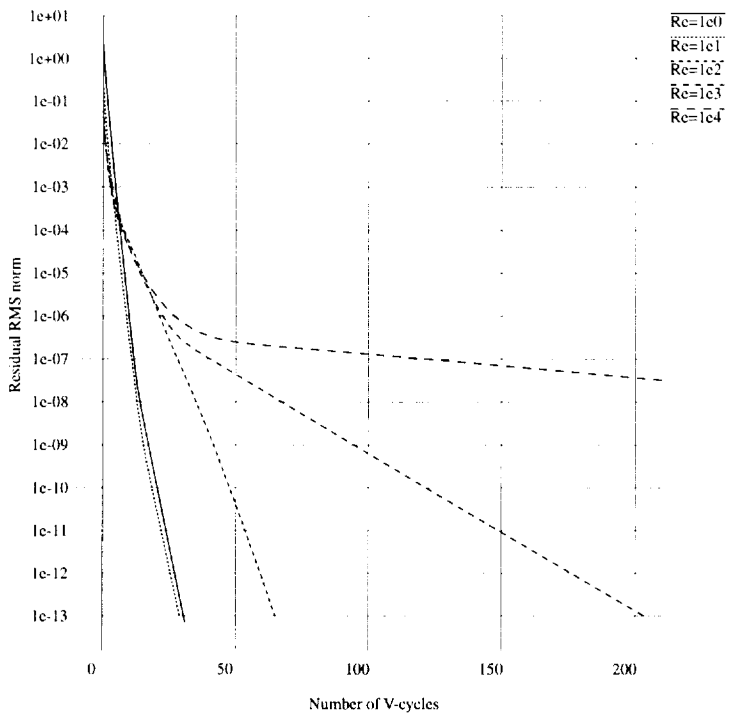


Figure 31: Reynolds Number Dependency for Circular Convection Problem

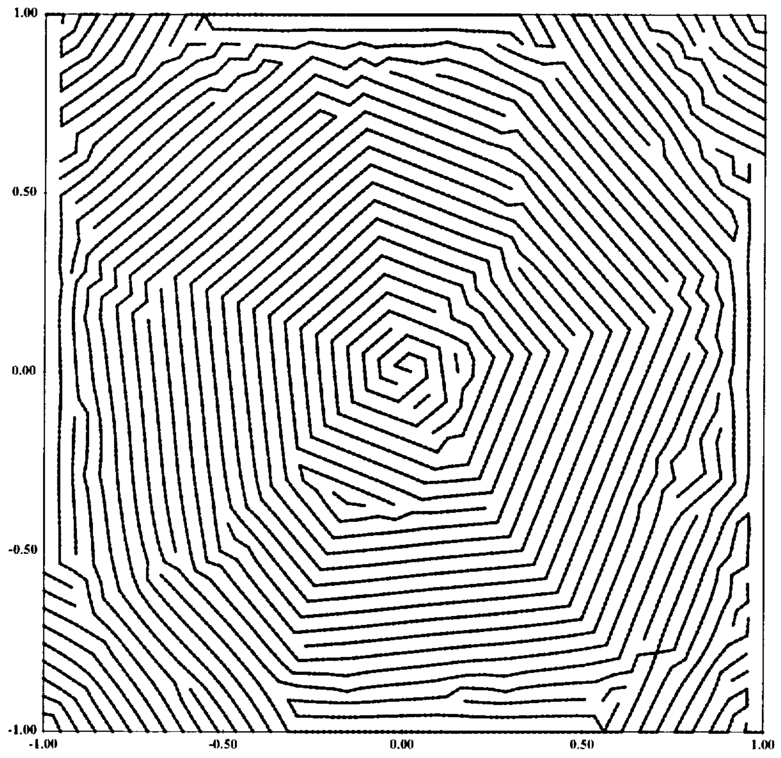


Figure 32: Fine mesh implicit lines

Mech. and Engngr., Dec. 2000.

[2] T.J. Barth, *Numerical methods for gasdynamic systems on unstructured meshes*, Lecture Notes in Computational Science and Engineering, 1998, pp. 195-284

[3] D. Choi and C.L. Merkle, *Application of time-iterative schemes to incompressible flow*, AIAA Journal, 23, 1985, pp. 1518-1524.

[4] Y.H. Choi and C.L. Merkle, *The application of preconditioning in viscous flows*, Journal of Computational Physics, 105, 1993, pp. 203-223.

[5] D.L. Darmofal and P.J. Schmid, *The importance of eigenvectors for local preconditioners of the Euler equations*, Journal of Computational Physics, 127, 1996, pp. 346-362.

[6] D.L. Darmofal and B. Van Leer, *Local preconditioning: Manipulating Mother Nature to fool Father Time*, Computing the Future II: Advances and Prospects for Computational Aerodynamics, M. Hafez and D.A. Caughey, John Wiley and Sons, 1998, pp. 211-239.

[7] D.R. Fokkema, *Subspace methods for linear, nonlinear and eigenproblems*, PhD Thesis, Utrech University, 1996.

[8] H. Guillard and C. Viozat, *On the behavior of upwind schemes in the low Mach number limit*, Computers and Fluids, 28, 1999, pp. 63-86.

[9] J. Guerra and B. Gustafsson, *A numerical method for incompressible and compressible flow problems with smooth solutions*, J. Comp. Phys., 63, 1986, pp. 377-397.

[10] B. Gustaffson, *Unsymmetric hyperbolic systems and the Euler equations at low mach numbers*, Journal of Scientific Computing, Vol. 2, No. 2, 1987.

[11] B. Gustafsson and H. Stoor, *Navier-Stokes equations for almost incompressible flow*, SIAM Journal of Numerical Analysis, 28, 1991, pp. 1523-1547.

[12] A. Harten, *On the symmetric for of systems of conservation laws with entropy*, Journal of Computational Physics, 49, 1983, pp. 151-164.

[13] T.J.R. Hughes, L.P.Franca and M. Mallet, *A new finite element formulation for computational fluid dynamics : I. Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics*, Comp. Meth. in Appl. Mech. and Engnr., 54, 1986, pp. 223-234.

[14] T.J.R. Hughes and M. Mallet, *A new finite element formulation for computational fluid dynamics : III. The generalized streamline operator for multidimensional advective diffusive systems*, Comp. Meth. in Appl. Mech. and Engnr., 58, 1986, pp. 305-328.

[15] T.J.R. Hughes, L.P. Franca and G.M. Hulbert, *A new finite element formulation for computational fluid dynamics : VIII. The Galerkin Least-Squares method for advective diffusive equations*, Comp. Meth. in Appl. Mech. and Engnr., 73, 1989, pp. 173-189.

[16] S. Klainerman and A. Majda, *Compressible and incompressible fluids*, Comm. Pure Appl. Math., 34, 1981, pp.629-651.

[17] L. Machiels, A.T. Patera, J. Peraire, and Y. Maday, *A general framework for finite element a posteriori error control: Application to linear and nonlinear convection-dominated problems*, ICFD Conference on Numerical Methods for Fluid Dynamics, Oxford, England, March 31-April 3, 1998.

[18] C.L. Reed and D.A. Anderson, *Application of low speed preconditioning to the compressible Navier-Stokes equations*, AIAA Paper 97-0873, 1997.

[19] P.L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, J. Comp. Phys., 43, 1981, pp. 357-72.

[20] F. Shakib, T.J.R. Hughes and Z. Johan *A new finite element formulation for computational fluid dynamics : X. The compressible Euler and Navier-Stokes equations*, Comp. Meth. in Appl. Mech. and Engnr., 89, 1991, pp. 141-219.

[21] G.L.G. Sleijpen, H.A. van der Vorst and D.R. Fokkema, *BiCGstab(1) and other hybrid Bi-CG methods*, Numerical Algorithms, 7, 1994, pp.75-109.

[22] E. Turkel, *Symmetrization of Fluid Dynamic Matrices with Application*, Mathematics of Computation, 27, 1973, pp. 729-736.

[23] E. Turkel, A. Fiterman, and B. Van Leer, *Preconditioning and the limit to the incompressible flow equations for finite difference schemes*, Computing the Future: Advances and Prospects for Computational Aerodynamics, M. Hafez and D.A. Caughey, John Wiley and Sons, 1994, pp. 215-234.

[24] B. Van Leer, W.T. Lee, and P.L. Roe, *Characteristic time-stepping or local preconditioning of the Euler equations*, AIAA Paper 91-1552, 1991.

[25] J.M. Weiss and W.A. Smith, *Preconditioning applied to variable and constant density flows*, AIAA Journal, 33, 1995, pp. 2050-2057.

[26] J.S. Wong, D.L. Darmofal and J. Peraire, *Transonic finite element solution of the Euler equations using quadratic approximations*, in preparation 2000.

[27] *Test Cases for Inviscid Flow Field Methods*, AGARD-AR-211, 7 Ancelle 92200, Neuilly sur Seine, France, 1985.

[28] J.C. Carette, *Adaptive Unstructured Mesh Algorithms and SUPG Finite Element Method for Compressible High Reynolds Number Flows*, PhD Thesis, von Karman Institute, 1997

[29] J.C. Carette, H. Deconinck, H. Paillere, and P.L. Roe, *Multidimensional upwinding: Its relation to finite elements*, Int. J. of Num Meth. Fluids, vol.20 pp935-955, 1995.

[30] T. Okusanya, D. Darmofal and J. Peraire, *Algebraic Multigrid for Convection-Diffusion Flows*, MIT-FDRL Report, 2000.

[31] T.F. Chan, S. Go, and L. Zikatanov. *Lecture notes on multilevel methods for elliptic problems on unstructured grids.* 28th Computational Fluid Dynamics, 1–76, March 1997.

[32] R. Bank and J. Xu. *An algorithm for coarsening unstructured meshes.* Numerical Mathematics, 1–36, 1996.

[33] D.J. Mavriplis. *Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes.* ICASE, 1–22, 1998.

[34] T.F. Chan, J. Xu, and L. Zikatanov. *An agglomeration multigrid method for unstructured grids.* in preparation.

[35] J.W. Ruge and K. Stüben. *Algebraic multigrid.* Frontiers in Applied Mathematics, SIAM, 73–130, 1987.

[36] S.R. Elias, G.D. Stubley, and G.D. Raithby. *An adaptive agglomeration method for additive correction multigrid.* Int. J. for Num. Meth. in Engngr., 887–903, 1997.

[37] B.R. Hutchinson and G.D. Raithby. *A multigrid method based on the additive correction strategy.* Numerical Heat Transfer, 511–537, 1986.

[38] M. Raw. *Robustness of coupled algebraic multigrid for the navier-stokes equations.* AIAA J., 1–16, 1996.

[39] R. Webster. *An algebraic multigrid solver for navier-stokes problems.* Int. J. for Num. Meth. in Fluids, 761–780, 1994.

[40] P. Wesseling. *An Introduction to Multigrid Methods.* John Wiley & Sons, New York, 1992.

[41] D.J. Mavriplis. *Directional agglomeration multigrid techniques for high reynolds number viscous flow ICASE*, 1–20, 1998.