HIGH ORDER RESOLUTION AND PARALLEL IMPLEMENTATION ON UNSTRUCTURED GRIDS

A Thesis

by

YUFENG YAO

Submitted for the degree of

DOCTOR OF PHILOSOPHY

December 1996

University of Glasgow

Department of Aerospace Engineering

ABSTRACT

High Order Resolution and Parallel Implementation on Unstructured Grids. (December 1996)

Yufeng Yao, University of Glasgow

Supervisor: Professor B.E. Richards

In this thesis the numerical solution of the two-dimensional compressible Navier-Stokes equations for the application on aerodynamic problems is tackled. The motivation is to develop a cell-centred upwind finite volume scheme with high order accuracy and parallelism.

A general description of the two-dimensional compressible Navier-Stokes equations for application to computational fluid dynamics has been given, which forms the basis for the overall research throughout the thesis.

The numerical solution of the two-dimensional inviscid Euler flow equations is given. The unstructured mesh is generated by the advancing front technique. A cell-centred upwind finite volume method has been adopted to discretize the Euler equations. Both explicit and point implicit time stepping algorithms are derived. The flux calculation using Roe's and Osher's approximate Riemann solvers are studied. It is shown that both the Roe and Osher's schemes produce an accurate representation of discontinuities (e.g. shock wave). It is also shown that better convergence performance has been achieved by the point implicit scheme than that by the explicit scheme. Validations have been done for subsonic and transonic flow over airfoils, supersonic flow past a compression corner and hypersonic flow past cylinder and blunt body geometries. An adaptive remeshing procedure is also applied to the numerical solution with the objective of getting improved results.

The issue of high order reconstruction on unstructured grids has been discussed.

The methodology of the Taylor series expansion is adopted. The calculation of the gradient at a reference point is carried out by the use of either the Green-Gauss integral formula or the least-square methods. Some recently developed limiter construction methods have been used and their performance has been demonstrated using the test example of the transonic flow over a RAE 2822 airfoil. It has been shown that similar pressure distributions are obtained by all limiters except for shock wave regions where the limiter is active. The convergence problem is illustrated by the mid-mod type limiter. It seems only the Venkatakrishnan limiter provides improved convergence. Other limiters do not appear to work as well as that shown in their original publications. Also the convergence history given by the least-square method appears better than that by the Green-Gauss method in the test.

The formulation of the viscous terms in the Navier-Stokes equations and the implementation of a turbulence model on hybrid structured/unstructured grids are presented in detail. Different from the discretization method for the inviscid terms, the central-difference scheme is used for viscous terms. In general the unstructured grid is not suitable for the viscous problem, because highly stretched grids, which are necessary in viscous flow computation, cannot easily be created. Here a hybrid grid generation approach referred to here as the "skin" method is proposed. The resulting grid with a structured grid in the near-wall region and an unstructured grid in other regions has been successfully used in the laminar flow calculation. For high Reynolds number cases, the flow becomes turbulent. Implementation of the Baldwin-Lomax algebraic turbulence model in the N-S flow solver has been completed. Good performence has been shown by the test case of the subsonic flow over a NACA 0012 airfoil and the transonic flow over a RAE 2822 airfoil (case 9). Excellent agreements with experiments have been achieved.

The domain decomposition method on an unstructured grid has been discussed. The definition of the partitioning problem is illustrated. Some popularly used methods, i.e. recursive coordinate bisection (RCB), recursive graph bisection (RGB) (to-

gether with the reverse Cuthill-Mckee (RCM) ordering), recursive spectral bisection (RSB) and multilevel graph partitioning (MGP), have been described and their performances have been illustrated by application on airfoil problems. Some pre-ordering and smoothing algorithms have been proposed in order to improve the partitioning results by RCB and RGB methods. The domain dividing technique (DDT) is also discussed. In general the RSB and MGP partitioning methods produce the better results and MGP is much cheaper than RSB. All methods have been applied on either single or multi-element airfoils resulting in load balanced partitioning.

The development of the parallel code based on a cell-centred upwind finite volume method and the domain decomposition approach has been completed. By use of the RSB and MGP methods, a good quality (the number of edge cuts is minimized) load balanced partitioning is achieved. A method of constructing the message passing relations and data structures is proposed. The definition of three types of elements is given and illustrated by the example. The N-S flow solver is parallelized by coupling with the standard subroutines of the parallel virtual machine (PVM) package and has been successfully executed on a parallel computing system based on a workstation cluster under the PVM environment. Performence of parallel computing on airfoil problems has been demonstrated. Message passing, data structure (sending and receiving) and communication graphs have been illustrated. Reasonable efficiency in CPU time reduction and speedup has been achieved.

To Jun Yao, Weiyi, and My Parents

ACKNOWLEDGMENTS

The author would like to express his greatful thanks to his supervisor, Professor B.E. Richards, for his guidance and encouragement throughout this research.

The ORS award from the CVCP committee and the financial support from Professor B.E. Richards in the first year study and Glasgow University scholarship in the later two years would be acknowledged.

The author also wishes to thank Drs. Ken Badcock, X. Xu and D.C. Jiang for their discussions and encouragements. Special thanks to Mr. L. Dubuc with his fruitful initial work on explicit unstructured grid flow solver, which supplies a good starting platform for the further researches carried out in this thesis. Also his continuious discussions and helps should be specially mentioned here.

Thanks are also due to those staffs in the Department, especially to Angus and Indy for their help in computer services, and to Mrs. E. Garman and Miss M. Simpson, secretaries of the department for their effective help.

The encouragement from Prof. Caoqi Peng of Nanjing University of Aeronautics and Astronautics, former supervisor of author, and Prof. H.Y. Wong of Aerospace Engineering Department during this research period is specially acknowledged.

Finally, the author would like to express his most sincere thanks to his wife Jun Yao , his son Weiyi, for their understanding and support, to his parents and parents-in-law for their help and encouragement over all these years either in spiritual or in material.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Computational fluid dynamics: an introduction

Fluid dynamics is fundamental to many engineering disciplines. The governing equations of fluid flow, which are derived from the basic physical laws, are in the form of nonlinear partial differential equations. Analytical solutions of these equations do not exist except for some cases with simplified forms of equations and boundary conditions. Physicists, mathematicians and engineers were required to seek other ways of handling these equations. Experimental methods with special facilities, such as wind or water tunnels, have been used extensively to understand the behaviour of flow phenomena and useful results can be obtained in this way. Analysis based on model equations can also help the understanding of fluid flow. These methods are very useful but do have some disadvantages. The experimental approach is costly and is sometimes difficult (even impossible) to simulate the real conditions of complex problems in the laboratory environment. On the other hand, analysis cannot give overall solutions of realistic problems. With the growth in capabilities of the digital computer, an alternative approach is now coming available through the solution of the partial differential equations directly by using numerical discretization methods. This approach is termed Computational Fluid Dynamics (CFD).

Computational fluid dynamics is now one of the largest disciplines in the field of numerical analysis. Started because of the demands of the aircraft industry, it is currently applied in a wide range of industrial fields, such as mechanical, civil, chemical as well as aeronautical and astronautical engineering. With the focus on aerospace applications only those CFD issues which deal with compressible flows will be mentioned in this thesis. With the development of powerful digital computers, the simulation of fluid flow by numerical methods is becoming commonplace and in most cases far less expensive than experimental observations. Through the use of computer simulation, fluid properties, such as the Mach number and Reynolds number, can easily be controlled or varied. CFD may also be used to improve the understanding of complex physical phenomena.

Most problems of current interest in the aerospace industry can be satisfactorily modelled by the compressible Reynolds-averaged Navier-Stokes equations, or in some cases by its inviscid limit, the Euler equations. These equations represent the basic conservation laws in nature, i.e. mass, momentum and energy. The Navier-Stokes and

Euler equations have highly nonlinear behaviour and normally produce solutions with very high gradients (or discontinuities). Accurate resolution is thus crucial. Hence the trend is to apply CFD complementary to experimental method to reduce costs, especially in the preliminary stage. The responsibility of CFD researchers is then the development of more accurate, robust and lower cost computer codes applied to a wide range of flow conditions and various complex geometries.

1.2. Historical background

The paper published by Courant, Friedrichs and Lewy [1] in 1928 can be viewed as the foundation of numerical methods for fluid dynamics. The proposed characteristics theory and the CFL stability condition are still concepts widely used nowadays by CFD researchers. Following this, many schemes were developed. Those appropriate to be mentioned here are the Crank-Nicolson scheme [2] in 1947, the leapfrog scheme of Du Fort and Frankel [3] in 1953, the alternating direction implicit (ADI) scheme of Peaceman and Rachford [4] in 1955, the one-step Lax-Wendroff scheme [5] in 1960, the two-step Lax-Wendroff scheme [6] in 1963, the MacCormack explicit predictor-corrector scheme [7] in 1969 and the Beam-Warming scheme [8] in 1976.

Application of the CFD approach to a real problem involves several procedures. First an appropriate mathematical model is selected to describe the physical problems properly. Secondly, methods to discretize both the physical domain of interest and the equation itself is chosen. The method of discretization of the physical domain into a series of cells or elements was developed later as a sub-discipline in CFD termed grid generation. Basically there are three methods used to discretize the equations, i.e. Finite Difference (FD), Finite Volume (FV) and Finite Element (FE) methods. The third step is to solve the resulting linear or nonlinear systems. The second step together with the third step developed for the field is called the flow solution. Finally flow visualization must be considered for large problems with massive output data. This last process is called post-processing.

In the early days most of the numerical schemes were done by simply replacing the differential operators in the equations with finite differences without paying much attention to the physical property of the flow. This often lead to non-physical oscillations in the region of discontinuities. To overcome this, von Neumann and Richtmyer [9] introduced the artificial viscosity method. They added a diffusion term explicitly in the scheme. Later Courant et al [10] proposed another approach to the discretization of the equations, which was based on the physical behaviour of the flow. These researches set up the basis of what was called *Upwinding* or *Upstreaming Differencing* described later.

The term "Upwinding" was derived from the first application of it in numerical weather forecasting. In upwinding, information is transported in the direction which is physically right. By "upwinding", an implicit viscosity is introduced into the scheme that damps out the high frequency oscillations. Another advantage of upwinding is that a boundary condition can be applied in a manner which relates closely to the physical aspects of the flow. The disadvantage of upwinding is that it only achieves first order accuracy in space. Extension of upwinding to systems of equations is carried out using techniques based on the Godunov theorem [11], which solves, over each mesh interface, the locally one-dimensional Euler equations for discontinuous neighbouring states, equivalent to solving a Riemann problem, by splitting the fluxes of the conserved quantities and choosing the correct flow information for each component. Within this context, Steger and Warming proposed the flux vector splitting method [12] in 1981, Van Leer introduced another flux vector splitting scheme [13] in 1982 and on the flux difference splitting schemes, those of Roe's [14] [15] and Osher's [16] [17] have been very successfully and widely applied.

Upwinding schemes have been used mostly in conjunction with the finite volume method. In such an approach the equations normally take integral forms and are also satisfied over a single cell (element). Generally the finite difference methods require a topologically quadrilateral grid. However the finite volume schemes can be implemented on grids of arbitrarily shaped elements, single or mixed. The main assumption in all upwinding finite volume based schemes is that the solution is updated by considering the effects of wave propagation in the direction normal to the sides of the control volume. In spite of the deficiency of the assumption of one-dimensional flow cross the sides of the cell (element), the upwinding schemes have been able to simulate rather complicated flows. This approach is used in the present research.

The extension of upwinding to second order accuracy schemes has been developed. The methods can be divided into two classes. The first class is that of the hybrid schemes in which a monotone first order scheme is combined with a high order scheme to exploit the advantages of both. A good example of this method is the Flux Corrected Transport (FCT) scheme proposed by Boris and Book [18]. The Total Variation Diminishing (TVD) scheme introduced by Harten [19] can be viewed as a one step hybridisation approach. Shu [20] has proposed the Total Variation Bounded (TVB) scheme which weakens the TVD property by allowing for small-scale oscillations. The second class consists of those schemes which are based on the generalisation of the Godunov method [11]. High order accuracy is achieved by defining the distribution of the variables by a set of interpolation polynomials over cells (elements). A good example of that is the Monotone Upstream-centred Scheme for Conservation

Laws (MUSCL) algorithm proposed by van Leer [21]. Another scheme such as the Essentially Non-Oscillatory (ENO) scheme proposed by Harten et al [22] [23] is also of interest.

1.3. Flow solver and parallel computing on unstructured grids: literature review

Unstructured grids together with the finite element method have long been used in solid mechanics applications. Recently the use of the unstructured grid has received much attention in CFD calculations. The reason is that they provide flexibility in dealing with complex geometries and the ease of adapting to flow features, such as shocks and boundary layers. The disadvantage of employing unstructured grid is the increased requirement for computational time and memory. AGARD report R-787 gives a good introduction to unstructured grid related issues. Recently an overall survey on flow solvers for unstructured grids has been given by Venkatakrishnan [24]. In this thesis the literature review only highlights issues related to the present researches.

1.3.1. Finite volume discretization

The concept of using arbitrary control volumes to solve numerically the conservation laws was established by the late 1970s. For application on unstructured grids the first successful case is given by Jameson and Mavriplis [25]. In their paper the two-dimensional Euler equations are solved on regular triangular grids (obtained by halving the structured quadrilateral grids) by extending the similar schemes already applied on structured grids, i.e. cell-centred, finite volume, central-difference, artificial dissipation and multigrid ideas. A milestone in the field of flow solvers on unstructured grids was achieved in 1986 when Jameson et al [26] published their excellent paper about the calculation of inviscid transonic flow over a complete aircraft. Their contributions included the grid generation for a complex 3D geometry by the Delauney triangulation and the development of a cell-vertex concept. Since this seminal effort, significant advances have been made in grid generation and flow solvers on unstructured grids.

The drawback in [26] is that only a first order scheme is considered. For most practical problems, a high order scheme should be adopted. This follows the same trend as in the structured grid area, where much effort has been made in the area of the construction of the high order scheme. Desideri and Dervieux [27] derived cell-vertex finite volume schemes for unstructured grids using MUSCL ideas. Lohner et al [28] tested an FEM-FCT scheme for Euler/Navier-Stokes equations. Batina [29]

constructed an upwind scheme for cell-centred triangular grids that also employed MUSCL ideas, as did Knight [30], Frink [31] and Venkatakrishnan and Barth [32]. All these are the simple extensions of that used on structured grids onto an unstructured grid. In another approach Barth and Jesperson [33] proposed a novel concept by exploiting a one-dimensional departure to satisfy the monotonicity principles. They extended the monotonicity principle in multiple dimensions, namely that the reconstructed distributions in the control volume should be bounded everywhere by the values of the neighbours and satisfying such principles by constructing a so-called limiter. The limiter in [33] may be thought of as a generalized mid-mod type limiter, which in application however leads to convergence difficulties. Venkatakrishnan [34] has analysised this problem and proposed a modified limiter that improved the situation at the expense of monotonicity, as also done by Bishop and Noack [35] and Rosendale [36]. Recently Aftosmis et al [37] have found that the Venkatakrishnan limiter significently improve the convergence as well as the solution accuracy in their test examples. Frink [31] and Frink et al [38] developed an upwind cell-centred 3D flow solver without using limiters. They employed a weighted averaging procedure that interpolated variables from the centres of cells to the vertices and using these vertex values to calculate the gradients in each cell. These gradients were also used to interpolate the variables to the centres of the faces of the cells. Although the procedure was linear, it seems that enough dissipation had been introduced during the averaging and interpolating procedures.

For the viscous flow computation on unstructured grids, the discretization of viscous fluxes can be carried out either by finite volume or by finite element approaches. In references [31] and [38] the finite volume method was used with the central-difference scheme for viscous terms. Barth [39] discretized the viscous terms by using a finite element approach. Mavriplis [40] also developed an explicit cell-vertex finite element multigrid scheme. All these are implemented using efficient edge-based data structures.

1.3.2. Cell-centred and cell-vertex

On a given grid one has at least two choices as to where to locate the variables, giving rise to the cell-centred and cell-vertex approaches. In the cell-centred approach, the variables are stored at the centroid of the cells, whilst in the cell-vertex approach they are stored at the vertices of the grid. The best choice between cell-centred vs cell-vertex storage is still an open question, particularly in 3D cases. In 2D the ratio of the number of cells to the number of vertices is 2, whereas in 3D this ratio could be arbitrarily larger, although it is typically round 5 or 6 for a quality mesh. In the

case of tetrahedral grids with a cell-vertex scheme, the flux computation can be cast as loops over edges, whereas for the cell-centred scheme they must loop over faces. The ratio of the number of faces to the number of edges is roughly 2. From this view-point the cell-vertex schemes seem better than the cell-centred schemes. On the other hand there has some evidence that on a given grid the solution quality by the cell-centred scheme is superior to that by the cell-vertex scheme [41]. This is likely because the control volume in a cell-centred scheme is smaller than that in a cell-vertex scheme for triangular/tetrahedral grids. It is still not clear whether the cell-vertex schemes require a grid that has as many vertices as the number of tetrahedra used by a cell-centred schemes to achieve the same quality. But it is clear that a cell-vertex scheme is better suited to compute viscous fluxes when coupling with the finite element method.

1.3.3. Explicit and implicit schemes

After the discretization procedure (e.g. by the cell-centred finite volume method) the governing equations will result in large sparse linear systems. Different ways can be used in the updating process. An explicit scheme involves a simple and lower-storage method. By coupling with the multigrid technique it gives reasonable convergence speed [26]. Thareja et al [42] and Hassan et al [43] have utilized a point implicit iterative procedure. Batina [44] and Anderson and Bonhaus [45] have used a Gauss-Seidel relaxation technique. Their results show that convergence is speeded up compared to that of an explicit scheme. Further it is also possible to use more sophisticated techniques for the solution of the linear system, such as Generalized Minimal RESidual (GMRES). Venkatakrishnan and Mavriplis [46] reported the application of GMRES with Incomplete LU factorization(ILU) preconditioning on 2D Navier-Stokes solution on an unstructured grid. Another two implicit methods that have been investigated are based on "snakes" [47] and "linelets" [48].

1.3.4. Adaption

Adaption is one of the most important techniques to capture the complex flow phenomena. General adaptive mesh refinement (AMR) includes three distinct ways called r-refinement, h-refinement and p-refinement. In r-refinement the nodes are re-distributed so that the regions with complex fuid flows have more nodes clustered and thus will be better resolved. In h-refinement or so-called mesh enrichment, the cells are locally subdivided or merged or in some cases, a complete remeshing is done to replace the grid spacing in regions of interest. In p-refinement the degree of the basis function is adjusted locally by matching the variation in solution. A survey

has been carried out recently by Powell et al [49]. Peraire et al [50] has proposed an adaptive remeshing procedure by re-generation of the grid using an advancing front technique. Barth [51] and Lo [52] proposed the adaptive method based on the use of solution contours.

1.3.5. Unstructured grid for viscous flows

The construction of a suitable unstructured grid for viscous flow simulation is still a field under development. The triangular grids generated by either the Delaunay triangulation or the advancing front technique do not naturally lend themselves to viscous flow computation, in which highly stretched cells are required in the viscous regions. It appears that it is essential to have a structured-like grid in boundary layer regions. Thompson and Weatherill [53] identify three categories of approaches. A brief description is given here. The first category is to consider the generation of a stretched unstructured grid by Delaunay triangulation, in which a mapping procedure is used to obtain a very high-aspect-ratio grid in boundary and wake regions [54] [55] [56]. However the extension of this approach to the 3D case has not yet proved successfully for general complex configurations. The second one is to use an intermediate stage which involves a modified form of standard unstructured grid generation. Hassan et al [57] and Pirzadeh [58] advocated this approach. In the third category, the approach is to create a thin layer around a given geometry within which a structured grid is created. Weatherill et al [59] and Holmes and Connell [60] tested this approach. In viscous flow computation the trend is to use the hybrid grid with a structured or semistructured grid in the near wall regions. In the 2D case a structured body-fitted grid is used whereas in the 3D case a prismatic grid is adopted.

1.3.6. Turbulence Modelling

Some popular turbulence models, such as the Baldwin-Lomax (B-L) algebraic model [61], have been used for implementation on unstructured grids in the early days of N-S solver research. Mavriplis [54] introduced a reference grid on which the B-L turbulence model is calculated and interpolated with the values on the global unstructured grid. Kallinderis [62] did similarly. With further development the trend is away from the simple algebraic model to more sophisticated field turbulence models like the one-equation models of Baldwin and Barth (B-B) [63] and Spalart and Allmaras (S-A) [64] and the two-equation models such as $k - \epsilon$. Mavriplis and Martinelli [65] reported their application of a $k - \epsilon$ model on two-dimensional Navier-Stokes solutions. Anderson and Bonhaus [45] published some progress on the use of B-B and S-A one-equation models applied in multi-element airfoil flows.

1.3.7. Domain decomposition

Domain decomposition (or graph partitioning) is an important problem that has extensively been used in many areas, particularly in parallel computing. The problem is to partition the vertices of a graph in p roughly equal parts, such that the number of edges connecting vertices in different parts is minimized. Mainly there are three classes of partitioning method. The first class is called recursive partitioning, which divides the domain into two sub-domains and each sub-domain is sub-divided into two more sub-domains and the process is repeated until the desired number of partitions is obtained. References [66] and [67] give the detailed discription and comparison of three recursive methods, i.e. recursive coordinate bisection (RCB), recursive graph bisection (RGB) and recursive spectral bisection (RSB). The RSB is improved as multilevel spectral bisection (MSB) by Barnard and Simon [68] in order to reduce the expense of RSB. The second class of graph partitioning is called geometric partitioning based on the geometric information of the graph. Geometric partitioning algorithms such as Miller et al [69] and [70] tend to be fast but often yield partitions that are worse than those obtained by the RSB or MSB methods. The third class of graph partitioning algorithm is called multilevel graph partitioning (MGP) [71] [72]. The MGP method works in three phases. In phase one it reduces the size of the graph (i.e. coarsens the graph) by collapsing vertices and edges. In phase two the partitioning of the small graph is completed. In phase three the graph is refined back to the original one. It is evident that in application the same good quality can be obtained by MGP as by RSB and MSB but the cost of the MGP method is much lower.

1.3.8. Parallel computing

The target of CFD research is to complete the solution within a practical time period. In the last 15 years CFD has greatly benefited from the revolution happening both in the computer industry and in computer science. The appearance of parallel computer systems offers more powerful tools for CFD computations. Except for the partitioning problems mentioned above, the other issues are message passing, data structure and parallel algorithms. Unstructured grid flow solvers have been implemented on various parallel machines, such as the Connection Machine [73] [74], Cray and iPSC/860 machines [75] [76]. These studies have shown that good performance may be obtained by paying careful attention to the issues above. Regarding the flow solver, explicit and point implicit schemes contain features of almost complete parallelism, except for the communication between neighbouring processors. For implicit schemes good performance can also be yielded by careful design [77] [78].

1.4. Scope of this thesis

The theme of this thesis is to study the cell-centred upwind finite volume method for the solution of the compressible Euler/Navier-Stokes equations with the implementation of high order resolution and parallel computing on unstructured grids.

Chapter 1 gives a general review on previous relevant research. The focus is mainly around the flow solver on unstructured grids.

In chapter 2, the mathematical model of the two-dimensional compressible Navier-Stokes equations is described.

The cell-centred upwind finite volume schemes for the solution of the 2D Euler equations are presented in chapter 3. The unstructured grid is generated using the advancing front technique. Both the numerical fluxes of Roe and Osher are considered. Explicit and point implicit iterative algorithms are derived and comparisons made by application to a wide range of flows from subsonic to hypersonic cases. An adaptive remeshing procedure is applied during the calculations.

The construction of a high order accurate scheme on unstructured grids is addressed in chapter 4. A general method is proposed based on Taylor series expansions. Two key problems are the calculation of gradients and the definition of limiters. The Green-Gauss integral formula and the least-square method are used for the gradient calculations. Some limiters, such as the Barth-Jesperson limiter, Venkatakrishnan limiter, Albada limiter, Bishop-Noack limiter and the general Albada limiter, are discussed and compared in the applications to test cases.

The discretization of the viscous terms in the upwind scheme and the implementation of turbulence model is the subject of chapter 5. A hybrid structured/unstructured grid generation method is proposed to fulfill the requirements in viscous flow computations. The point implicit formulation of viscous terms is developed in detail. Laminar flow is considered using the NACA 0012 airfoil test case A5. The implementation of the Baldwin-Lomax algebraic turbulence model on a hybrid grid is described. The performance of the hybrid grid with the NS flow solver will be demonstrated by application to NACA 0012 airfoil subsonic and RAE 2822 airfoil transonic turbulence flow (case 9) cases.

The discussion on the domain decomposition method is made in chapter 6. The definition of the partitioning issue is given firstly. Several efficient and popularly used methods, like recursive coordinate bisection (RCB), recursive graph bisection (RGB), recursive spectral bisection (RSB) and multilevel graph partitioning (MGP), are discussed and the comparisons made by the application of partitioning to unstructured grid problems.

To develop a parallel version of the NS flow solver is the subject discussed in

chapter 7. A brief description of the graph partitioning problem is given firstly. Then the focus turns to the message passing issue and the construction of the data structure. Three types of communication elements are defined. Parallel computing is carried on a workstation cluster, a parallel computing system based on a distributed network, under a parallel virtual machine (PVM) environment. Performance optimization is given through comparisons of the reduction of CPU time and speed-up in efficiency.

Finally conclusions and the suggestion of further researches are given in chapter 8.

CHAPTER 2

MATHEMATICAL MODELLING

2.1. Introduction

Most flow problems in aerodynamics can be suitably modelled mathematically by the full Navier-Stokes equations, which are originally derived from three conservation laws in nature, i.e. (i) Mass can be neither created nor destroyed; (ii) The time rate of change of momentum of a body equals the net force exerted on it; (iii) Energy can be neither created nor destroyed, it can only change in form. In the Navier-Stokes equations there are two main items, i.e. convective fluxes and diffusive fluxes. The convective fluxes appear in first-order derivative forms and describe the transport properties in the fluid flow, while the diffusive fluxes appear as second-order derivative terms and express the essence of the molecular diffusion phenomenon. Each of them will influence the mathematical feature of the Navier-Stokes equations, i.e. highly non-linear and mixed elliptic, parabolic and hyperbolic types of the equations. The solutions produced often exhibit very high-gradients or discontinuities in some cases, e.g. shock waves. Despite the high-complexity of the Navier-Stokes equations, solving the full equations is still the ultimate goal for CFD researchers.

In the following section 2.2 the two-dimensional Navier-Stokes equations in conservation law forms will be described. Section 2.3 illustrates the state equation and section 2.4 gives the non-dimensional form of Navier-Stokes equations.

2.2. Navier-Stokes equations in conservation law

The flow of a compressible heat conducting viscous fluid is governed by the Navier-Stokes equations. These equations represent the conservation law of mass, momentum and energy. The time-dependent two-dimensional Navier-Stokes equations in Cartesian coordinate system (x,y) can be expressed as follows:

The conservation law of mass:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0 \tag{2.1}$$

The conservation law of momentum:

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial(\rho u^2+p)}{\partial x}+\frac{\partial(\rho uv)}{\partial y}=\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y} \tag{2.2}$$

$$\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho vu)}{\partial x}+\frac{\partial(\rho v^2+p)}{\partial y}=\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y} \tag{2.3}$$

The conservation law of energy:

$$\frac{\partial(\rho \varepsilon)}{\partial t}+\frac{\partial[u(\rho \varepsilon+p)]}{\partial x}+\frac{\partial[v(\rho \varepsilon+p)]}{\partial y}=\frac{\partial(u\tau_{xx}+v\tau_{xy}-q_x)}{\partial x}+\frac{\partial(u\tau_{yx}+v\tau_{yy}-q_y)}{\partial y} \tag{2.4}$$

where ρ, u, v, p and ε are the density, velocity components in the cartesian coordinates, pressure and specific total energy of the flow. Symbol τ represents the stress tensor and q is the heat flux vector. These quantities are related to the velocity or temperature gradients by the following relationships under the assumption of Newtonian fluid:

$$\tau_{xx}=2\mu\frac{\partial u}{\partial x}+\lambda(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}) \tag{2.5}$$

$$\tau_{xy}=\mu(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}) \tag{2.6}$$

$$\tau_{yx}=\mu(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}) \tag{2.7}$$

$$\tau_{yy}=2\mu\frac{\partial v}{\partial y}+\lambda(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}) \tag{2.8}$$

$$q_x=-k\frac{\partial T}{\partial x} \tag{2.9}$$

$$q_y=-k\frac{\partial T}{\partial y} \tag{2.10}$$

Here μ and λ represent the viscosity coefficient and second viscosity coefficient, which charactizes a Newtonian fluid. The symbol k is the coefficient of thermal

conductivity and T is the temperature of the fluid.

Generally we consider that the fluid behaviour is in local thermodynamic equilibrium, for which the Stokes relation $\lambda = -\frac{2}{3}\mu$ is valid. As a result the shear stresses can be expressed with only the viscosity coefficient μ.

2.3. State equations

In order to close the NS equations described above it is necessary to construct the relationships between the thermodynamic variables (p,ρ,T) as well as to relate the transport properties (μ, k) to these thermodynamic variables. Under the assumption of a perfect gas the following relationship can be established.

The state equation of a perfect gas is

$$p = \rho RT \tag{2.11}$$

where R is the gas constant, and for standard air, $R = 287 m^2/s^2 K$

Also the following relationships exist:

$$e = c_v T; \quad h = c_p T; \quad \gamma = \frac{c_p}{c_v}; \quad c_v = \frac{R}{\gamma - 1}; \quad c_p = \frac{\gamma R}{\gamma - 1}, \tag{2.12}$$

where e is the internal energy per unit mass; h is the enthalpy per unit mass; c_v is the specific heat at constant volume; c_p is the specific heat at constant pressure and γ is the ratio of specific heats.

Further equations can be derived from equations (2.11) and (2.12) as

$$p = (\gamma - 1)\rho \left[\varepsilon - \frac{1}{2}(u^2 + v^2)\right] \tag{2.13}$$

$$T = \frac{\gamma - 1}{R} \left[\varepsilon - \frac{1}{2}(u^2 + v^2)\right] \tag{2.14}$$

The coefficient of viscosity μ and thermal conductivity k have to be related to the thermodynamic variables using kinetic theory. For example the coefficient of viscosity is expressed by the Sutherland's formula as

$$\mu = s_1 \frac{T^{\frac{3}{2}}}{T + s_2} \tag{2.15}$$

where s_1 and s_2 are experimental constants. For air at moderate temperatures $s_1 = 1.458 \times 10^{-6} k_g/(ms\sqrt{K})$ and s_2 is taken as $110^o K$.

k is related to μ through the Prandtl number which is a constant and can be expressed as

$$P_r = \frac{c_p \mu}{k} \tag{2.16}$$

For air at standard states, P_r=0.72 is assumed.

2.4. The non-dimensional form of NS equations

To obtain the flow behaviour around geometries with similar shape at minimum computational effort, it is desirable to re-write the NS equation in non-dimensional form.

The following non-dimensionalizing procedures are adopted:

$$x^* = \frac{x}{L}; \quad y^* = \frac{y}{L}; \quad t^* = \frac{tU_\infty}{L},$$

$$u^* = \frac{u}{U_\infty}; \quad v^* = \frac{v}{U_\infty},$$

$$\rho^* = \frac{\rho}{\rho_\infty}; \quad p^* = \frac{p}{\rho_\infty U_\infty^2}; \quad \varepsilon^* = \frac{\varepsilon}{U_\infty^2},$$

$$\mu^* = \frac{\mu}{\mu_\infty}; \quad T^* = \frac{T}{T_\infty},$$

where the non-dimensional variables are denoted by an asterisk. Free-stream conditions are denoted by ∞ and L is a representative length for the problem used in the Reynolds number

$$Re_\infty = \frac{\rho_\infty U_\infty L}{\mu_\infty} \tag{2.17}$$

If the non-dimensionalizing procedure is applied to equations (2.1), (2.2), (2.3) and (2.4), the following non-dimensional equations in vector form can be obtained

$$\frac{\partial U^*}{\partial t^*} + \frac{\partial F_1^*}{\partial x^*} + \frac{\partial F_2^*}{\partial y^*} = \frac{\partial G_1^*}{\partial x^*} + \frac{\partial G_2^*}{\partial y^*} \tag{2.18}$$

where U^* is the vector of conservation variables, F_1^*, F_2^* are the inviscid flux vectors and G_1^*, G_2^* are the viscous flux vectors.

$$U^* = \begin{pmatrix} \rho^* \\ \rho^* u^* \\ \rho^* v^* \\ \rho^* \varepsilon^* \end{pmatrix} \tag{2.19}$$

$$F_1^* = \begin{pmatrix} \rho^* u^* \\ \rho^* u^{*2} + p^* \\ \rho^* u^* v^* \\ u^*(\rho^* \varepsilon^* + p^*) \end{pmatrix} \quad F_2^* = \begin{pmatrix} \rho^* v^* \\ \rho^* v^* u^* \\ \rho^* v^{*2} + p^* \\ v^*(\rho^* \varepsilon^* + p^*) \end{pmatrix} \tag{2.20}$$

$$G_1^* = \begin{pmatrix} 0 \\ \tau_{xx}^* \\ \tau_{xy}^* \\ u^* \tau_{xx}^* + v^* \tau_{xy}^* - q_x^* \end{pmatrix} \quad G_2^* = \begin{pmatrix} 0 \\ \tau_{yx}^* \\ \tau_{yy}^* \\ u^* \tau_{yx}^* + v^* \tau_{yy}^* - q_y^* \end{pmatrix} \tag{2.21}$$

The formulas of the shear stress tensor and heat flux vector in non-dimensional form are as follows

$$\tau_{xx}^* = \frac{1}{Re_\infty}[2\mu^* \frac{\partial u^*}{\partial x^*} - \frac{2}{3}\mu^*(\frac{\partial u^*}{\partial x^*} + \frac{\partial v^*}{\partial y^*})] \tag{2.22}$$

$$\tau_{xy}^* = \frac{1}{Re_\infty}\mu^*(\frac{\partial u^*}{\partial y^*} + \frac{\partial v^*}{\partial x^*}) \tag{2.23}$$

$$\tau_{yx}^* = \frac{1}{Re_\infty}\mu^*(\frac{\partial u^*}{\partial y^*} + \frac{\partial v^*}{\partial x^*}) \tag{2.24}$$

$$\tau_{yy}^* = \frac{1}{Re_\infty}[2\mu^* \frac{\partial v^*}{\partial y^*} - \frac{2}{3}\mu^*(\frac{\partial u^*}{\partial x^*} + \frac{\partial v^*}{\partial y^*})] \tag{2.25}$$

$$q_x^* = -\frac{\mu^*}{(\gamma - 1)M_\infty^2 Re_\infty P_r}\frac{\partial T^*}{\partial x^*} \tag{2.26}$$

$$q_y^* = -\frac{\mu^*}{(\gamma-1)M_\infty^2 Re_\infty P_r}\frac{\partial T^*}{\partial y^*} \tag{2.27}$$

where M_∞ is the free-stream Mach number defined as

$$M_\infty = \frac{U_\infty}{c_\infty} \tag{2.28}$$

The speed of sound in the free-stream is related to other variables through

$$c_\infty^2 = \frac{\gamma p_\infty}{\rho_\infty} \tag{2.29}$$

The state equations of perfect gas become

$$p^* = (\gamma-1)\rho^*\left[\varepsilon^* - \frac{1}{2}(u^{*2}+v^{*2})\right] \tag{2.30}$$

and

$$T^* = \frac{\gamma-1}{R}\left[\varepsilon^* - \frac{1}{2}(u^{*2}+v^{*2})\right] \tag{2.31}$$

or in terms of the Mach number by

$$T^* = \frac{\gamma M_\infty^2 p^*}{\rho^*} \tag{2.32}$$

For inviscid flow, equations (2.18) become the Euler equations

$$\frac{\partial U^*}{\partial t^*} + \frac{\partial F_1^*}{\partial x^*} + \frac{\partial F_2^*}{\partial y^*} = 0 \tag{2.33}$$

In the following chapters the asterisk will be dropped from the non-dimensional equation for convenience.

CHAPTER 3

NUMERICAL SOLUTION OF TWO-DIMENSIONAL EULER EQUATION

3.1. Introduction

Computational fluid dynamics has now progressed to a stage where the simulation of compressible, inviscid flow modelled by the Euler equations, is commonplace. Such simulation normally includes two main parts, i.e. to generate a grid system over the flow domain and to develop a flow solver for the governing equations, e.g. the Euler equations, by numerical approximation methods.

Generally two different types of grids, i.e. structured and unstructured grids, are used. Grid generation techniques have been recently reviewed in survey papers by Thompson and Weatherill [53] and Mavriplis [79]. Although structured grids have the advantage of simplicity for implementation of numerical algorithms, they still meet the challenges in their capability of generating grids around general complex geometries. Special techniques such as multi-block [108] or chimera grids are commonly used to solve this problem. On the other hand unstructured grids, which have long been used by the finite element community, offer a means of treating very complex geometries. And they also provide the flexibility of mesh adaptivity which is also important for complex flows. Because of these we mainly focus our research using the unstructured grid.

There are many numerical approximation methods developed to solve flow problems on unstructured grids. Most of them can be traced in the recent review paper of Venkatakrishnan [24] on flow solvers on unstructured grids. Typically in the area of Euler flow solvers, one approach, which has been used frequently, is upwinding together with the finite volume method. The strength of upwinding is its capacity of treating discontinuities in the flow field (e.g. shock waves) and its ability to simulate directly the physics of the directional propagation of information. The most commonly used algorithms are the flux vector splitting methods of Steger and Warming [12] and van Leer [13] and the flux difference splitting methods of Roe [14] [15] and Osher [16] [17]. These techniques, traditionally used on structured grids, have been successfully applied on unstructured grids recently [31] [32] [33] [39] [42] [44] [80].

Also in compressible flow there often appears a narrow region with high gradients embedded in large areas in which flow variables vary slowly. As the position of these high gradient regions is not known to the analysist a priori, it is apparent that

adaptive mesh techniques will play an important role in the efficient solution of such flow problems.

In this chapter an upwinding-biased finite volume method with a flux difference splitting algorithm is discussed and applied to the numerical solution of the compressible two-dimensional inviscid Euler equations on unstructured grids. The discretization method is based on the cell-centred concept with explicit and point-implicit time-stepping iteration procedure. The advancing front technique (AFT) [50] has been used to generate the unstructured grid. An adaptive remeshing procedure based on a series of successive analysis of the flow has been used to improve the quality of simulation. The above approach will be demonstrated on problems covering a wide range of flow speeds.

Section 3.2 describes the integral form of the Euler equations and its finite volume discretization by the cell-centred concept. Section 3.3 discusses upwinding flux difference splitting algorithms. Both Roe and Osher's approximate Riemann solver on an unstructured grid will be mentioned. Section 3.4 describes the explicit and point-implicit iteration scheme. Section 3.5 discusses the boundary condition related to inviscid Euler flow. Section 3.6 describes the Monotone Upstream-centred Scheme for Conservation Laws (MUSCL) approach to construct the linear-resolution scheme on an unstructured grid. Section 3.7 describes the unstructured grid generator and an adaptive remeshing method. Finally some numerical results are illustrated in section 3.8.

3.2. Euler equation in integral form and discretization

The numerical method employed in this chapter is an implementation of the finite volume (FV) method on unstructured grids. Using the finite volume technique the integral form of the conservation laws can be discretized directly in the physical domain. It takes full advantage of adopting the arbitrary grid (e.g. single or mixed type) and thus avoids the transformation calculation between physical and computational domains, as normally do on structured grids.

Generally the two-dimensional compressible inviscid Euler equations (2.33) in the conservation laws can be written in their integral form as

$$\iint_{\Omega} \frac{\partial U}{\partial t} d\Omega + \iint_{\Omega} \left(\frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} \right) d\Omega = 0 \tag{3.1}$$

where Ω represents the physical flow domain of interest.

Under the philosophy of unstructured grid methods, the flow domain is discretized to a series of triangular or quadrilateral elements or a mixture of both. The

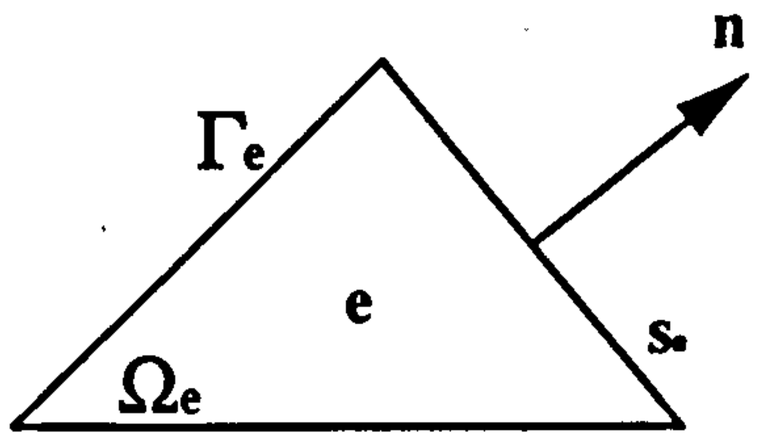


Fig. 1. Notations of the single element

above governing equation in integral form is also valid when applied to a single element "e". By using the divergence theorem the following equation can be derived

$$\iint_{\Omega_e} \frac{\partial U}{\partial t} d\Omega + \oint_{\Gamma_e} F_n d\Gamma = 0 \tag{3.2}$$

Where Ω_e is the area of element "e", Γ_e is the boundary of the element. Normal flux is defined as $F_n = \vec{F}_i \cdot \vec{n}_i$, in which $\vec{F}_i = (F_1, F_2)$ denotes the inviscid vector fluxes. Also $\vec{n}_i = (n_x, n_y)$ denotes the unit vector outward normal to the boundary Γ_e of control volume Ω_e (see figure 1).

Based on the cell-centred concept we assume a piecewise constant distribution of the unknown variables U_e on the single element "e", thus equation (3.2) can be approximated in the form as

$$\frac{\Delta U_e}{\Delta t_e} \Omega_e + F^I = 0 \tag{3.3}$$

Where $\Delta U_e = U_e^{n+1} - U_e^n$. The symbol U_e^n denotes the values of U_e at time level $t = t^n$. U_e^{n+1} denotes the values of U_e at time level $t = t^{n+1}$. The symbol $\Delta t_e = t_e^{n+1} - t_e^n$ is the time step and F^I denotes the inviscid flux contributions defined as

$$F^I = \oint_{\Gamma_e} F_n d\Gamma \tag{3.4}$$

which can be evaluated by summing the contributions from each individual element side Γ_{es},

$$F^I = \sum_{s_e} \oint_{\Gamma_{es}} F_n d\Gamma \tag{3.5}$$

Further the normal flux F_n can be replaced approximately by a numerical flux $\tilde{F}_n$ evaluated at the middle point of each side s_e, so that

$$F^I = \sum_{s_e} \tilde{F}_n \delta s_e \tag{3.6}$$

where δs_e denotes the length of side s_e (figure 1).

3.3. Upwinding flux difference splitting algorithm

To evaluate the inviscid numerical flux $\tilde{F}_n$, two methods of approximate Riemann solver developed individually by Roe [14] [15] and Osher [16] [17] are applied locally at each interface between neighbouring cells, assuming a local Riemann problem in the normal direction of the interface.

3.3.1. Numerical flux by Roe's approximate Riemann solver

The numerical flux of Roe's approximate Riemann solver can be written in terms of two discrete Riemann states at the left (denoted by e) and right (denoted by r) side, with respect to the interface as:

$$\tilde{F}_n(U_e, U_r) = \frac{1}{2}[F(U_e) + F(U_r) - |A_{Roe}|(U_r - U_e)] \tag{3.7}$$

Where $F(U_e)$ and $F(U_r)$ represent the fluxes on the left and right sides, which can be obtained from

$$F(U_e) = \begin{pmatrix} \rho_e(U_e)_n \\ (\rho_e u_e)(U_e)_n + p_e n_x \\ (\rho_e v_e)(U_e)_n + p_e n_y \\ (U_e)_n(\rho_e \varepsilon_e + p_e) \end{pmatrix}; \quad F(U_r) = \begin{pmatrix} \rho_r(U_r)_n \\ (\rho_r u_r)(U_r)_n + p_r n_x \\ (\rho_r v_r)(U_r)_n + p_r n_y \\ (U_r)_n(\rho_r \varepsilon_r + p_r) \end{pmatrix}, \tag{3.8}$$

in which

$$(U_e)_n = u_e n_x + v_e n_y \tag{3.9}$$

$$(U_r)_n = u_r n_x + v_r n_y \tag{3.10}$$

And matrix A_{Roe} is the flux Jacobian matrix which is defined so as to satisfy the following properties:

(i) $F(U_e) - F(U_r) = A(U_e, U_r)(U_r - U_e)$
(ii) $A(U_e, U_e) = A(U_e)$
(iii) the eigenvectors of $A(U_e, U_r)$ are linearly independent.

The absolute value symbols in equation (3.7) indicate that the absolute value of the eigenvalues were used to evalute A_{Roe}. Furthermore matrix $|A_{Roe}|$ can be

decomposed in terms of its eigenvectors and eigenvalues as

$$|A_{Roe}| = R|\Lambda|R^{-1} \tag{3.11}$$

Where R, R^{-1} denote the right and left eigenvectors respectively and Λ is a diagonal matrix containing the eigenvalues λ_i of matrix A_{Roe}.

The matrix of the right eigenvector R is given by

$$R = \begin{pmatrix} 1 & 1 & 1 & 0 \\ u - cn_x & u & u + cn_x & -n_y \\ v - cn_y & v & v + cn_y & n_x \\ H - cU_n & \frac{u^2+v^2}{2} & H + cU_n & V_t \end{pmatrix} \tag{3.12}$$

The matrix of the left eigenvector R^{-1} is given by

$$R^{-1} = \begin{pmatrix} \frac{1}{2}(b_1 + \frac{U_n}{c}) & \frac{1}{2}(-b_2 u - \frac{n_x}{c}) & \frac{1}{2}(-b_2 v - \frac{n_y}{c}) & \frac{b_2}{2} \\ 1 - b_1 & b_2 u & b_2 v & -b_2 \\ \frac{1}{2}(b_1 - \frac{U_n}{c}) & \frac{1}{2}(-b_2 u + \frac{n_x}{c}) & \frac{1}{2}(-b_2 v + \frac{n_y}{c}) & \frac{b_2}{2} \\ -V_t & -n_y & n_x & 0 \end{pmatrix} \tag{3.13}$$

and the diagonal matrix of the eigenvalue is by

$$\Lambda = \begin{pmatrix} U_n - c & 0 & 0 & 0 \\ 0 & U_n & 0 & 0 \\ 0 & 0 & U_n + c & 0 \\ 0 & 0 & 0 & U_n \end{pmatrix} \tag{3.14}$$

In equations (3.12), (3.13) and (3.14) , U_n and V_t are the normal and tangential components of the velocity to the interface. They are defined by the following

$$U_n = un_x + vn_y \tag{3.15}$$

$$V_t = -un_y + vn_x \tag{3.16}$$

In equation (3.12) above, H is total enthalpy which is defined as

$$H = \gamma\varepsilon - \frac{\gamma - 1}{2}(u^2 + v^2) \tag{3.17}$$

and in equation (3.13), the coefficients b_1 and b_2 are defined as

$$b_1 = b_2\frac{(u^2+v^2)}{2} \quad b_2 = \frac{\gamma-1}{c^2}$$

In order to ensure Roe's properties (i)-(iii) the flow variables in (3.12), (3.13) and (3.14) must use the following average state values

$$u = \frac{u_e + R_\rho u_r}{1+R_\rho} \tag{3.18}$$

$$v = \frac{v_e + R_\rho v_r}{1+R_\rho} \tag{3.19}$$

$$H = \frac{H_e + R_\rho H_r}{1+R_\rho} \tag{3.20}$$

$$R_\rho = \sqrt{\frac{\rho_e}{\rho_r}} \tag{3.21}$$

The corresponding speed of sound is then determined by

$$c = (\gamma-1)(H - \frac{u^2+v^2}{2}) \tag{3.22}$$

If the eigenvalue of matrix A_{Roe} equals zero then the flux formula given by (3.7) may lead to non-physical expansion shocks. To avoid this a local expansion fan is introduced in the approximate Riemann solver when an expansion is detected through a sonic point. This can be done by restricting the mininum allowable value for λ_i in equation (3.14). According to the method proposed by Harten [81] that is :

$$|\lambda_i| = \begin{cases} |\lambda_i| & \text{if } |\lambda_i| > \epsilon_\lambda \\ 0.5(\lambda_i^2/\epsilon_\lambda + \epsilon_\lambda) & \text{if } |\lambda_i| \leq \epsilon_\lambda \end{cases} \tag{3.23}$$

where ϵ_λ is the eigenvalue limiter. Normally we set ϵ_λ between 0.1 to 0.3 .

3.3.2. Numerical flux by Osher's approximate Riemann solver

The numerical flux of Osher in terms of the left and the right states can be defined as

$$\tilde{F}_n(U_e, U_r) = \frac{1}{2}\left[F(U_e) + F(U_r) - \int_{U_e}^{U_r} |A_{Osher}| dQ\right] \tag{3.24}$$

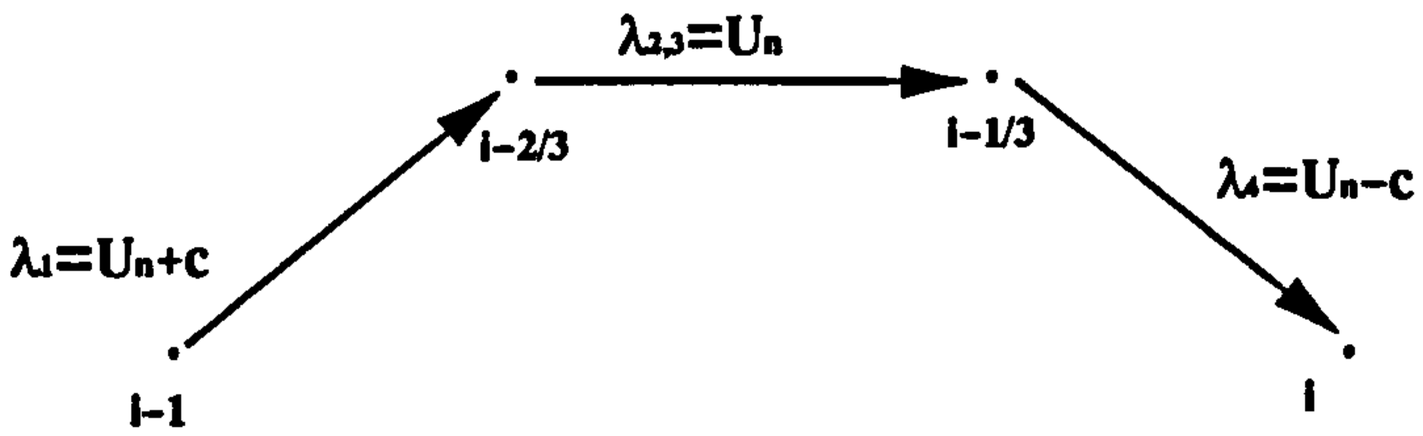


Fig. 2. Integration path for Osher's flux

where the fluxes at the right and left states are calculated in the same way as in Roe's scheme. The integration in the above expression is performed by the following procedure.

Considering 2-D flow, there are four characteristic fields of which the two corresponding to $\lambda_{2,3}$ are identical. The invariant functions are:

For $\lambda_1 = U_n + c$ we have

$$\Psi_2^1 = U_n - \frac{2c}{\gamma - 1} \quad \Psi_3^1 = \frac{p}{\rho^\gamma} \quad \Psi_4^1 = V_t \tag{3.25}$$

For $\lambda_{2,3} = U_n$ we have

$$\Psi_1^{2,3} = p \quad \Psi_4^{2,3} = U_n \tag{3.26}$$

For $\lambda_4 = U_n - c$ we have

$$\Psi_1^4 = U_n + \frac{2c}{\gamma - 1} \quad \Psi_2^4 = \frac{p}{\rho^\gamma} \quad \Psi_3^4 = V_t \tag{3.27}$$

In the above expressions, U_n and V_t are the normal and tangential velocities to the cell side defined as before. And c is the local speed of sound.

The first and fourth characteristic fields are non-linear and the second and third are linear. The path of integration in the state space is as shown in figure 2.

By writing the invariant functions along each subpath, we obtain eight equations which can be solved to get the eight variables that define the value of the intermediate points. They are

$$\rho_{i-\frac{2}{3}}^{\left(\frac{\gamma-1}{2}\right)} = \frac{\left[\frac{\gamma-1}{2}((U_n)_i - (U_n)_{i-1}) + c_i + c_{i-1}\right]}{c_{i-1}\left[1 + \left(\frac{p_i}{p_{i-1}}\right)^{\frac{1}{2\gamma}}\left(\frac{\rho_{i-1}}{\rho_i}\right)^{\frac{1}{2}}\right]} \rho_{i-1}^{\frac{\gamma-1}{2}} \tag{3.28}$$

$$\rho_{i-\frac{1}{3}}^{\left(\frac{\gamma-1}{2}\right)} = \frac{\left[\frac{\gamma-1}{2}((U_n)_i - (U_n)_{i-1}) + c_i + c_{i-1}\right]}{c_i\left[1 + \left(\frac{p_{i-1}}{p_i}\right)^{\frac{1}{2\gamma}}\left(\frac{\rho_i}{\rho_{i-1}}\right)^{\frac{1}{2}}\right]} \rho_i^{\frac{\gamma-1}{2}} \tag{3.29}$$

$$p_{i-\frac{2}{3}} = p_{i-1}\left(\frac{\rho_{i-\frac{2}{3}}}{\rho_{i-1}}\right)^{\gamma} \tag{3.30}$$

$$p_{i-\frac{1}{3}} = p_i\left(\frac{\rho_{i-\frac{1}{3}}}{\rho_i}\right)^{\gamma} \tag{3.31}$$

$$(U_n)_{i-\frac{2}{3}} = (U_n)_{i-1} - \frac{2}{\gamma-1}(c_{i-1} - c_{i-\frac{2}{3}}) \tag{3.32}$$

$$(U_n)_{i-\frac{1}{3}} = (U_n)_i + \frac{2}{\gamma-1}(c_i - c_{i-\frac{1}{3}}) \tag{3.33}$$

$$(V_t)_{i-\frac{2}{3}} = (V_t)_{i-1} \tag{3.34}$$

$$(V_t)_{i-\frac{1}{3}} = (V_t)_i \tag{3.35}$$

The sonic points (denoted by "s") are determined using a similar procedure. There is no sonic point on the second subpath. For the first and third subpaths, there are

$$(\rho^s)_{i-\frac{2}{3}}^{\left(\frac{\gamma-1}{2}\right)} = \frac{-(U_n^s)_{i-\frac{2}{3}}}{c_{i-1}}\rho_{i-1}^{\frac{\gamma-1}{2}} \tag{3.36}$$

$$(\rho^s)_{i-\frac{1}{3}}^{\left(\frac{\gamma-1}{2}\right)} = \frac{(U_n^s)_{i-\frac{1}{3}}}{c_i}\rho_i^{\frac{\gamma-1}{2}} \tag{3.37}$$

$$(U_n^s)_{i-\frac{2}{3}} = \frac{\gamma-1}{\gamma+1}[(U_n)_{i-1} - \frac{2}{\gamma-1}c_{i-1}] \tag{3.38}$$

$$(U_n^s)_{i-\frac{1}{3}} = \frac{\gamma-1}{\gamma+1}[(U_n)_i + \frac{2}{\gamma-1}c_i] \tag{3.39}$$

$$(V_t^s)_{i-\frac{2}{3}} = (V_t)_{i-1} \tag{3.40}$$

$$(V_t^s)_{i-\frac{1}{3}} = (V_t)_i \tag{3.41}$$

$$(p^s)_{i-\frac{2}{3}} = p_{i-1}\left(\frac{(\rho^s)_{i-\frac{2}{3}}}{\rho_{i-1}}\right)^{\gamma} \tag{3.42}$$

$$(p^s)_{i-\frac{1}{3}} = p_i\left(\frac{(\rho^s)_{i-\frac{1}{3}}}{\rho_i}\right)^{\gamma} \tag{3.43}$$

Having determined the intermediate and the sonic points, the integration can be carried out using the same formula described in [16] [17].

3.4. Iteration algorithms

3.4.1. Explicit scheme

An explicit time stepping scheme results from an evaluation of the forms in equation (3.7) at time level n. Hence, the formulation using Roe's scheme will take the

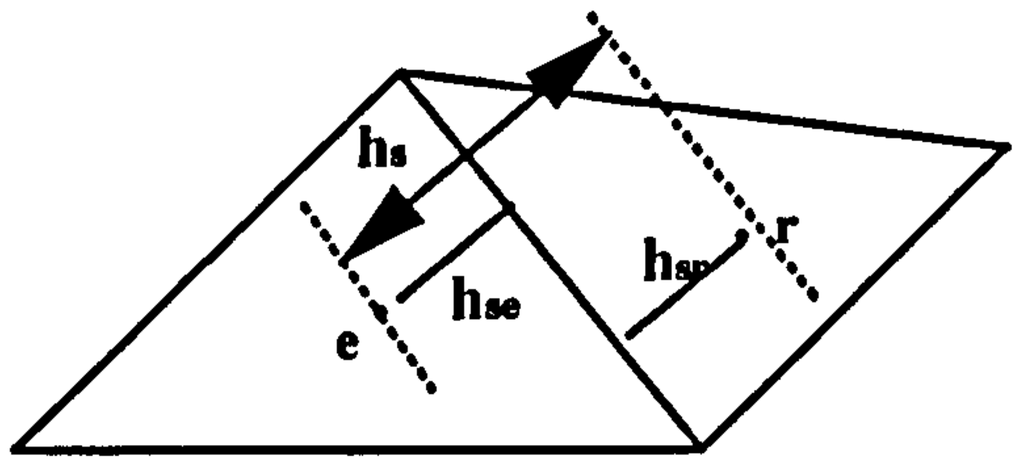


Fig. 3. The definition of hs

form

$$\Delta U_e = -\frac{\Delta t_e}{2\Omega_e}\sum_{s_e}[F_e^n + F_r^n - \mid A_{roe}^n \mid (U_r^n - U_e^n)]\,\delta s_e \tag{3.44}$$

While using Osher's scheme, the formulation will take the form

$$\Delta U_e = -\frac{\Delta t_e}{2\Omega_e}\sum_{s_e}\left[F_e^n + F_r^n - \int_{U_e^n}^{U_r^n} |A_{Osher}| dQ\right]\delta s_e \tag{3.45}$$

where the subscripts e and r denote the value at the left and right elements respectively and δs_e is the length of the side Γ_{es} of the interface. Both equations above can be iterated by the local time step to steady state [82].

The maximum allowable time step for a two-dimensional unstructured grid is defined according to the maximum eigenvalue (in the form of the absolute value) $(\lambda_{max})_s$ and the representative length h_s at its side "s_e" as

$$(\delta t)_s \leq \frac{h_s}{(\lambda_{max})_s} \tag{3.46}$$

where $(\lambda_{max})_s = |U_n| + c$ and $h_s = (h_s)_e + (h_s)_r$ are defined in figure 3.

The elemental local time step is then determined as the minimum of the time steps calculated at all its sides, i.e.

$$(\delta t)_e = min[(\delta t)_s] \tag{3.47}$$

where $s = 1, 3$ for triangular element and $s = 1, 4$ for quadrilateral element.

3.4.2. Point-implicit scheme

Firstly we consider the Roe's scheme. If the inviscid contributions are evaluated at time t^{n+1} , equation (3.7) will lead to the fully implicit scheme as

$$\Delta U_e = -\frac{\Delta t_e}{2\Omega_e}\sum_{s_e}[F_e^{n+1} + F_r^{n+1} - \mid A_{roe}^{n+1} \mid \left(U_r^{n+1} - U_e^{n+1}\right)]\delta s_e \tag{3.48}$$

Linearization of the equation for the values of the unknowns and fluxes at time level $(n+1)$ in the terms of the time level (n) result in

$$U_e^{n+1} = U_e^n + \Delta U_e \tag{3.49}$$

$$U_r^{n+1} = U_r^n + \Delta U_r \tag{3.50}$$

$$F_e^{n+1} = F_e^n + A_e^n \Delta U_e \tag{3.51}$$

$$F_r^{n+1} = F_r^n + A_r^n \Delta U_r \tag{3.52}$$

where matrix $A = \partial F/\partial U$ is the flux Jacobian matrix defined for a general direction $\vec{n} = (n_x, n_y)$ as

$$A = \begin{pmatrix} 0 & n_x & n_y & 0 \\ \alpha\beta n_x - uU_n & (2-\gamma)un_x + U_n & un_y - \beta v n_x & \beta n_x \\ \alpha\beta n_y - vU_n & vn_x - \beta u n_y & (2-\gamma)vn_y + U_n & \beta n_y \\ (2\alpha\beta - \gamma\varepsilon)U_n & \kappa n_x - \beta u U_n & \kappa n_y - \beta v U_n & \gamma U_n \end{pmatrix} \tag{3.53}$$

where

$$\alpha = \frac{u^2+v^2}{2}; \quad \beta = \gamma - 1; \quad \kappa = \gamma\varepsilon - (\gamma-1)\alpha,$$

Replacing the above expressions into equation (3.48), result in

$$\Delta U_e = RHS_{exp} - \frac{\Delta t_e}{2\Omega_e} \sum_{s_e} [(A_r^n - |A_{roe}^n|)\Delta U_r + |A_{roe}^n|\Delta U_e]\delta s_e \tag{3.54}$$

The first term RHS_{exp} on the right hand side of the above equation is equivalent to the right hand side of the explicit formulation given by equation (3.44).

This equation can be re-arranged as

$$\left[I + \frac{\Delta t_e}{2\Omega_e} \sum_{s_e} |A_{roe}^n| \delta s_e\right] \Delta U_e = RHS_{exp} - \frac{\Delta t_e}{2\Omega_e} \sum_{s_e} [(A_r^n - |A_{roe}^n|)\Delta U_r]\, \delta s_e \tag{3.55}$$

where I represents the unit matrix.

The above system of equations can be solved in each time step, using either a point Gauss- Jacobi procedure resulting in

$$\left[I + \frac{\Delta t_e}{2\Omega_e} \sum_{s_e} |A_{roe}^n| \delta s_e\right] \Delta U_e^{n+1} = RHS_{exp}^n - \frac{\Delta t_e}{2\Omega_e} \sum_{s_e} [(A_r^n - |A_{roe}^n|)\Delta U_r^n]\, \delta s_e \tag{3.56}$$

or a point Gauss-Seidel scheme which use the latest available value for the neigh-

bouring elements "r" written as

$$\left[I + \frac{\Delta t_e}{2\Omega_e} \sum_{s_e} |A^*_{roe}| \delta s_e\right] \Delta U_e^{n+1} = -\frac{\Delta t_e}{2\Omega_e} \sum_{s_e} \left[F_e^n + F_r^* - |A^*_{roe}| (U_r^* - U_e^n)\right] \delta s_e \quad (3.57)$$

where the terms denoted by an asterisk means the latest available values of the variables. In this case the linearization is only performed for the unknown variables and fluxes at the current element "e".

Secondly we consider the Osher's scheme. After following the same procedure as above, equation (3.24) will lead to the fully implicit scheme as

$$F^{n+1}(U_e, U_r) = \frac{1}{2}\left[F^{n+1}(U_e) + F^{n+1}(U_r) - \int_{U_e^{n+1}}^{U_r^{n+1}} |A_{Osher}| dQ\right] \quad (3.58)$$

Here we use the following approach to determine the left hand side of the implicit system of equations by replacing the numerical flux of Osher with a flux vector splitting scheme. Consider the flux vector splitting scheme of Steger and Warming [12], which can be expressed as

$$F(U_e, U_r) = F^+(U_e) + F^-(U_r) \quad (3.59)$$

For an implicit formulation, this equation can be linearized and written as

$$F^{n+1}(U_e, U_r) = F^n(U_e, U_r) + \left(\frac{\partial F^+(U_e)}{\partial U_e}\right)^n \Delta U_e + \left(\frac{\partial F^-(U_r)}{\partial U_r}\right)^n \Delta U_r \quad (3.60)$$

The Jacobian matrices in the above equation can be approximated by

$$\frac{\partial F^+(U_e)}{\partial U_e} = \left(\frac{\partial F(U_e)}{\partial U_e}\right)^+ = A_e^+ \quad (3.61)$$

$$\frac{\partial F^-(U_r)}{\partial U_r} = \left(\frac{\partial F(U_r)}{\partial U_r}\right)^- = A_r^- \quad (3.62)$$

Substituting these expressions into equation (3.60) results in the following linearization

$$F^{n+1}(U_e, U_r) = F^n(U_e, U_r) + A_e^+ \Delta U_e + A_r^- \Delta U_r \quad (3.63)$$

The Jacobian matrices in the above expressions are defined as

$$A^+ = R\Lambda^+ R^- \quad (3.64)$$

$$A^- = R\Lambda^- R^- \quad (3.65)$$

Where Λ^+ and Λ^- are the diagonal matrices of positive and negative eigenvalues

respectively, i.e

$$\lambda_i^+ = \max(0, \lambda_i) \tag{3.66}$$

$$\lambda_i^- = \min(0, \lambda_i) \tag{3.67}$$

The definition of R, R^- and Λ are the same as those in Roe's flux.

Now the term at time level n on the right hand side of the above equation (3.58) is replaced by the numerical flux of Osher in its explicit form. Hence the linearised implicit finite volume formulation will be given as

$$\Delta U_e = RHS_{exp} - \frac{\Delta t_e}{\Omega_e} \sum_{s_e} \left[A_e^+ \Delta U_e + A_r^- \Delta U_r \right] \delta s_e \tag{3.68}$$

where the term RHS_{exp} represents the right hand side of equation (3.45).

Taking all the terms depending on ΔU_e to the left hand side results in the following point Gauss-Jacobi iterative procedure as

$$\left[I + \frac{\Delta t_e}{\Omega_e} \sum_{s_e} A_e^+ \delta s_e \right] \Delta U_e^{n+1} = RHS_{exp}^n - \frac{\Delta t_e}{\Omega_e} \sum_{s_e} [A_r^- \Delta U_r^n] \delta s_e \tag{3.69}$$

Similiar to that of the numerical flux of Roe, an alternative point Gauss-Seidel formulation can be obtained by using the latest available values (denoted by asterisk) to determine the fluxes at the neighbouring elements. In this case equation (3.58) is written as

$$\left[I + \frac{\Delta t_e}{\Omega_e} \sum_{s_e} A_e^+ \delta s_e \right] \Delta U_e^{n+1} = -\frac{\Delta t_e}{2\Omega_e} \sum_{s_e} \left[F_e^n + F_r^* - \int_{U_e^n}^{U_r^*} |A_{Osher}| dQ \right] \delta s_e \tag{3.70}$$

3.5. Boundary conditions

All the boundary conditions used for the exterior boundary are based on the method of characteristics. For the wall boundary, both the method of characteristics and extrapolation from the interior flow field are used [83].

At the exterior boundary, we wish to minimize the reflection of outgoing disturbances. Consider the flow normal to this boundary. Assuming it to be locally one-dimensional, we introduce the fixed and extrapolated Riemann invariants according to 1-D Riemann relations

$$R_\infty = U_\infty \vec{n} - \frac{2c_\infty}{\gamma - 1} \tag{3.71}$$

$$R_e = U_e \vec{n} + \frac{2c_e}{\gamma - 1} \tag{3.72}$$

corresponding to incoming and outgoing characteristics. The normal velocity

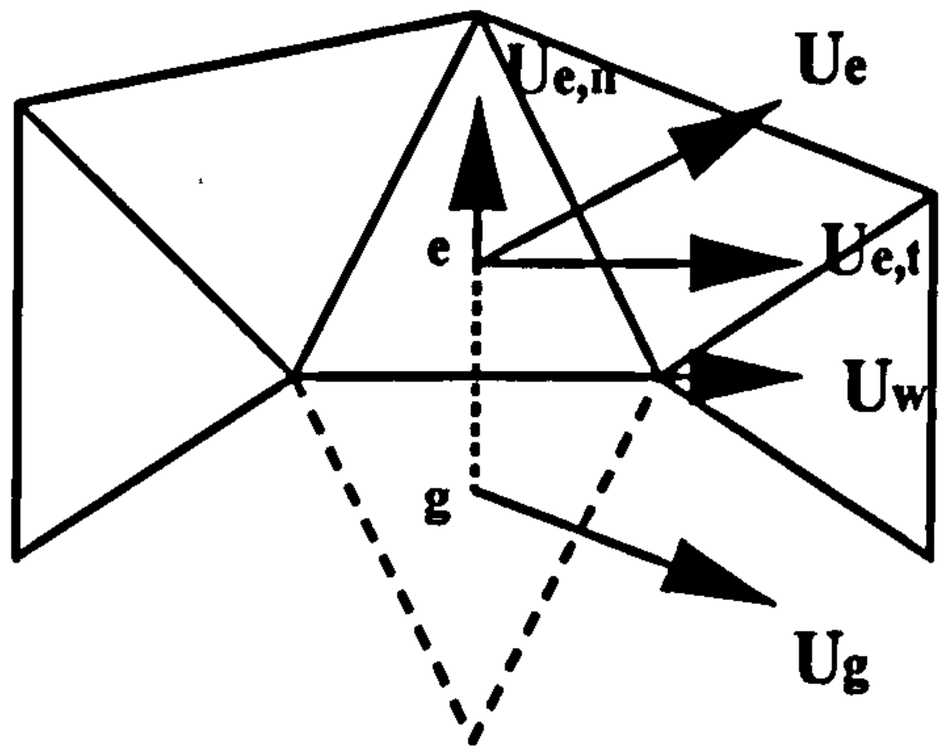


Fig. 4. The ghost element at the slip wall

and local speed of sound may thus be determined by

$$U\vec{n} = \frac{1}{2}(R_e + R_\infty) \tag{3.73}$$

$$c = \frac{\gamma - 1}{4}(R_e - R_\infty) \tag{3.74}$$

Two other independent conditions are needed to complete the definition of the outer boundary condition. These are given by the values of tangential velocity and entropy. For an outer flow boundary these are extrapolated from the interior values, whereas for an inflow boundary they are set equal to their freestream values.

At the inner boundary, e.g. a solid wall, the appropriate boundary conditions are the wall slip boundary condition, which means that the normal component of the velocity to the wall is zero. This can be implemented numerically in two ways described as follows.

3.5.1. Strong formulation

To specify the values of the unknowns, a set of ghost elements is introduced inside the wall boundary. The values for the velocity variables for these elements are set so that the average interface value satisfy the tangency condition. i.e. $U_n = 0$ (see figure 4). The values of the other two parameters (density and pressure) are taken to be the same as the values inside the domain. They are

$$\rho_g = \rho_e \tag{3.75}$$

$$u_g = -(U_e)_n n_x - (V_e)_t n_y \tag{3.76}$$

$$v_g = -(U_e)_n n_y + (V_e)_t n_x \tag{3.77}$$

$$p_g = p_e \tag{3.78}$$

3.5.2. Weak formulation

Using the velocity tangency condition in equation (3.8) of F_n, i.e. $U_n = 0$, the fluxes at the wall are obtained in the following expression

$$F_w = \begin{pmatrix} 0 \\ p_w n_x \\ p_w n_y \\ 0 \end{pmatrix} \tag{3.79}$$

It is necessary to determine the pressure at the wall. This also means only the pressure contribution remains at the walls.

Various methods can be applied in order to obtain the wall pressure. The ones mostly used are characteristic relations based on the Riemann invariant and extrapolations.

For the explicit scheme, the numerical fluxes of equation (3.79) can be imposed directly at the sides on the wall.

For the implicit scheme, however, further care is needed. The Jacobian matrix of the transformation $U_e \rightarrow F_w$ must also be used on the left hand side of the implicit equation system. It is

$$A_w = (\gamma - 1) \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{u^2+v^2}{2} n_x & -u n_x & -v n_x & n_x \\ \frac{u^2+v^2}{2} n_y & -u n_y & -v n_y & n_y \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.80}$$

As an example, the implicit formulation for an element adjacent to the wall using the numerical flux of Roe is explained. Equation (3.48) is now written as

$$\Delta U_e = -\frac{\Delta t_e}{\Omega_e} \left\{ \sum_{s_e \neq w} \frac{1}{2} \left[F_e^{n+1} + F_r^{n+1} - |A_{Roe}^{n+1}| (U_r^{n+1} - U_e^{n+1}) \right] \delta s_e + F_w^{n+1} \delta w \right\} \tag{3.81}$$

Using the linearizations

$$F_w^{n+1} = F_w^n + A_w^n \Delta U_e \tag{3.82}$$

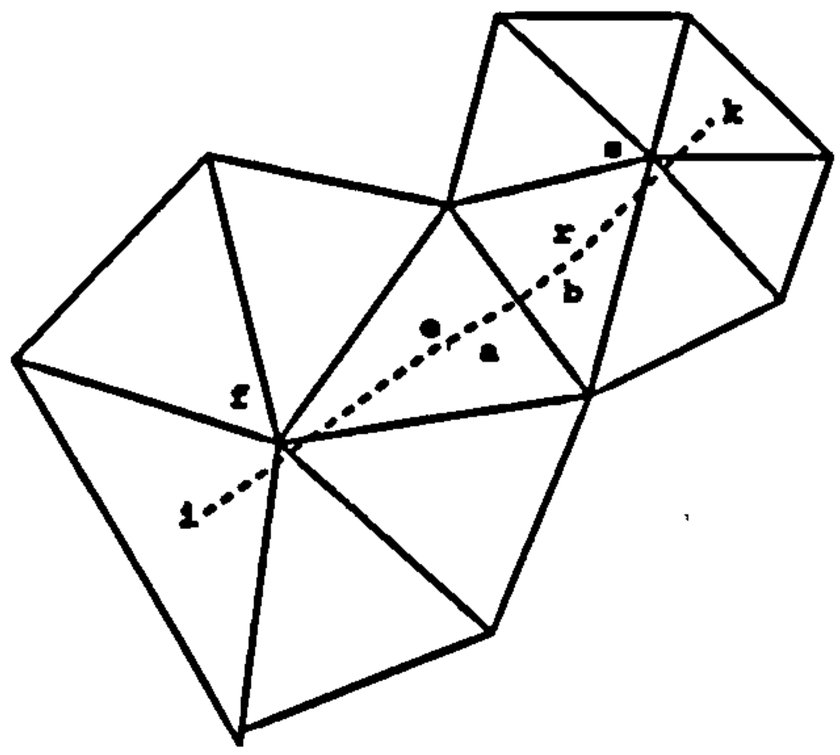


Fig. 5. The diagram of MUSCL approach

it can be re-written as (in the form of the point Gauss-Seidel method)

$$\begin{aligned}\Delta U_e &= -\frac{\Delta t_e}{\Omega_e}\{\sum_{s_e \neq w}\frac{1}{2}[F_e^n + F_r^* - |A_{Roe}^*|(U_r^* - U_e^n) + A_e^n \Delta U_e \\ &\quad + |A_{Roe}^*|\Delta U_e]\delta s_e + (F_w^n + A_w^n \Delta U_e)\delta w\} \end{aligned} \tag{3.83}$$

The above formula, upon taking terms involving ΔU_e, can be written as

$$A\Delta U_e = b \tag{3.84}$$

where

$$\begin{aligned} A &= I + \frac{\Delta t_e}{\Omega_e}\sum_{s_e \neq w}[(A_e^n + |A_{roe}^*|)\delta s_e + 2A_w^n \delta w \\ b &= -\frac{\Delta t_e}{\Omega_e}\sum_{s_e \neq w}\{\frac{1}{2}[F_e^n + F_r^* - |A_{roe}^*|(U_r^* - U_e^n)\delta s_e + F_w^n \delta w\} \end{aligned}$$

The implicit formulation for an element adjacent to wall using the numerical flux of Osher is carried out in a similar way.

3.6. High order resolutions based on MUSCL approach

High order resolution has been achieved by the use of the variable extrapolation method reported in [29] [30] [31]. Similar to that carried out on a structured grid, the variable extrapolation, i.e. MUSCL approach, is also used to determine the variables at both sides of the interface to calculate the Riemann solver.

For two given neighbouring elements "e" and "r" for example and considering the figure 5, a kappa-parameter family of high order schemes corresponding to side

"s" can be written as

$$q_s^+ = q_e + \left\{ \frac{s_1}{4}[(1-\kappa s_1)\Delta_- + (1+\kappa s_1)\Delta_+]q \right\}_e \tag{3.85}$$

where $\Delta_+ = q_r - q_e$, $\Delta_- = q_e - q_i$ and

$$q_s^- = q_r - \left\{ \frac{s_2}{4}[(1+\kappa s_2)\Delta_- + (1-\kappa s_2)\Delta_+]q \right\}_r \tag{3.86}$$

where $\Delta_+ = q_k - q_r$ and $\Delta_- = q_r - q_e$

In the above equations, q_e and q_r are the vectors of primitive variables at the centroids of elements "e" and "r" respectively and q_i, q_k are the vectors of primitive variables at the elements which are nearest to the line connecting two centroid point (e,r). The advantage of this method is that no approximation need be introduced but on the other hand searching and storing of the address of i,k is needed. Here we use another simple approach. Instead of using the value at elements "i" and "k", the values at nodes "f","s" are used to determine the extrapolation in equations (3.85) and (3.86). The node value at "f" and "s" are calculated by the weighted average of the flow variables in all the surrounding elements. That is

$$q_{node} = \sum_{j=1}^{Node} q_j \omega_j \tag{3.87}$$

where ω_j is the weighted coefficient.

By considering this change in equation (3.85) the item Δ_- becomes $q_e - q_f$ instead of $q_e - q_i$. Also the Δ_+ becomes $q_s - q_r$ instead of $q_k - q_r$ in equation (3.86).

The parameter κ controls a family of difference schemes by appropriately weighting Δ_+ and Δ_- . On structured meshes, it is easy to show that $\kappa = -1$ corresponds to a full upwind second order scheme, $\kappa = 0$ yields Fromm's scheme and $\kappa = 1$ yields a central difference scheme. The value $\kappa = 1/3$ leads to a third order accurate upwind-biased scheme.

The parameter s_1, s_2 serves to limit high order terms in the extrapolation in order to avoid oscillations in the solutions at discontinuities such as shock waves. According to van Leer et al [84], the limiting is implemented by locally modifying the difference values in the extrapolation to ensure monotone extrapolation as

$$s_1 = \frac{2\Delta_+ q \Delta_- q + \delta}{(\Delta_+ q)^2 + (\Delta_- q)^2 + \delta} \tag{3.88}$$

$$s_2 = \frac{2\Delta_+ q \Delta_- q + \delta}{(\Delta_+ q)^2 + (\Delta_- q)^2 + \delta} \tag{3.89}$$

where δ is a small number preventing division by zero in regions of null gradients.

On highly stretched meshes, the formula for Δ_+ is modified to be

$$\Delta_+ = \frac{2a}{a+b}(q_r - q_e) \tag{3.90}$$

and Δ_- is also modified to be

$$\Delta_- = \frac{2b}{a+b}(q_r - q_e) \tag{3.91}$$

where a and b are the distances from the midpoint of an edge to the centroid of elements "e" and "r", respectively, as shown in figure 5.

This formula weights differentially the flow variables in the extrapolation formula, to account for the stretching of the mesh. For example, by substituting equation (3.90) into equation (3.85) and letting $\kappa = 0, s_1 = 1$ yields

$$q_s^+ = \frac{b}{a+b}q_e + \frac{a}{a+b}q_r \tag{3.92}$$

For the case shown in figure 5, this means more weight in the calculation of q_s^+ of the flow variables at the centroid of element "e" than to the flow variables at the centroid of element "r", since $b > a$.

3.7. Unstructured grid generator and adaptive remeshing procedure

The algorithm for the grid generation is the advancing front technique proposed by Peraire et al [50] which consists the following steps. Firstly the generation process is started by constructing a grid called the background grid which completely covers the solution domain of interest and contains the definition of element size, stretch direction etc. Secondly the boundary curve of the domain is discretised into a set of segments, and boundary nodes are placed at the points of intersection of these segments. The segments of the exterior boundary are defined in an anti-clockwise fashion while the segments of the interior boundaries are specified in a clockwise manner. This means that the region to be triangulated always lies to the left hand side as the boundary curve is traversed. The third step is triangle generation. At the start of the process the initial front consists of the sequence of straight line segments which connect consecutive boundary nodes. The length of these segments must therefore, be consistent with the desired local distribution of grid size. This operation is repeated for each boundary curve in turn. The front is a dynamic data structure which changes continuously. During the generation process a straight line segment which is available to form an element side is termed active, whereas any segment which is no longer active is removed from the front. Thus while the domain

boundary will always remain the same, the generation front will change continuously and has to be updated whenever a new triangle is formed. The generation process ceases when the number of active sides in the front list is empty. The size and shape of the generated triangles must be consistent with the local desired size and shape of the final grid.

It is known that mesh adaption is also an important procedure in numerical flow simulation. It offers the prospect of accurate flow field simulations without the use of excessively fine, computationally expensive meshes. In this case small elements are used only in the regions where the flow is complex whereas large elements are used in the rest of the domain. A general scheme using this strategy requires three steps. Firstly, an unstructured mesh generator, able to control the sizes of the elements everywhere, is needed. Secondly, the flow solver is used to calculate physical flow variables. Thirdly, an a posteriori error estimator or indicator is required, which reveals where in the flow field the mesh is deficient and requires some smaller elements in these region. Of the many methods available for mesh adaptation [85] the remeshing procedure is a fairly easy one to implement on unstructured grids and has now been coupled with the inviscid Euler flow solver.

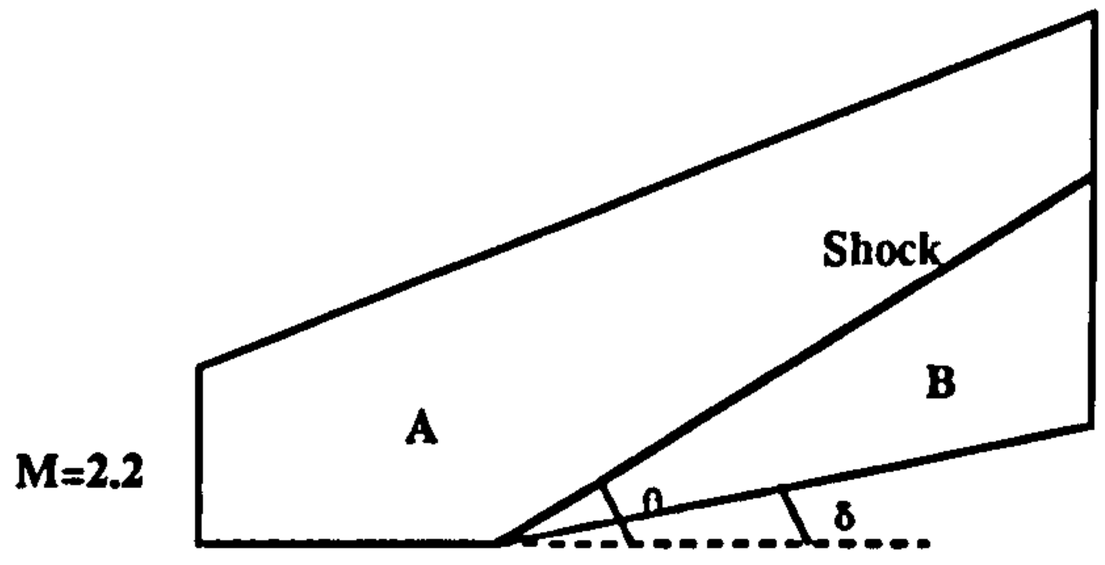
The basic idea of the adaptive remeshing procedure is to use the computed solution on the initial mesh to provide the information on the flow, in particular the region with high flow gradient values. This can be done exactly by calculating the gradient value across an edge, for example

$$\delta = \frac{\rho_e - \rho_r}{\rho_e + \rho_r} \tag{3.93}$$

if the density ρ has been chosen as the indicator, or by another way through contour line drawing. Clustering of contour line identifies a high gradient region. Either of the methods above can be used to locate the position of the high gradient region and the remeshing procedure is then carried out by changing the mesh size parameters.

3.8. Results and discussions

To validate the present inviscid Euler codes, calculations were performed on five typical inviscid flow test cases. They are supersonic flow over a compression corner, hypersonic flow over a cylinder and a blunt body, subsonic flow over a NACA 0012 airfoil and transonic flow over a NACA 0012 and a RAE 2822 airfoil. The MUSCL approach is used in the computation of supersonic and hypersonic flows. The adaptive remeshing procedure is considered in all of the numerical test cases.



δ =16 (deg) angle of flow deflection across an oblique shock-wave
θ =42.5 (deg) shock-wave angle measured from upstream flow direction

Region A	Region B
M=2.2	M=1.58
P=0.1475	P=0.3554
ρ=1.0	ρ=1.8382

Fig. 6. Definition of the supersonic flow past a compression corner

CPU time per ite (seconds)	Code					
	Ex-Roe	PGS-Roe	PGJ-Roe	Ex-Osher	PGS-Osher	PGJ-Osher
1st-order,sp	0.0872	0.1385	0.1497	0.1187	0.2589	0.2638

Table I. Comparison of the CPU time on coarse mesh

3.8.1. Supersonic flow over a compression corner

The definitions of the corner flow can be found in figure 6. The deflection angle is 16°. The analytical solution to this problem can be obtained from elementary gas dynamics. The solution consists of two different regions of constant states which are separated by an oblique shock wave as is sketched in figure 6. It can be seen that the flow remains supersonic behind the shock wave. Therefore the flowfield is supersonic throughout the domain.

The initial mesh used is the coarse one which contains only 213 elements and 131 nodes in the flow domain (figure 9). Six codes (i.e. Explicit-Roe, PGS-Roe, PGJ-Roe, Explicit-Osher, PGS-Osher and PGJ-Osher) are validated on this mesh and results are compared. Figure 7 presents two pictures of the comparison of convergence history by using Roe's and Osher's schemes. It is found that the implicit method converges nearly twice as fast as the explicit one. Table I lists the CPU time consumed per iteration by each code. The calculations are performed on a SG Indy workstation.

It is also found that the definition of precision affects the value of residual reached.

CPU time per ite (seconds)	Code		
	Ex-Roe	PGS-Roe	PGJ-Roe
1st-order,sp	0.95	1.32	1.76
1st-order,dp	1.087	1.7784	—
2nd-order,dp	3.377	4.074	—

Table II. Comparison of the CPU time on fine mesh

By using single precision (sp) the value of residual will reduce to 0.5E-4 – 1.E-5 and then stall, while by using double precision (dp) the residual can go down further to nearly 1.E-14.

Figure 9 shows the initial mesh and the corresponding pressure and Mach number contours obtained by the PGS-Roe code with the first order in Riemann solver.

Application of the remeshing procedure is performed based on the results of the initial mesh. The numerical computation has been repeated on the refined meshes. The number of elements now increases to 2390 (figure 9). Most of the re-generated elements are concentrated in the shock wave region, located by the computational results on the initial mesh. Figure 8 gives the convergence history of the Explicit, PGS and PGJ codes using Roe's scheme. It can be seen that the PGS-Roe code converges faster than the Explicit-Roe code and the PGJ-Roe code. The comparison of the effect of single and double precision is illustrated with the PGS-Roe code. Table II lists the CPU time per iteration of different codes used on the fine mesh on the SG Indy computers.

Figure 9 shows the fine mesh and the corresponding pressure and Mach number contours obtained by the PGS-Roe code with the first order approach. It is clear that the results on the fine mesh improved considerally more than those on the coarse mesh. However the shock wave is seen to be little resolved because of the first order approach used. To improve the capture of the shock wave with more accuracy the linear-resolution scheme using MUSCL approach is adopted. Figure 10 illustrates the comparison of convergence history between the first and high order schemes. It can be seen that the reconstructed high order scheme takes more iterations to reach the same convergence criteria. Figure 11 provides the results of pressure and Mach number contours using the high order scheme. It is clear that improved shock wave band is reached with comparison of those in figure 9.

3.8.2. Hypersonic flow

(a) Hypersonic flow over a cylinder

The test case is a hypersonic flow over a sector of a cylinder with a free stream Mach number of 8.0 with zero incidence. The computational results on a coarse mesh show there exists an arc-type shock wave through high gradient value in the flow domain. After remeshing, a fine mesh with 2985 elements and 1560 nodes is generated. Smaller elements are located mainly in the detected shock wave region. Figure 12 shows the unstructured mesh, the flow vector field and the contours of pressure, density and Mach number. Accurate simulation is reached using the PGS-Roe code with the MUSCL approaches.

(b) Hypersonic flow over a blunt body

The case is the hypersonic flow over a blunt body with a free stream Mach number of 10.0 and a zero angle of attack. Following the same procedure as that in the computation of the cylinder , a fine mesh, which contains 2697 elements and 1400 nodes, is obtained based on the results of the coarse mesh calculation. From figure 13 it can be seen that more elements are concentrated in the shock wave region. Also figure 13 illustrates the flow vector field and the contours of pressure, density and Mach number. The MUSCL approach together with the PGS-Roe scheme are used in this simulation.

3.8.3. Subsonic flow

The behaviour of the point Gauss-Seidel implicit scheme is examined by considering the case of a Mach number of 0.63 inviscid flow past a NACA 0012 airfoil with an incidence of 2^o. The flow is subsonic everywhere. An unstructured grid consisting of 6354 elements and 3267 nodes is employed. The far field boundary is placed at 10 chords lengths away. Figure 14 gives the details of the mesh and the results of the pressure distribution along the airfoil surface. Figure 15 shows the pressure and Mach number contours of the flow field. It should be noted that the computation here is based on the first order scheme.

3.8.4. Transonic flow

(a) Transonic flow over NACA 0012 airfoil

The first example considered here is the transonic flow over a NACA 0012 airfoil.

Two test cases are considered. They are the transonic flow with the flow conditions of (i) a freestream Mach number of 0.75 and the incidence of 2^o and (ii) a freestream Mach number of 0.80 and the incidence of 1.25^o.

The initial mesh is shown in figure 16, which consists of 3246 elements and 1670 nodes. The correspondent pressure and Mach number contours on an initial mesh are shown in figure 16. The second mesh is shown with 6354 elements and 3267 nodes, with more elements deployed around the surface of the airfoil especially in the leading edge regions and it shows the improvement in pressure and Mach number contour definition. Figure 17 gives the meshes and results of contours of test case (ii).

From the initial result of test case (i) it is found that there exists a high gradient region, i.e. shock wave on the upper surface about 40% chord. After the process of remeshing, smaller elements have been included in that region to simulate the shock wave with more accuracy. Figure 18 gives the result of the final generated mesh which contains 8558 elements and 4381 nodes and the pressure and Mach number contours calculated on that mesh with the PGS-Roe code. The result is obviously improved especially in the shock region.

The same procedure is followed on test case (ii). Figure 19 provides the remeshed mesh with 9769 elements and 4988 nodes. More smaller elements are clustered in the vicinity of the 60% chord of the upper surface and 34% chord of the lower surface where high-gradient values are detected. Also the pressure and Mach number contours illustrated in this figure clearly show the improved result reached.

Figure 20 gives the comparison of the C_p distributions on different meshes for test case (i) (upper) and case(ii) (lower). The results are obtained by using the PGS-Roe code with first order schemes.

(b) Transonic flow over RAE 2822 airfoil

The second example considered here is the transonic flow over a RAE 2822 airfoil with a freestream Mach number of 0.75 and incidence of 3^o . The initial mesh is shown in Figure 21. This mesh consists of 6674 elements and 3426 nodes. The pressure and Mach number contours for this case are shown in the same figure. Following the adaptive remeshing procedure, a high-gradient region can be detected. Smaller elements are again used in that region to capture the smaller changes in it. The resulting fine grid, which now contains 9506 elements and 4850 nodes, together with the pressure and Mach number contours are shown in figure 21. The results are obviously improved especially in the shock region.

Figure 22 gives the comparison of the C_p distributions on two meshes. The results are obtained by the first order approach using the PGS-Roe code.

3.9. Conclusions

Explicit and point implicit (Gauss-Seidel and Gauss-Jacobi) iteration schemes with the upwind cell-centred finite volume method on an unstructured grid have been

proposed and tested. For the inviscid Euler flow, the code with the point implicit scheme achieves more efficiency than that with the explicit approach. Validation was carried out on different geometries such as corner, cylinder, blunt body and airfoils and over a wide range of Mach numbers from subsonic to hypersonic flow. The incorporation of mesh adaptivity, i.e. using the adaptive remeshing strategy, substantially improves the quality of the flow simulation.

Through calculation the following conclusions are made:

(1) The convergence history improves quantitative when using a point implicit scheme instead of an explicit one. Although the implicit code takes a little more CPU time per iteration, the total improvement in efficiency is notable. Hence the implicit scheme is suggested to be used.

(2) For single precision the residual does not decrease below 1.E-5 in the corner flow case, while for double precision it reaches 1.E-14. It is suggested that double precision be always used in the calculations.

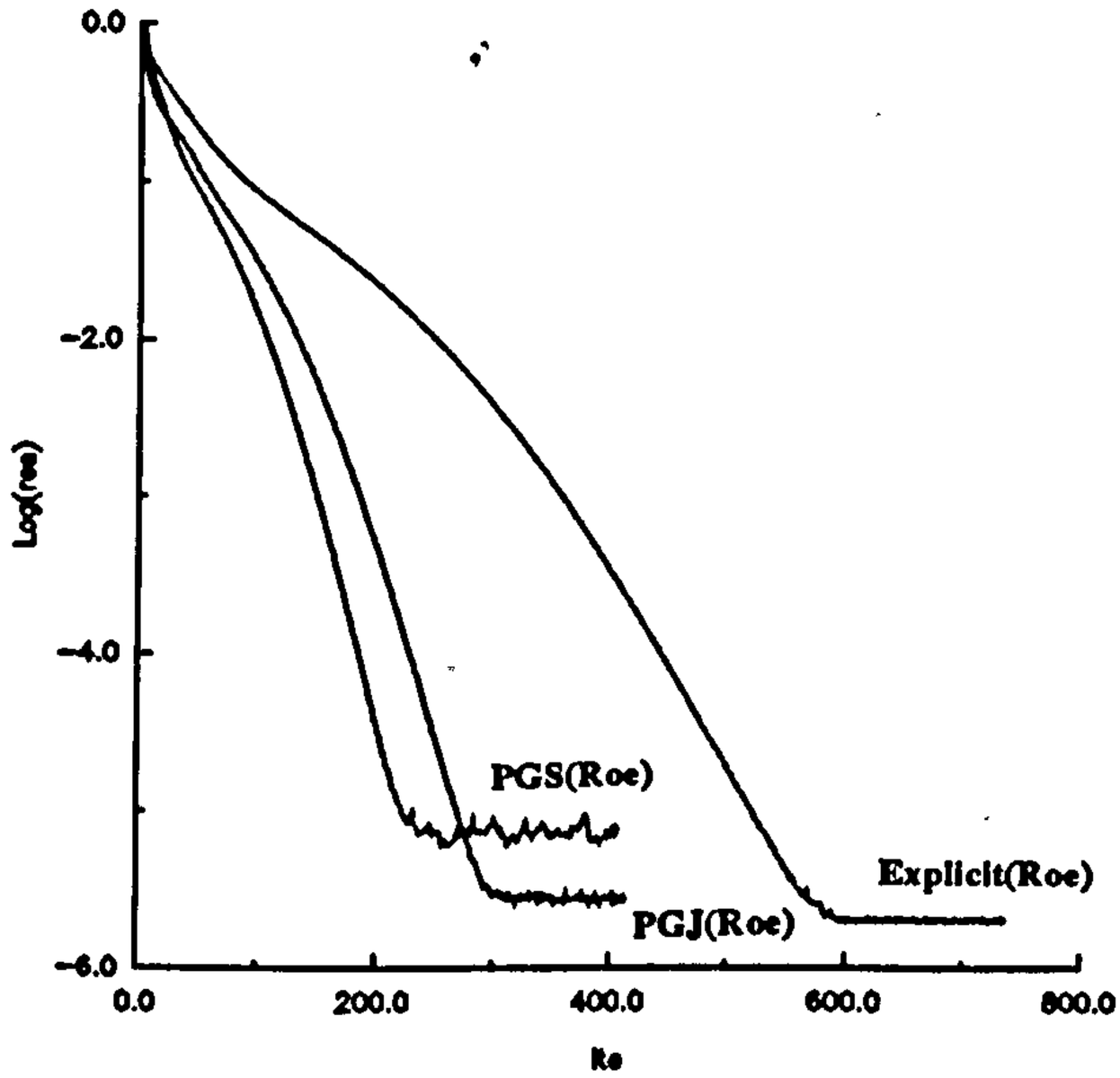
(3) From the test case of corner flow both Roe's and Osher's schemes give nearly the same satisfactory results. For the implicit scheme we prefer to use Roe's scheme rather than Osher's scheme as it takes less CPU time.

(4) Both the PGS and PGJ methods give similar convergence rates on coarse mesh. But on fine mesh the convergence performance of the PGS method is observed to be better than that of the PGJ method.

(5) The PGS-Roe code together with the MUSCL approach gives more accurate results than that with the first order approach for corner, cylinder and blunt body flows.

(6) Even using the first order scheme, good results are achieved in the calculation of the subsonic/transonic airfoil using the process of the adaptive remeshing procedure in capturing the shock wave region.

The experience with above method appears quite promising. However the problem still remains of how to implement the high order scheme to improve the accuracy on airfoil cases. This topic will be discussed in the next chapter.



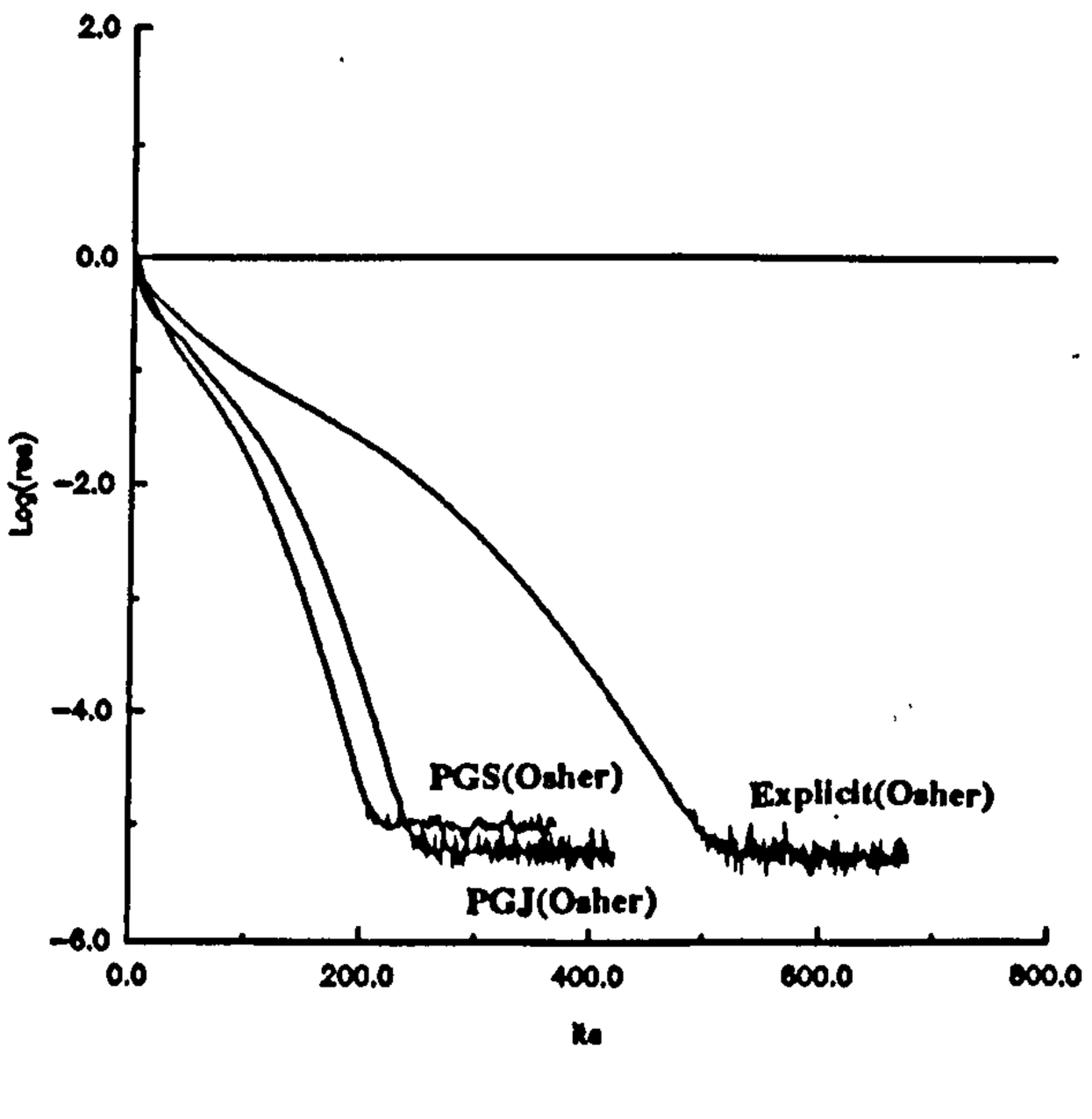
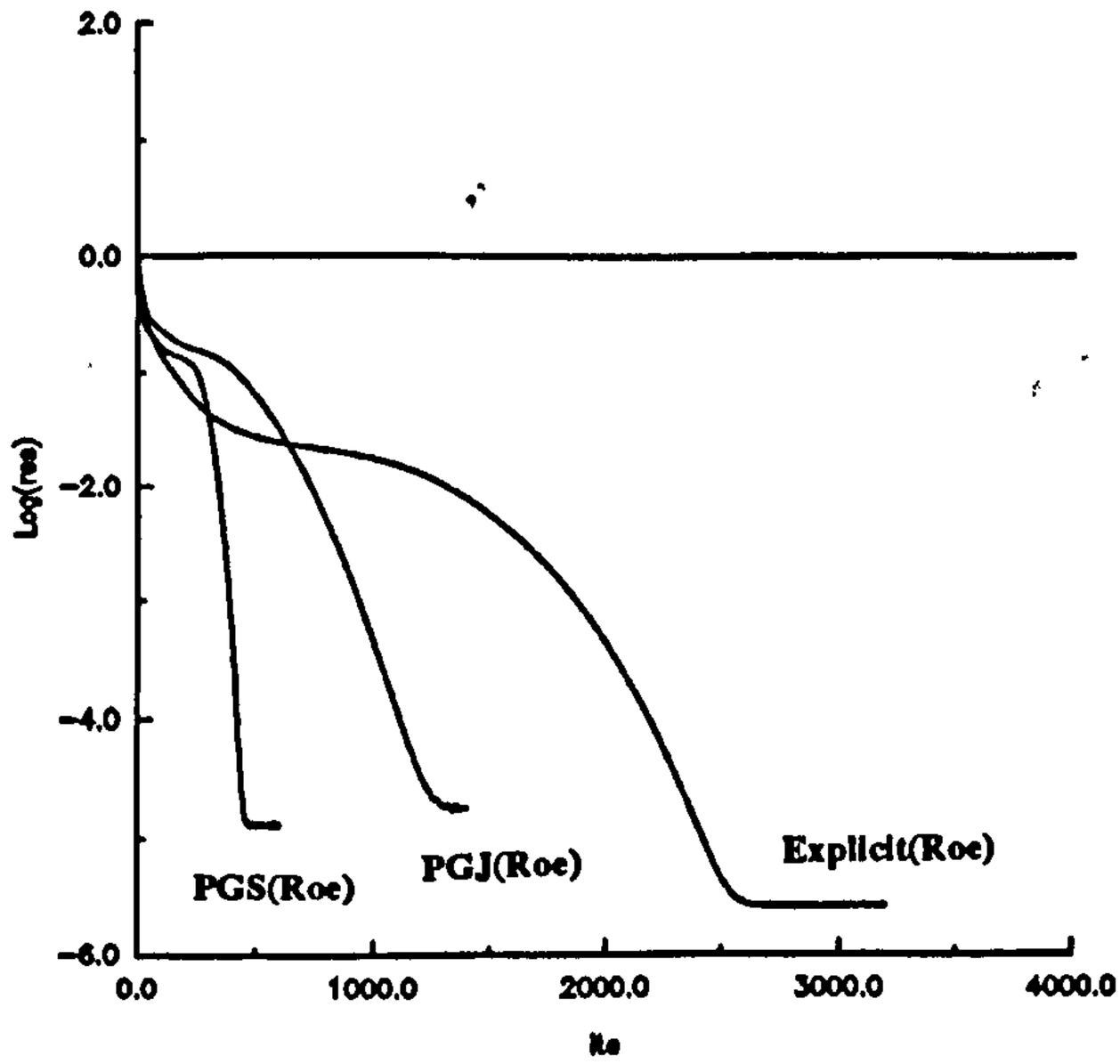


Fig. 7. Convergence history of the supersonic compression corner flow on initial mesh



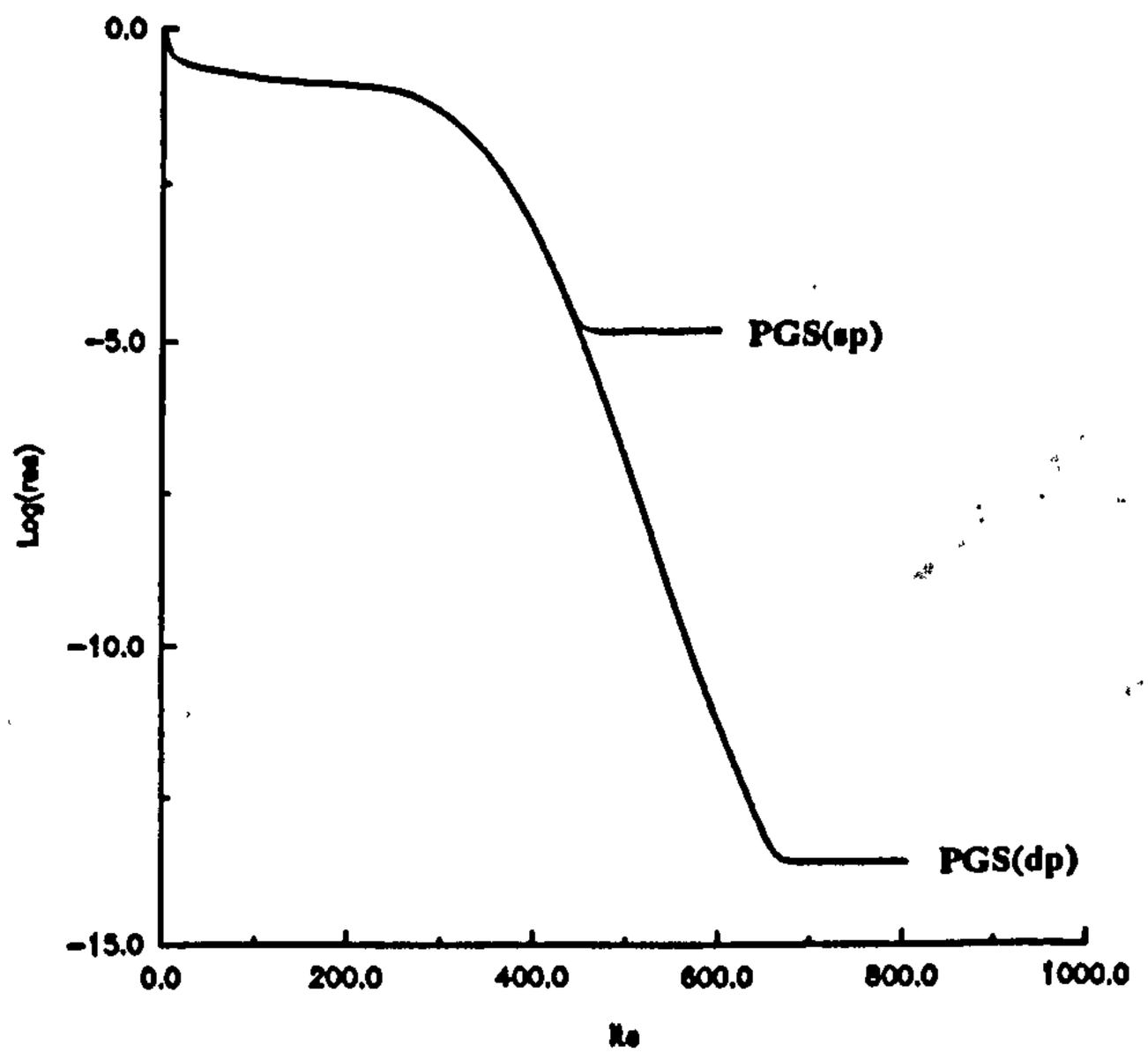
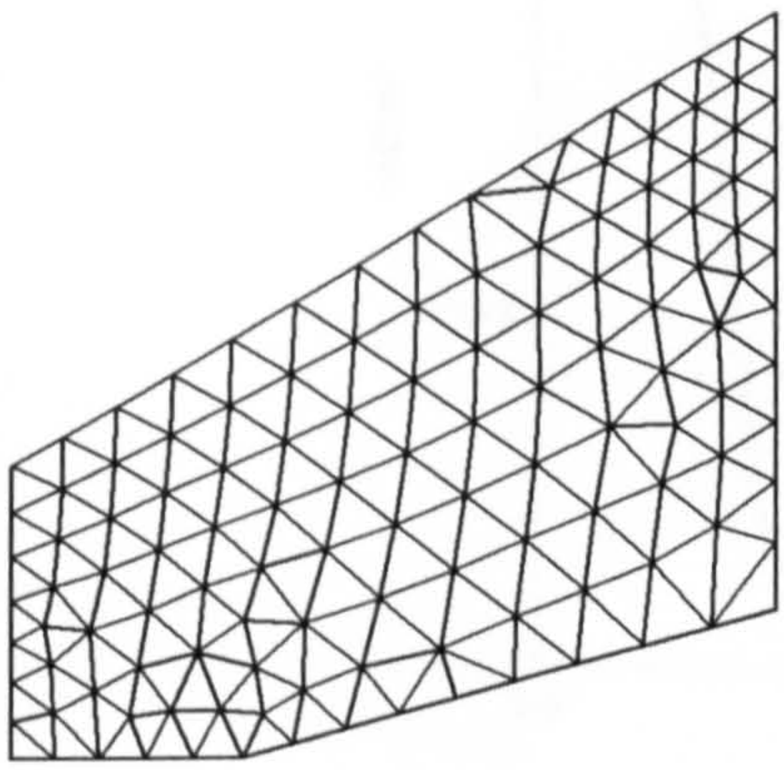
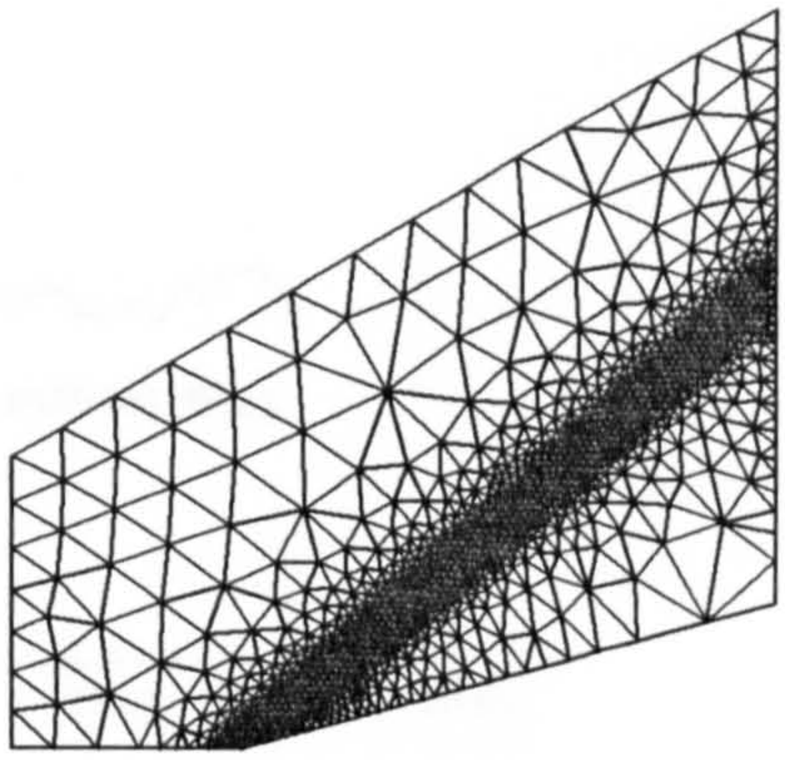


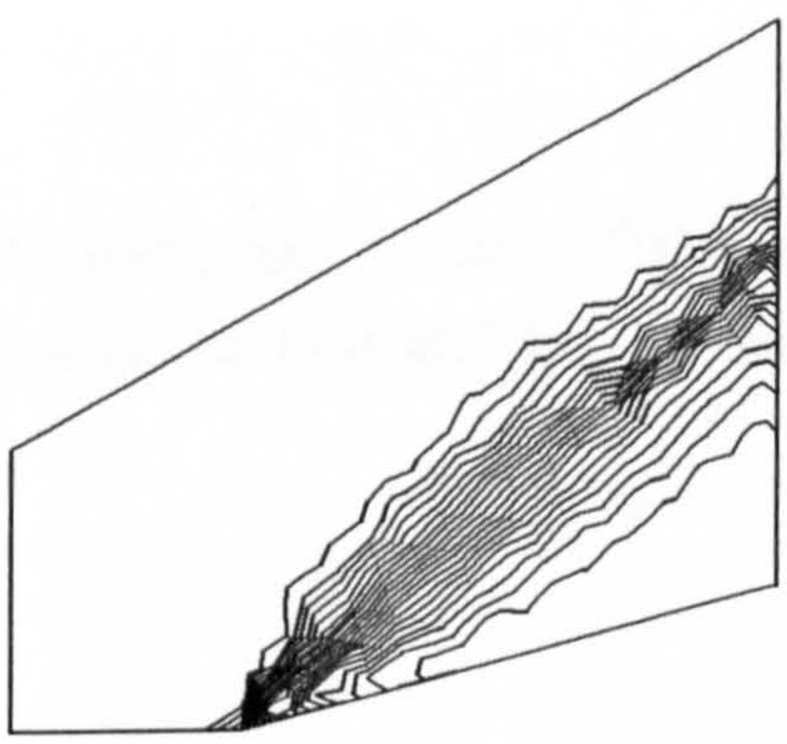
Fig. 8. Convergence history of the supersonic compression corner flow on fine mesh



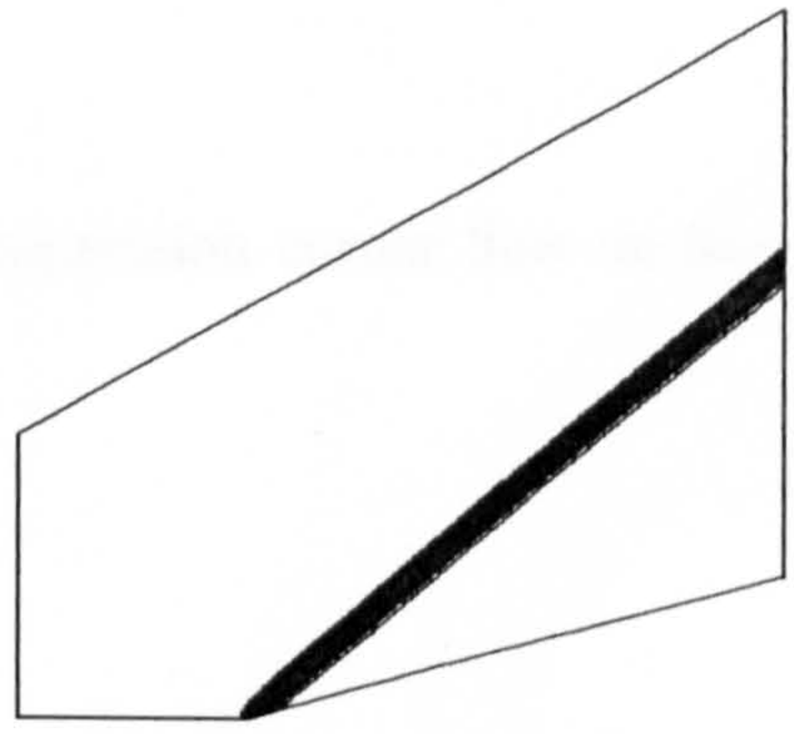
Coarse mesh



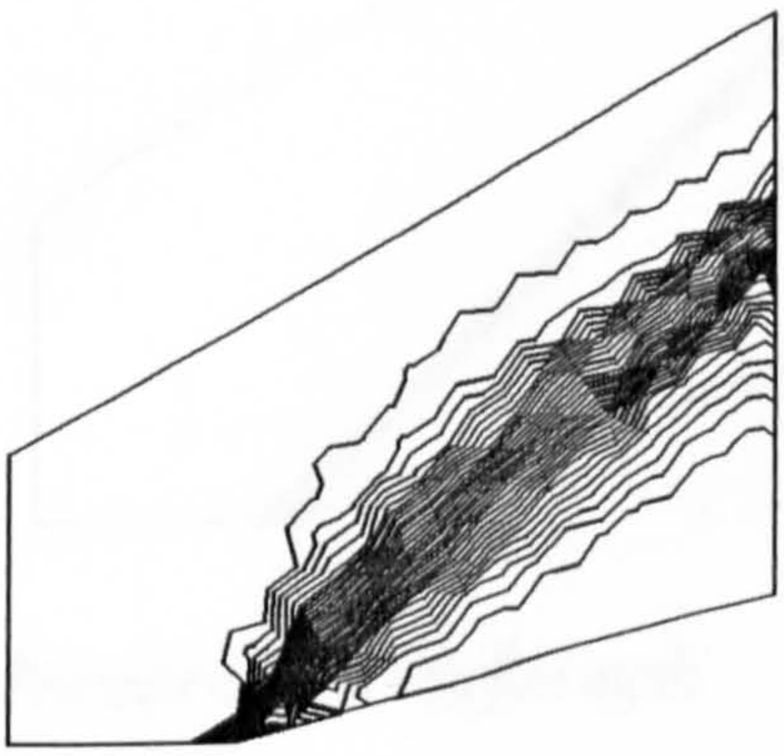
Fine mesh



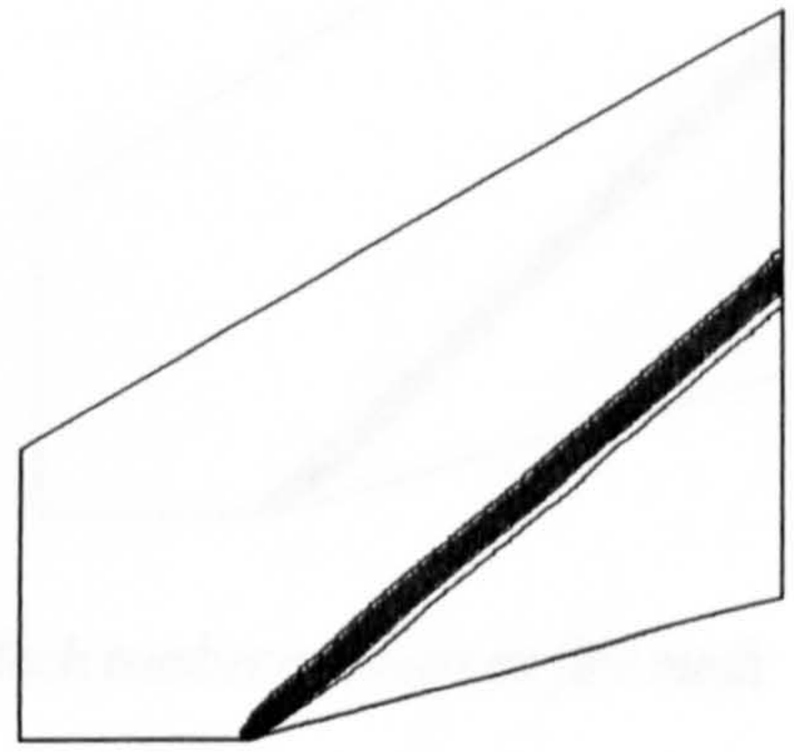
Pressure contours on coarse mesh



Pressure contours on fine mesh



Mach number contours on coarse mesh



Mach number contours on fine mesh

Fig. 9. Meshes, pressure and Mach number contours of the supersonic compression corner flow by PGS-Roe code with the first order scheme

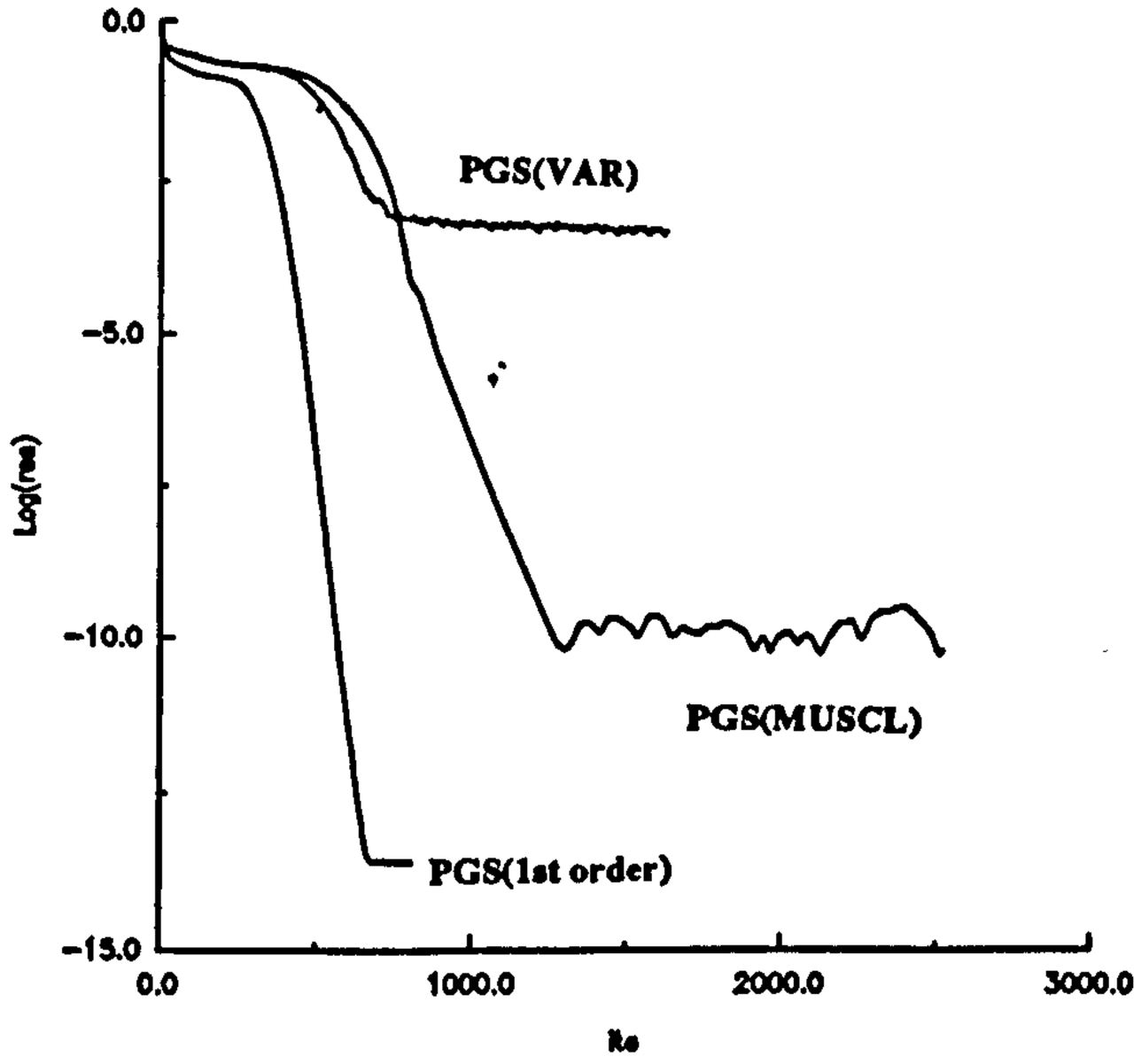
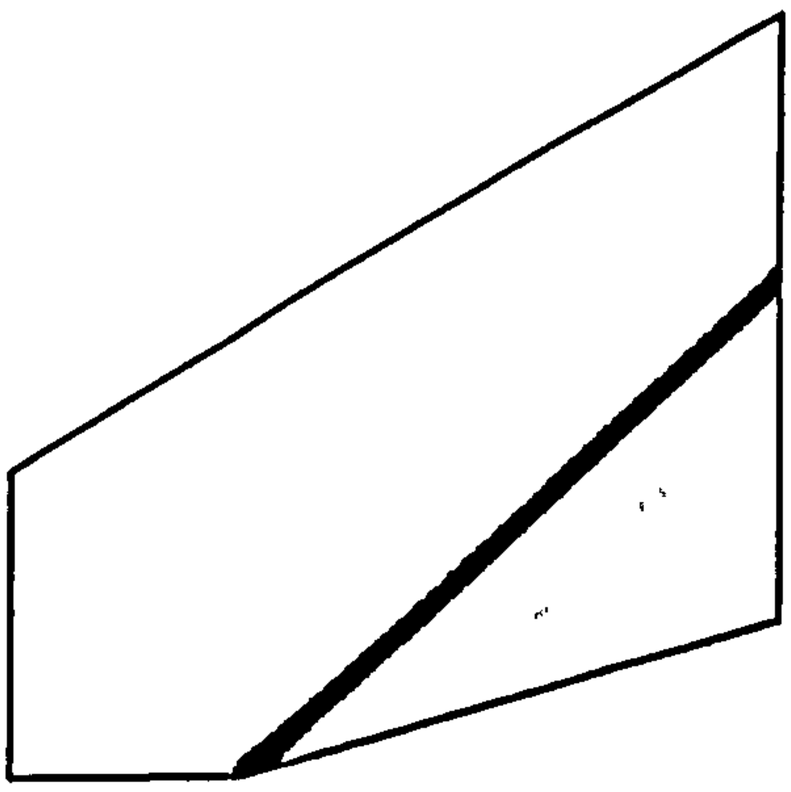
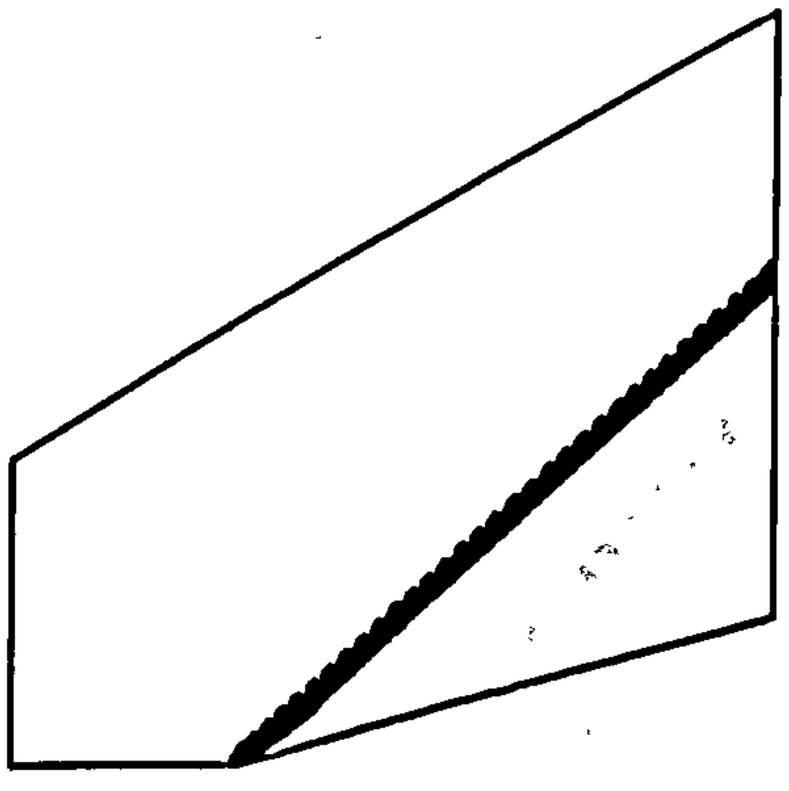


Fig. 10. Convergence history of the supersonic compression corner flow on fine mesh using the first and high order schemes

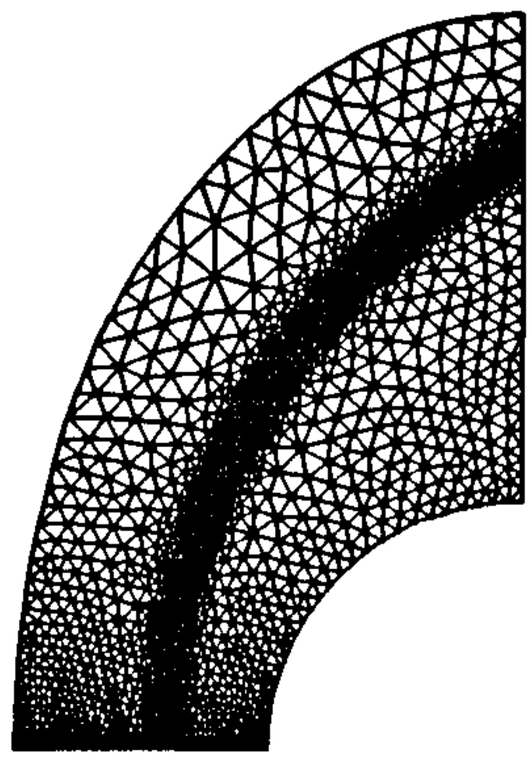


Pressure contours on fine mesh

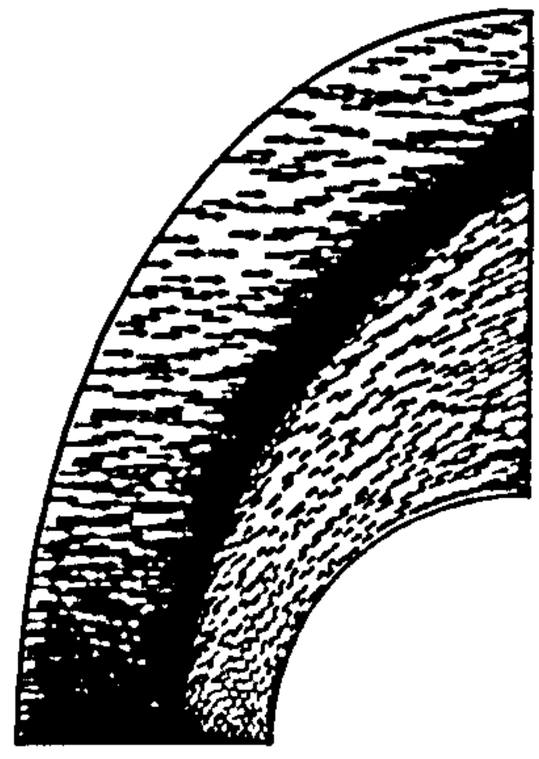


Mach number contours on fine mesh

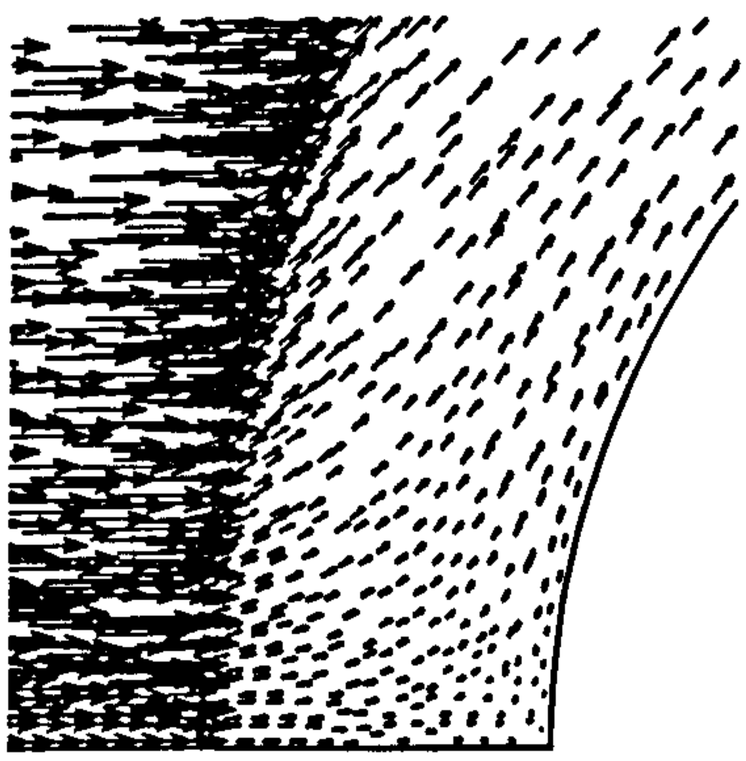
Fig. 11. Pressure and Mach number contours of the supersonic compression corner flow by the PGS-Roe code with MUSCL approach on fine mesh



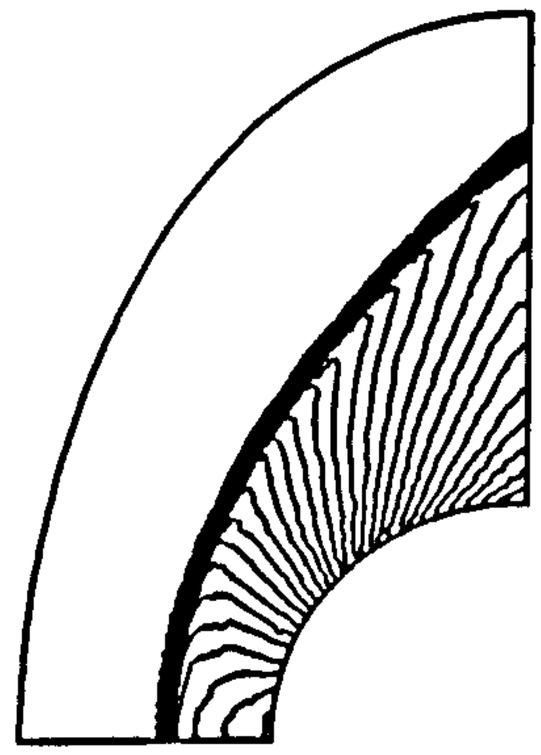
Mesh



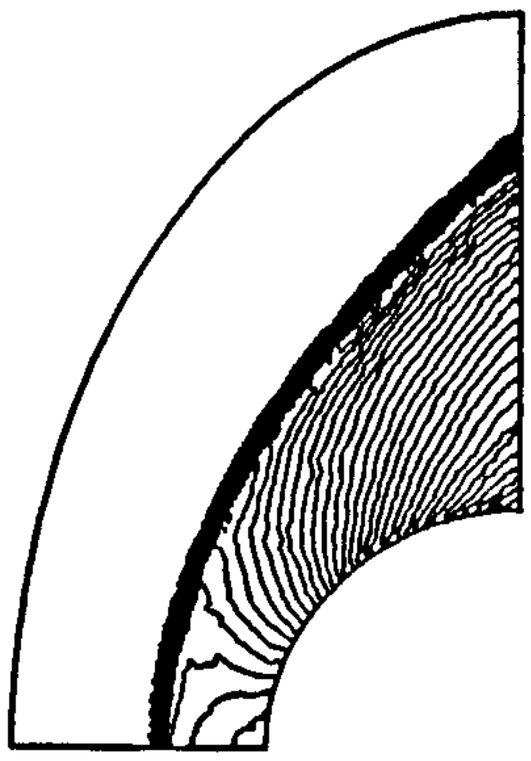
Velocity flow field



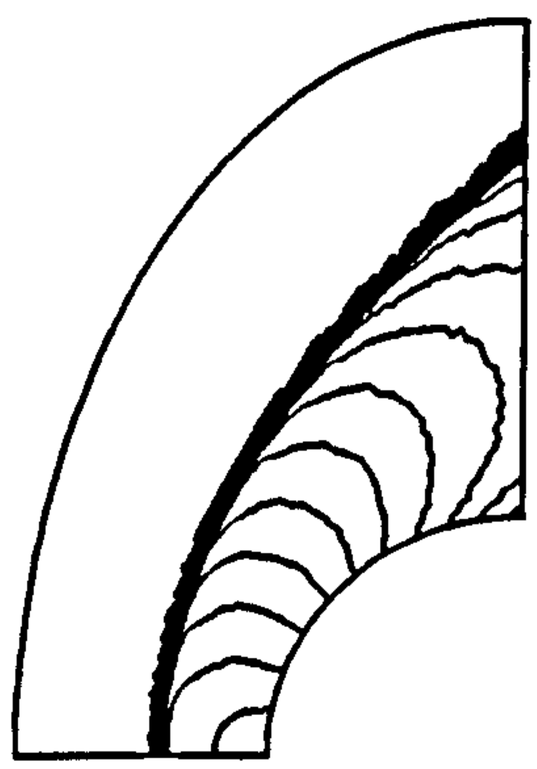
Velocity flow field (zoom)



Pressure contours

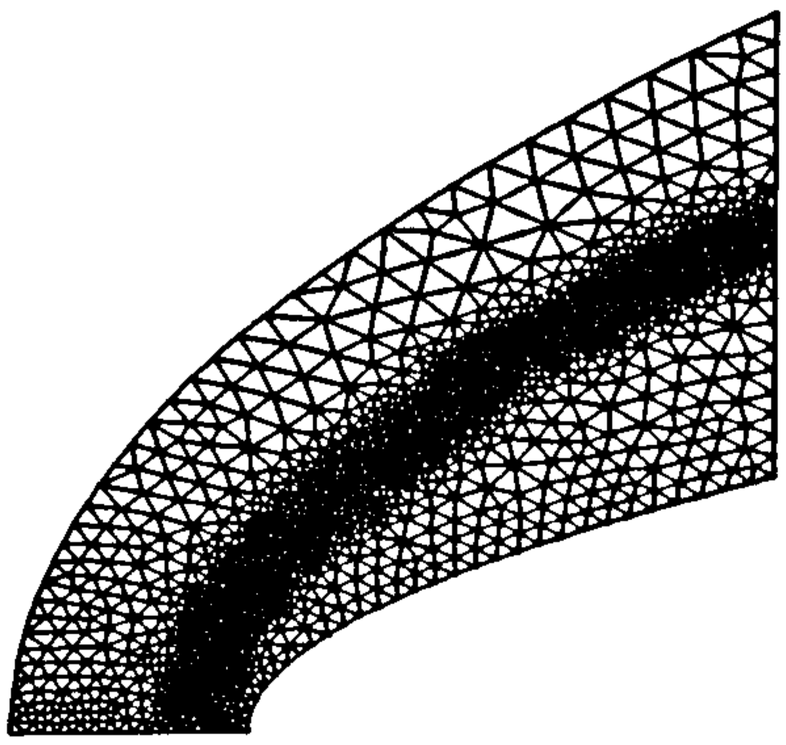


Density contours

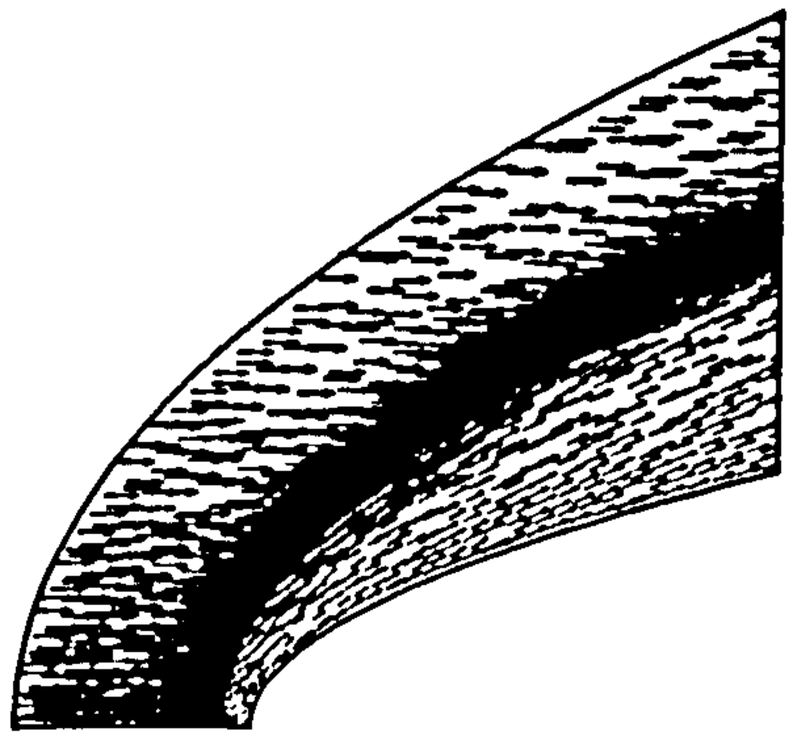


Mach number contours

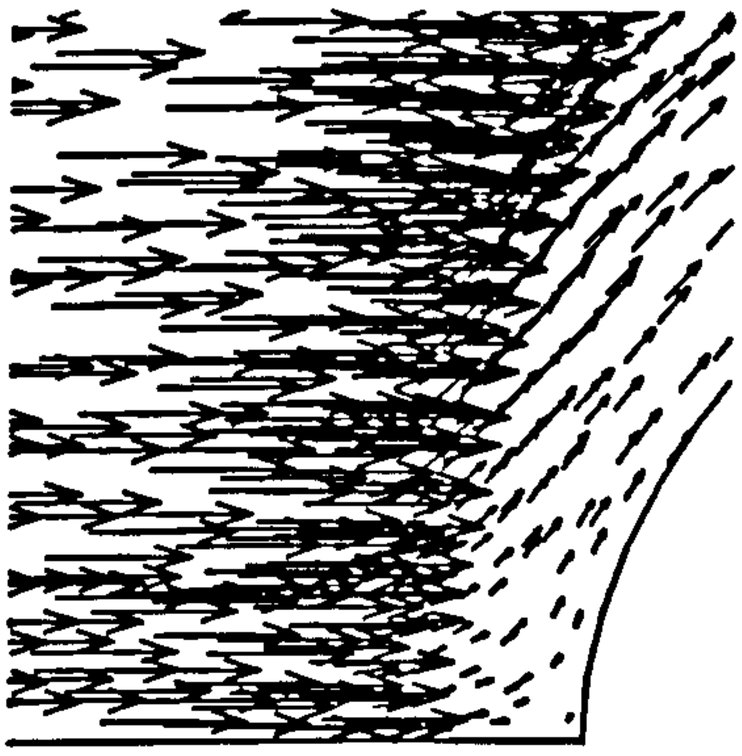
Fig. 12. Adaptive mesh, flow vector field, pressure, density and Mach number contours of the hypersonic flow over a cylinder



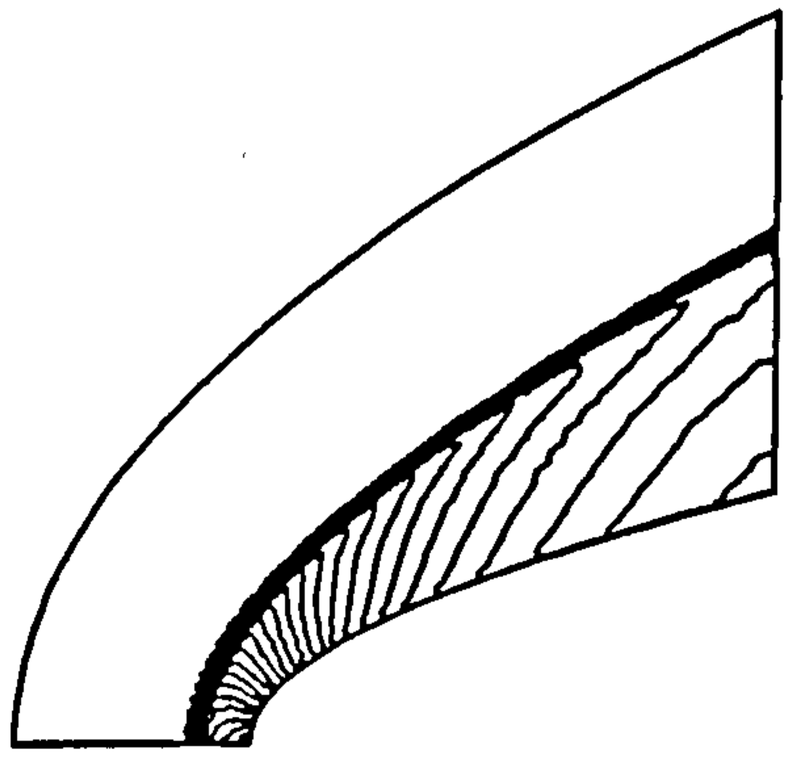
Mesh



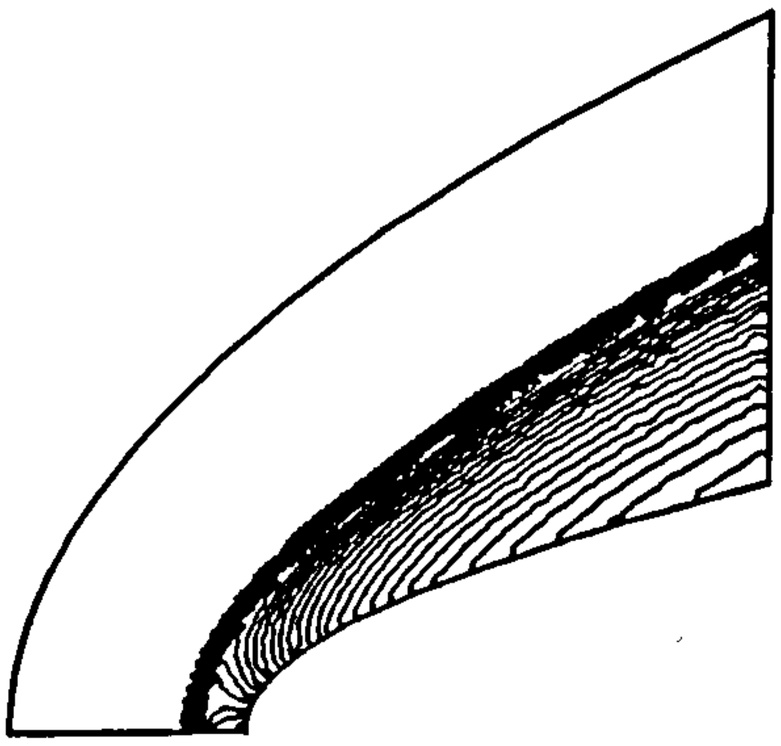
Velocity flow field



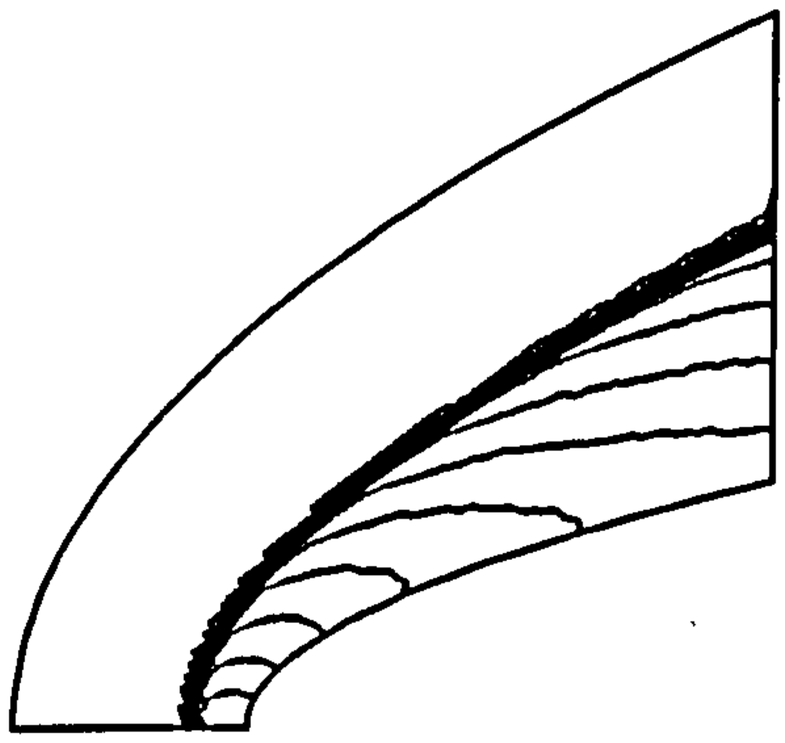
Velocity flow field (zoom)



Pressure contours

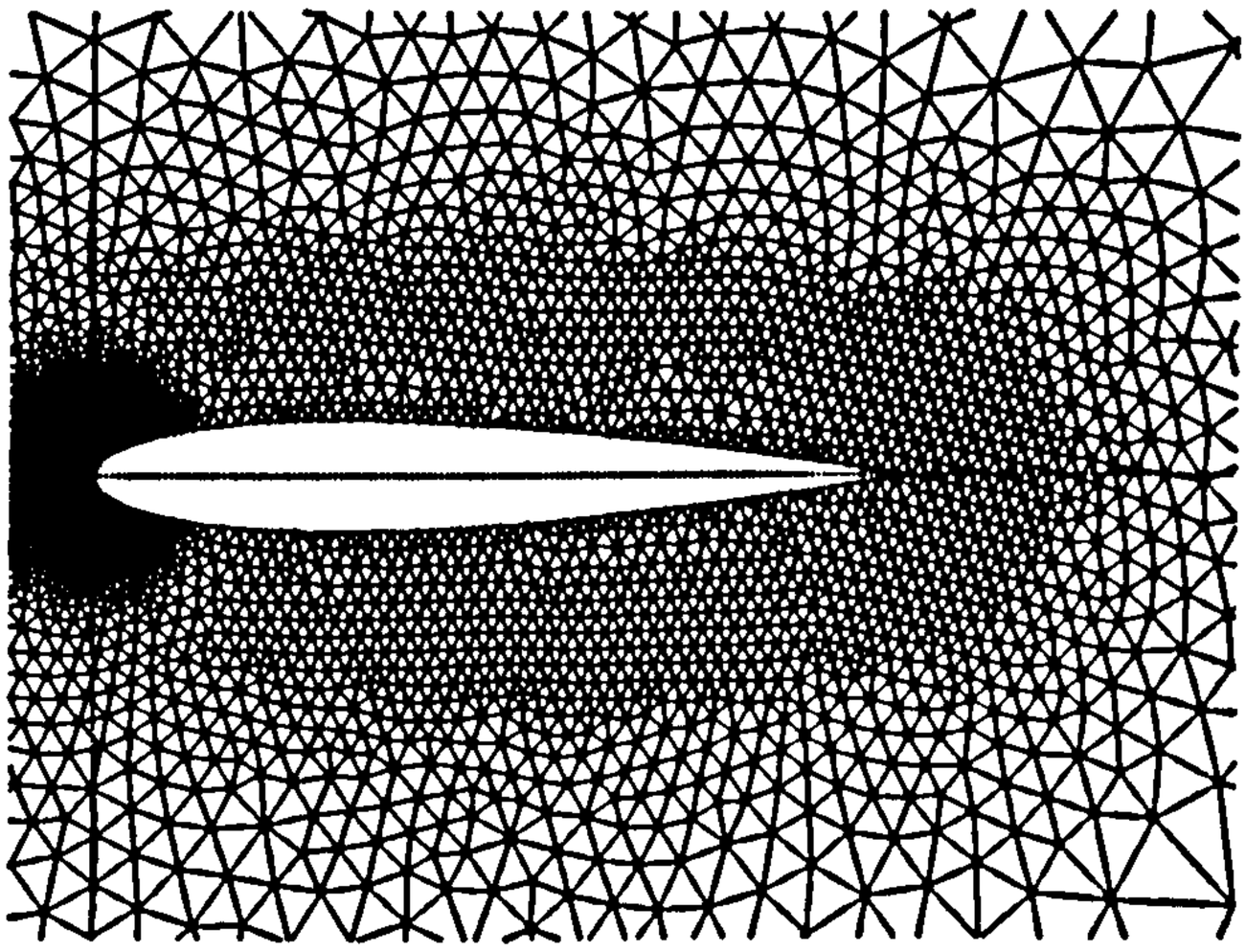


Density contours



Mach number contours

Fig. 13. Adaptive mesh, flow vector field, pressure, density and Mach number contours of the hypersonic flow over a blunt body



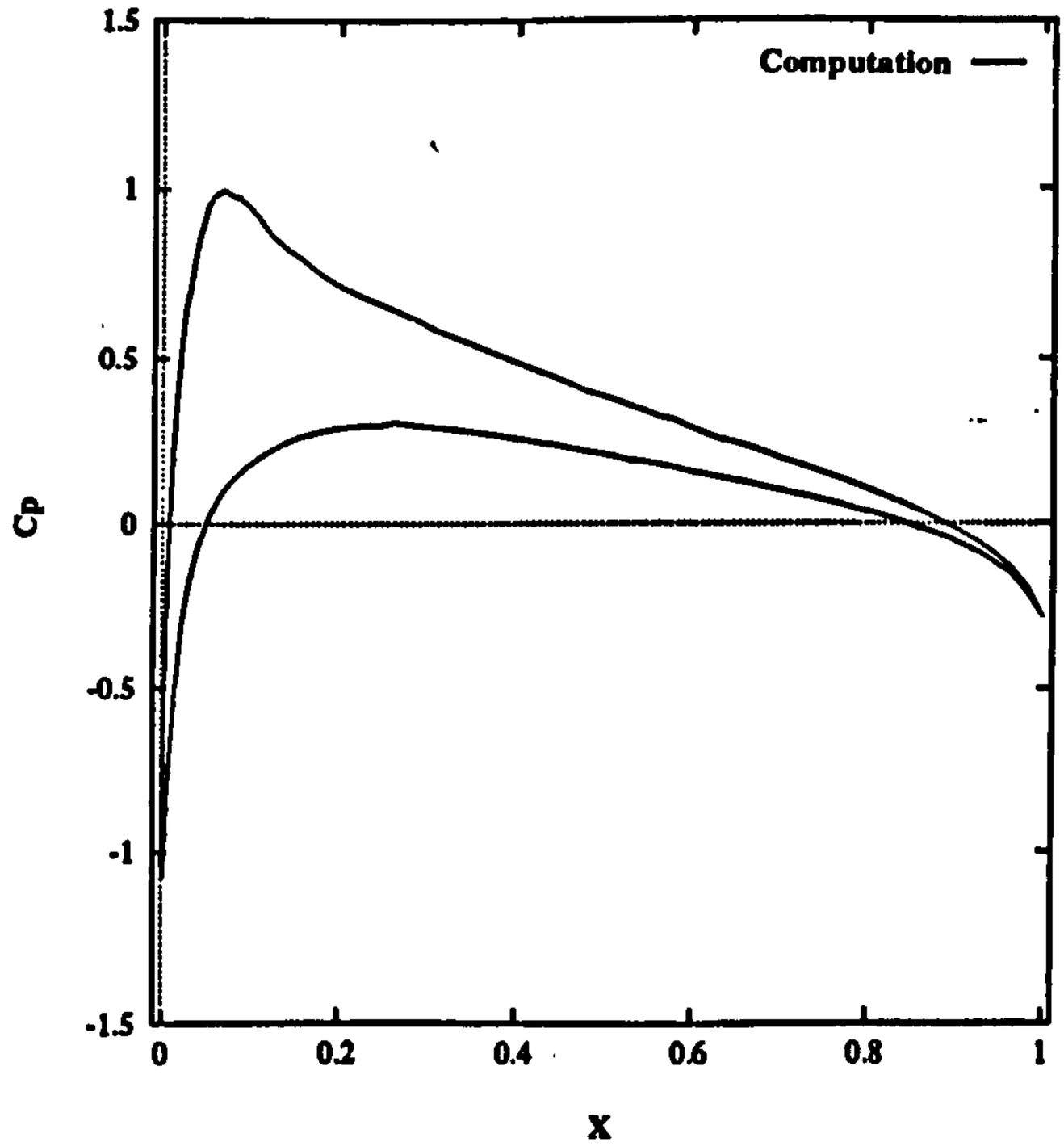
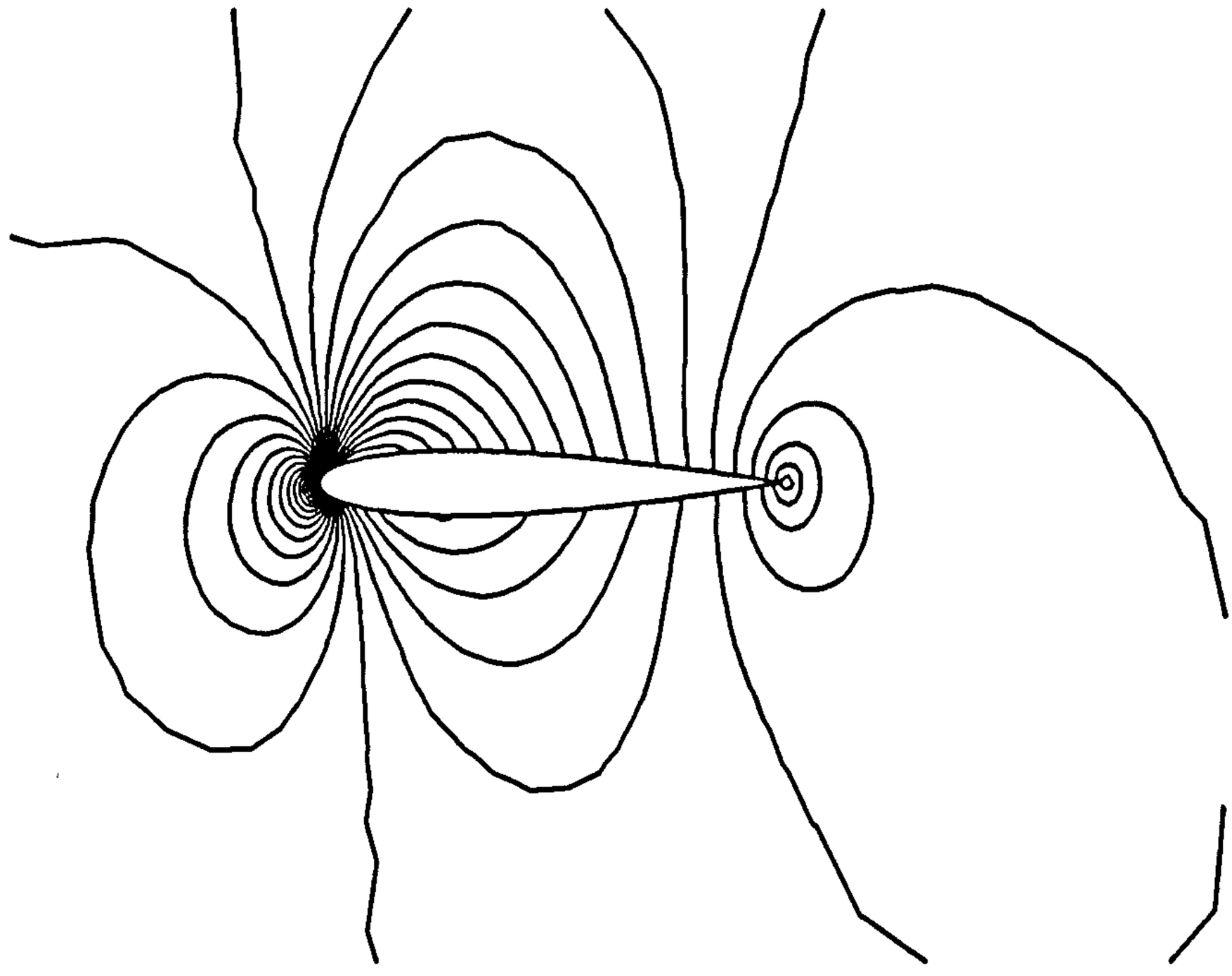


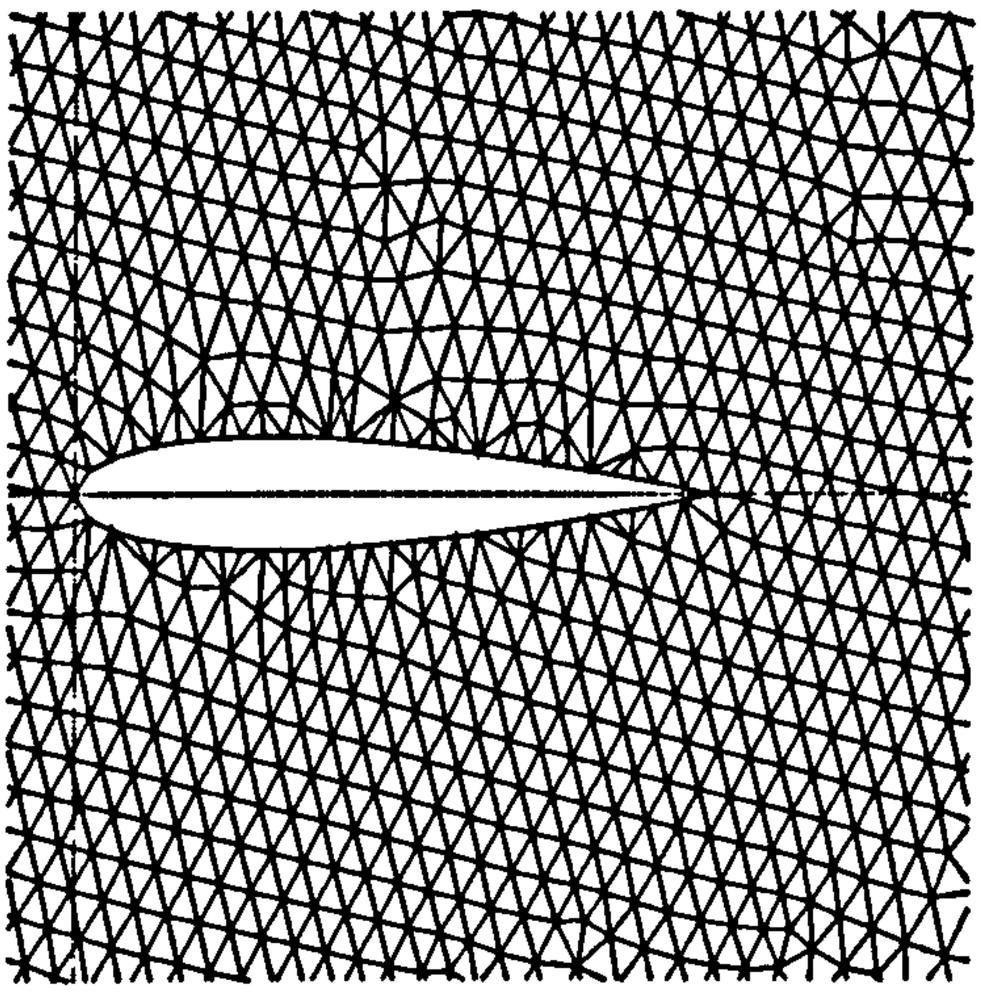
Fig. 14. Mesh and C_p distribution over NACA 0012 airfoil under flow conditions of $M_\infty = 0.63$ and $\alpha = 2^o$



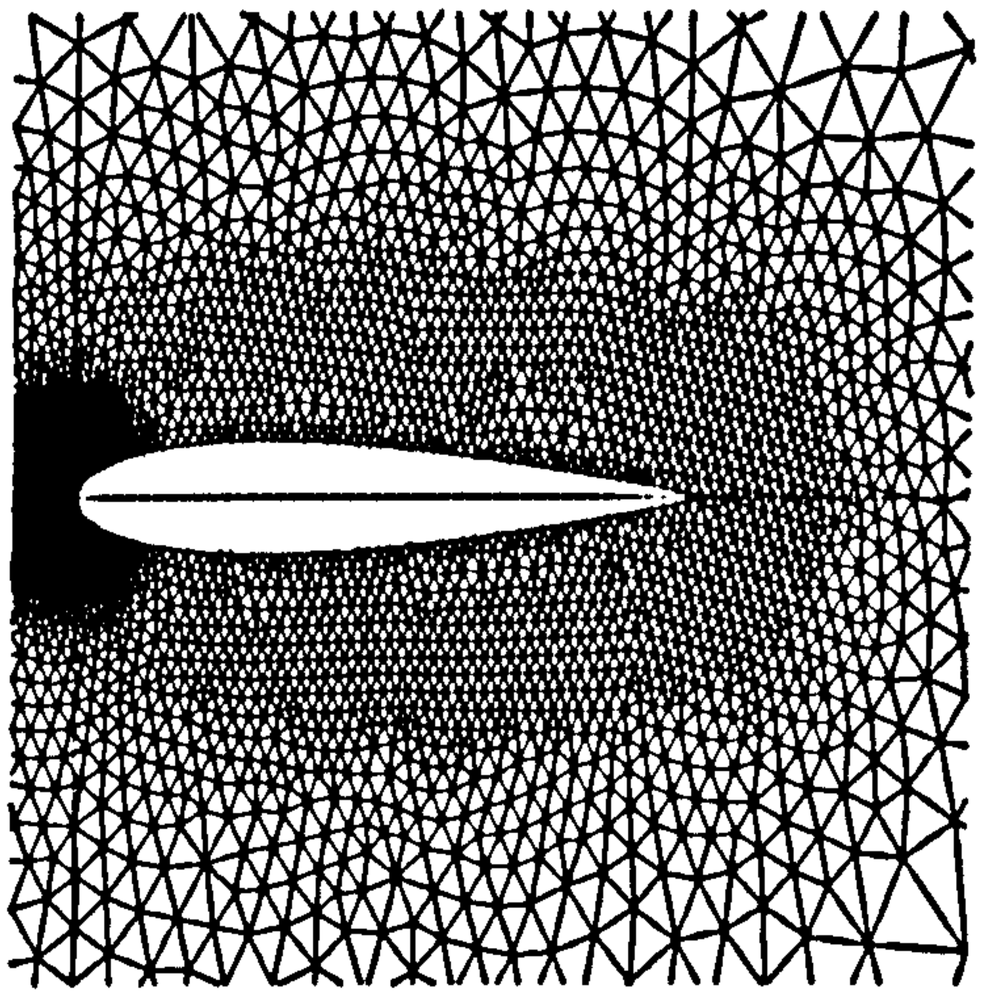
Pressure contours

Mach number contours

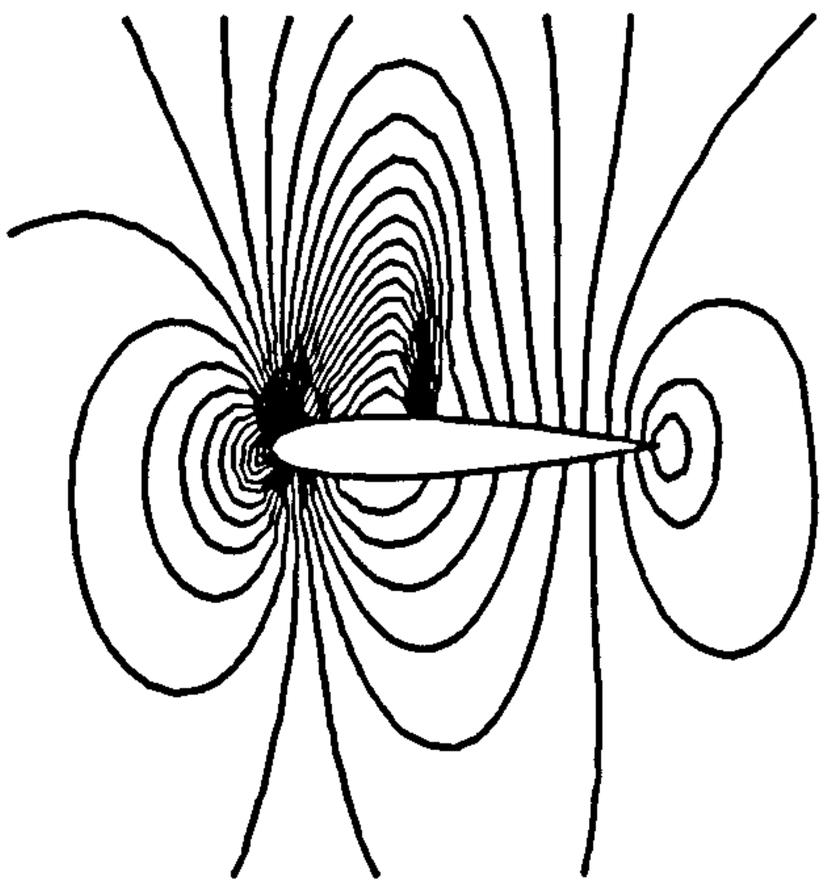
Fig. 15. Pressure and Mach number contours of NACA 0012 airfoil under the flow conditions of $M_\infty = 0.63$ and $\alpha = 2^o$



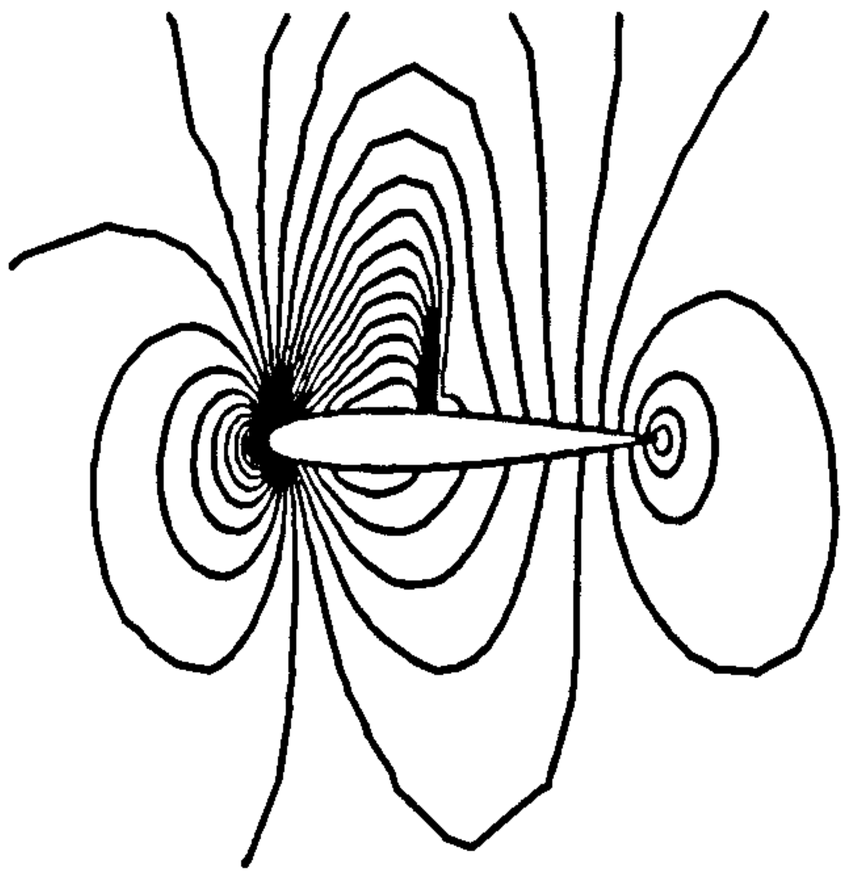
Coarse mesh



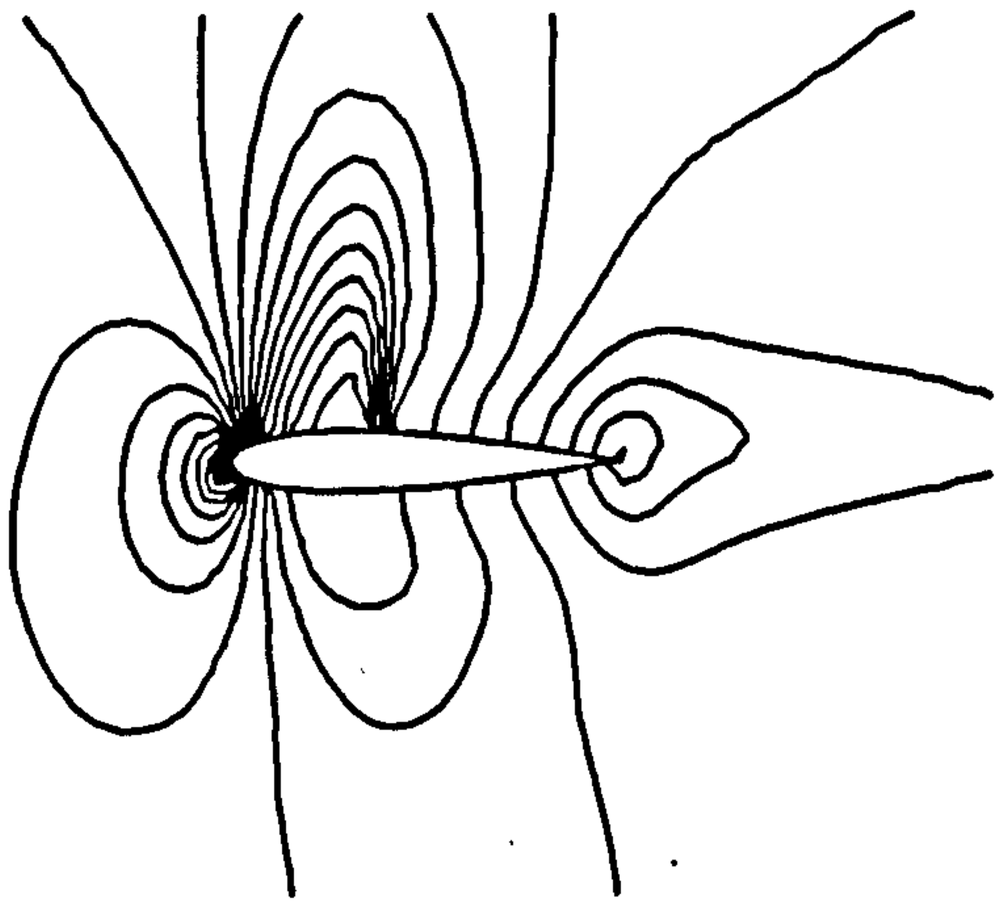
Middle mesh



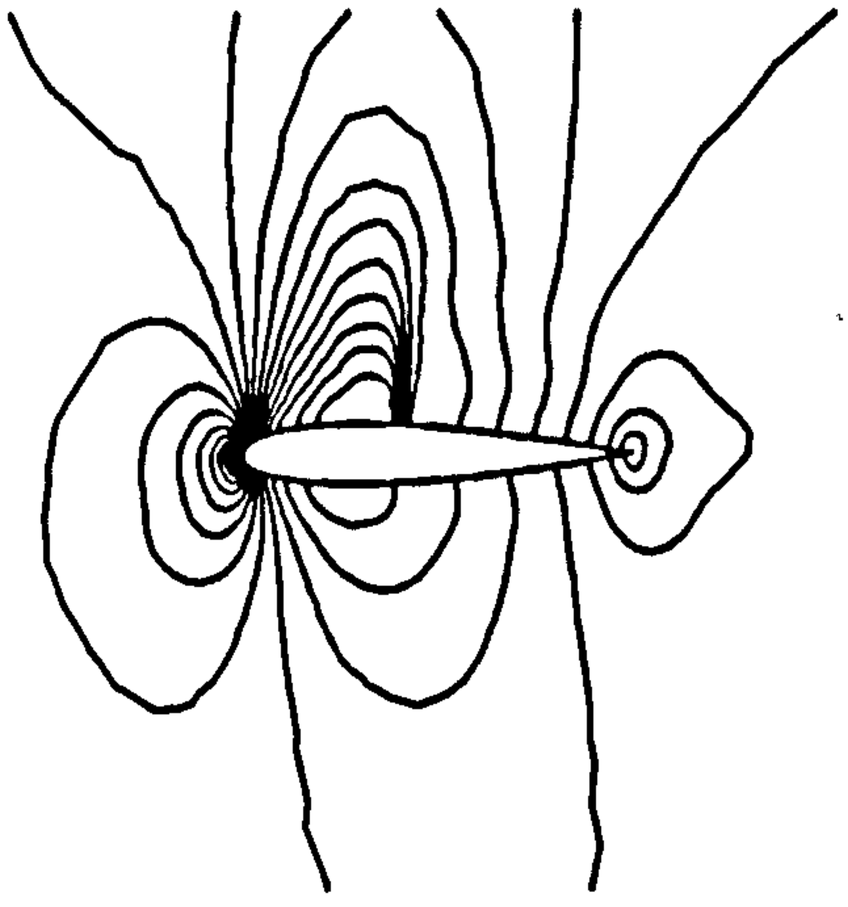
Pressure contours on coarse mesh



Pressure contours on middle mesh

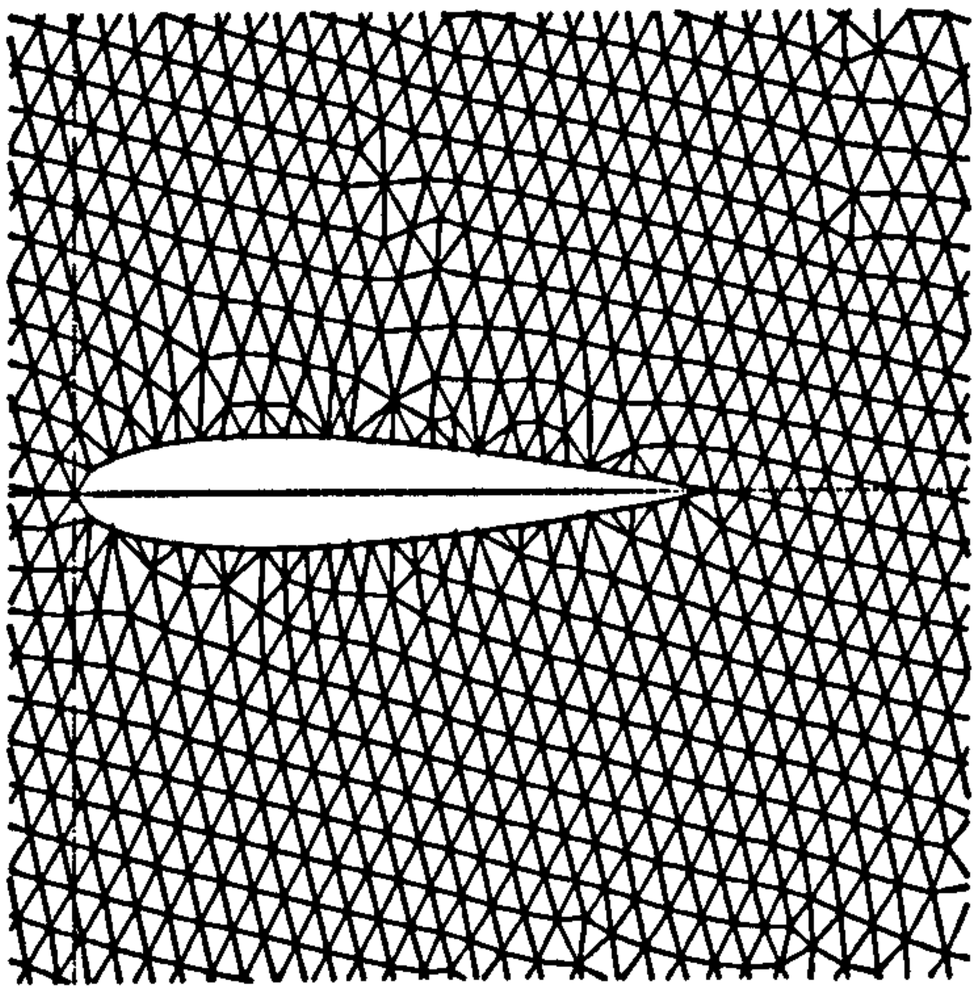


Mach number contours on coarse mesh

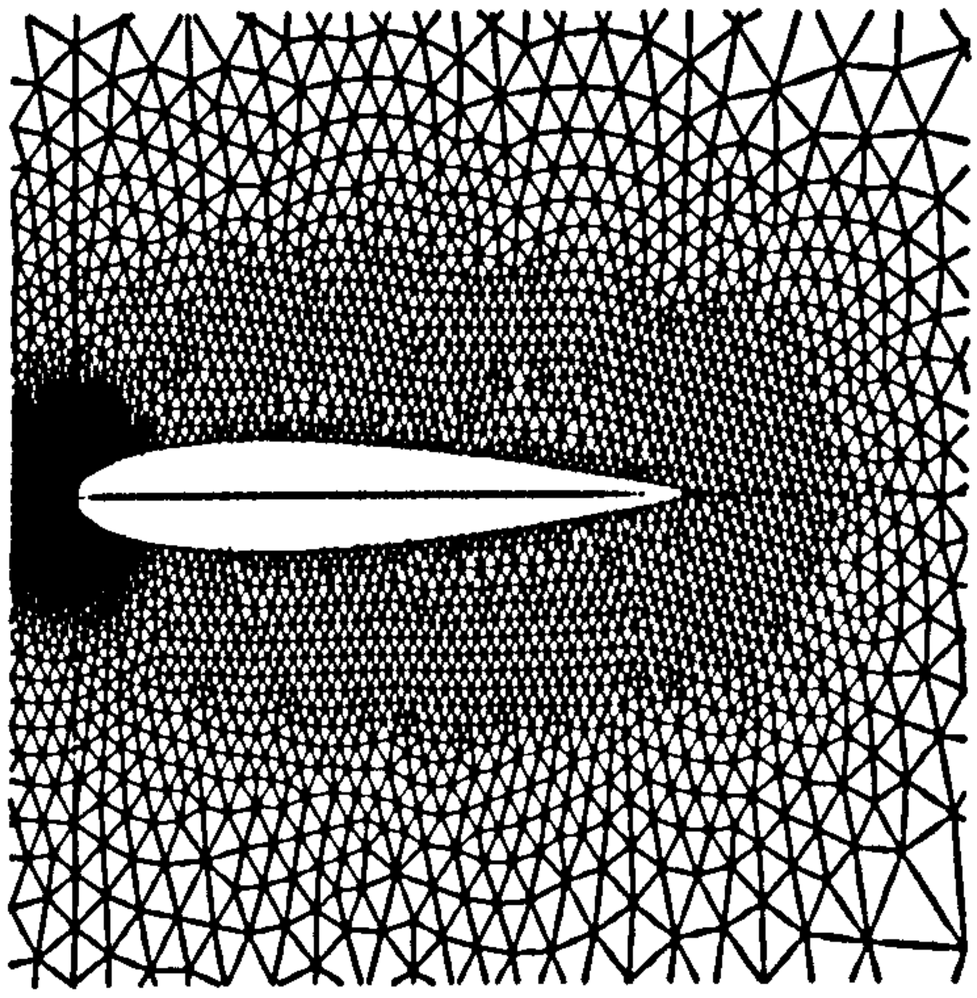


Mach number contours on middle mesh

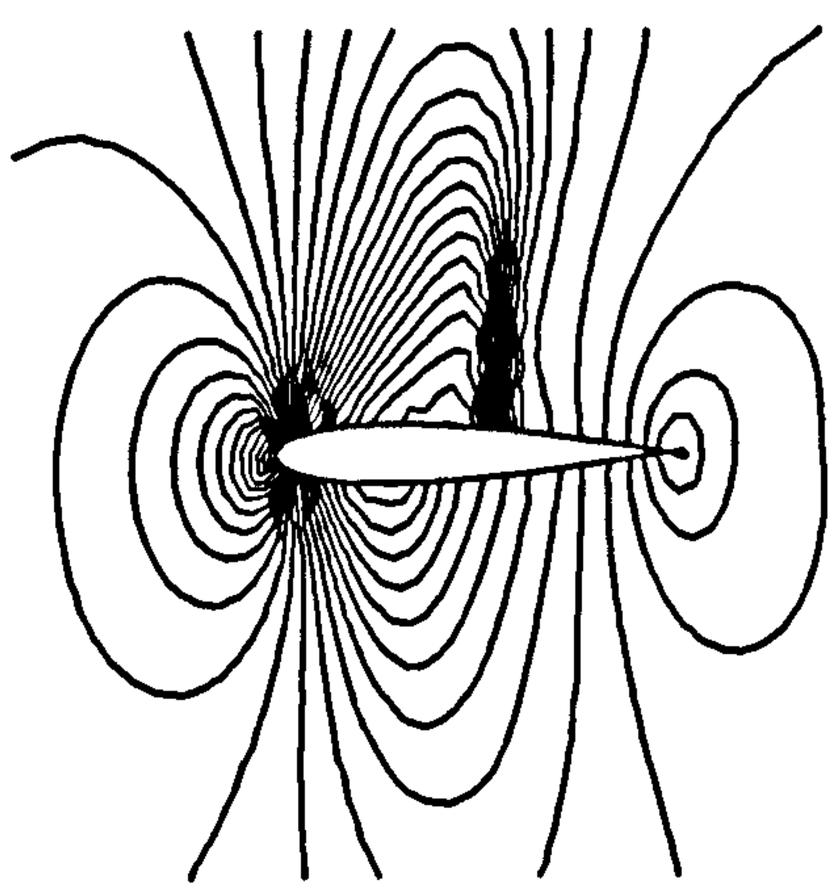
Fig. 16. Meshes, pressure and Mach number contours of the transonic flow over NACA 0012 airfoil ($M_\infty = 0.75$ and $\alpha = 2^o$).



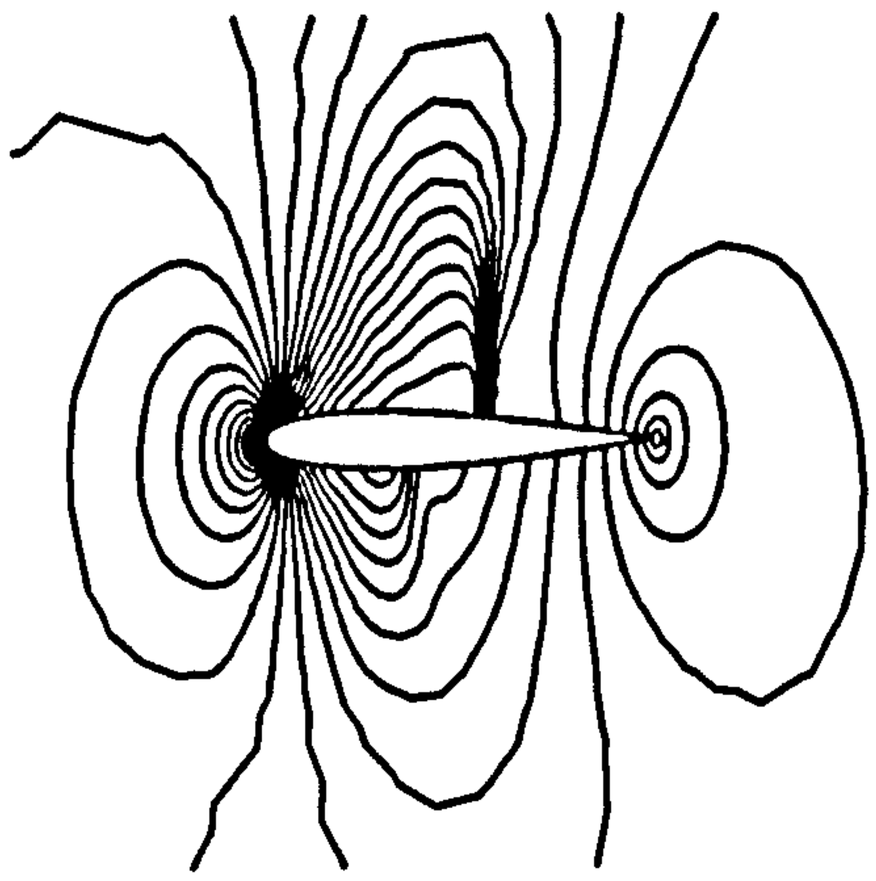
Coarse mesh



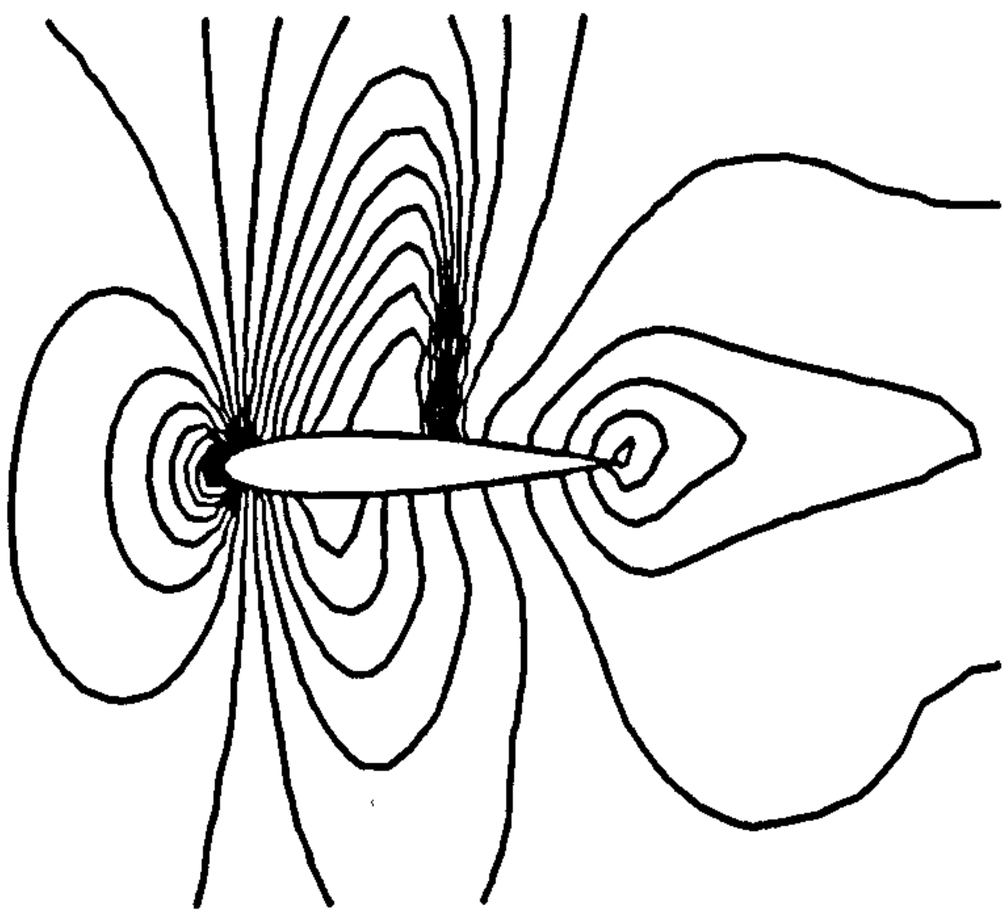
Middle mesh



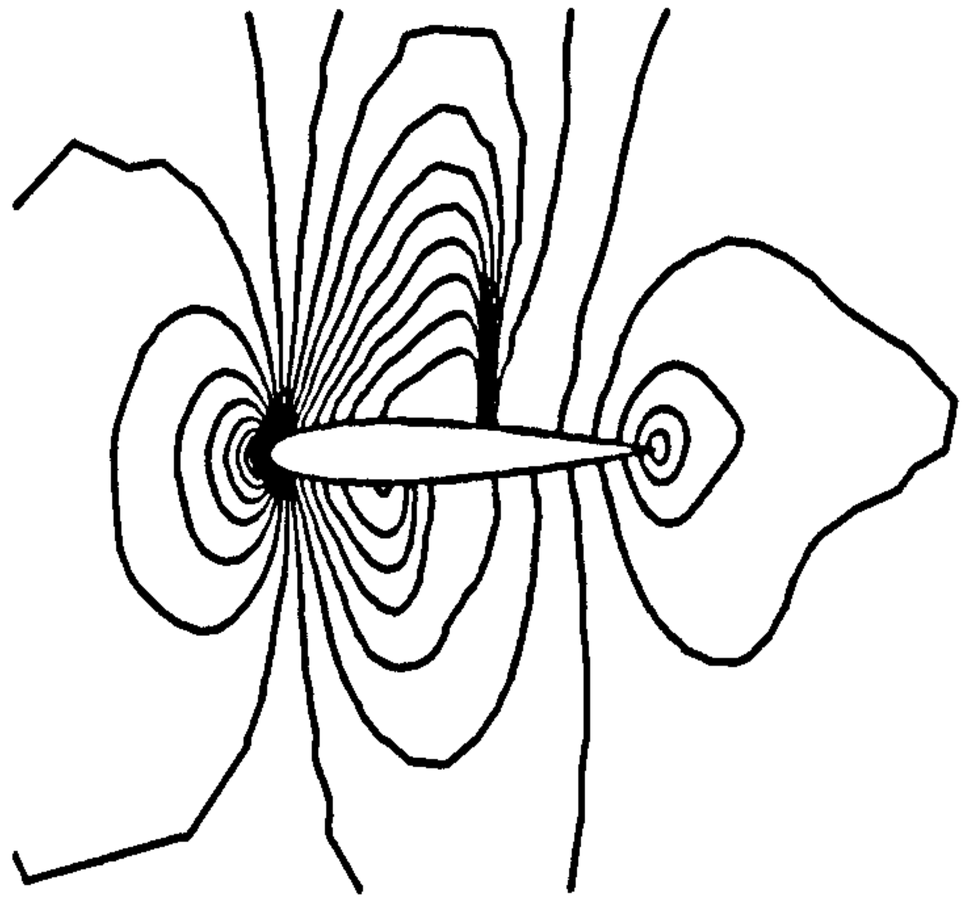
Pressure contours on coarse mesh



Pressure contours on middle mesh

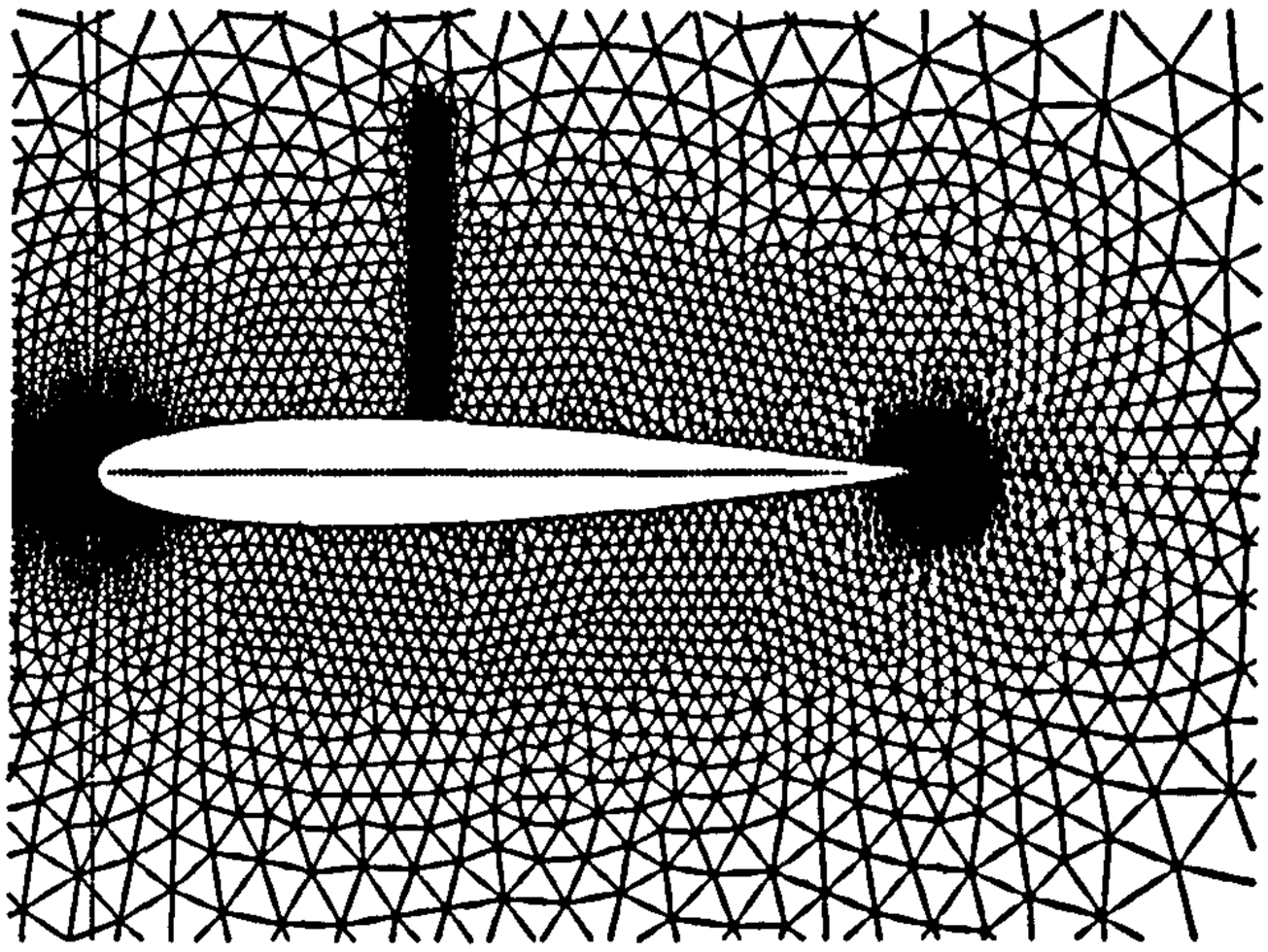


Mach number contours on coarse mesh

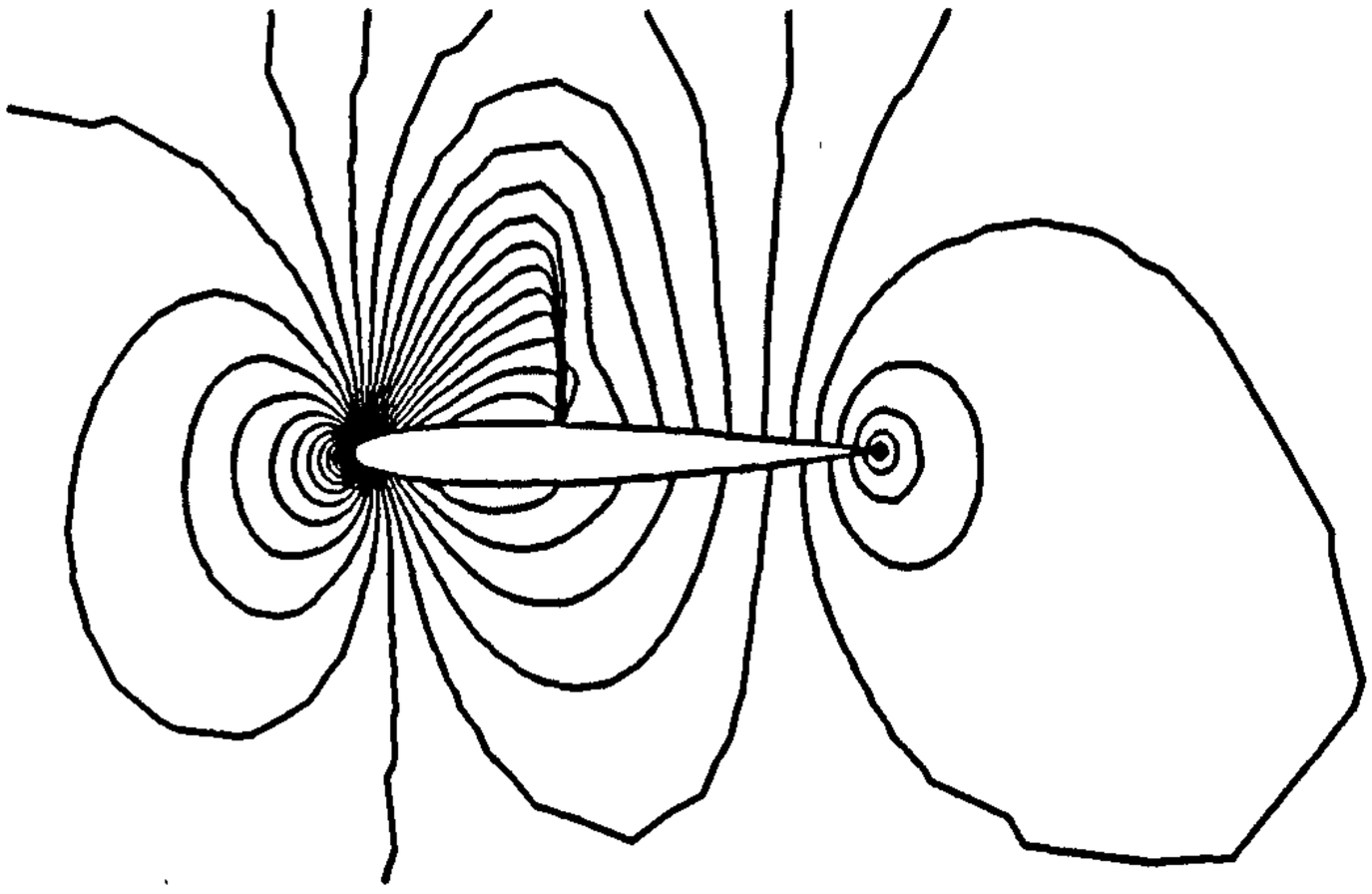


Mach number contours on middle mesh

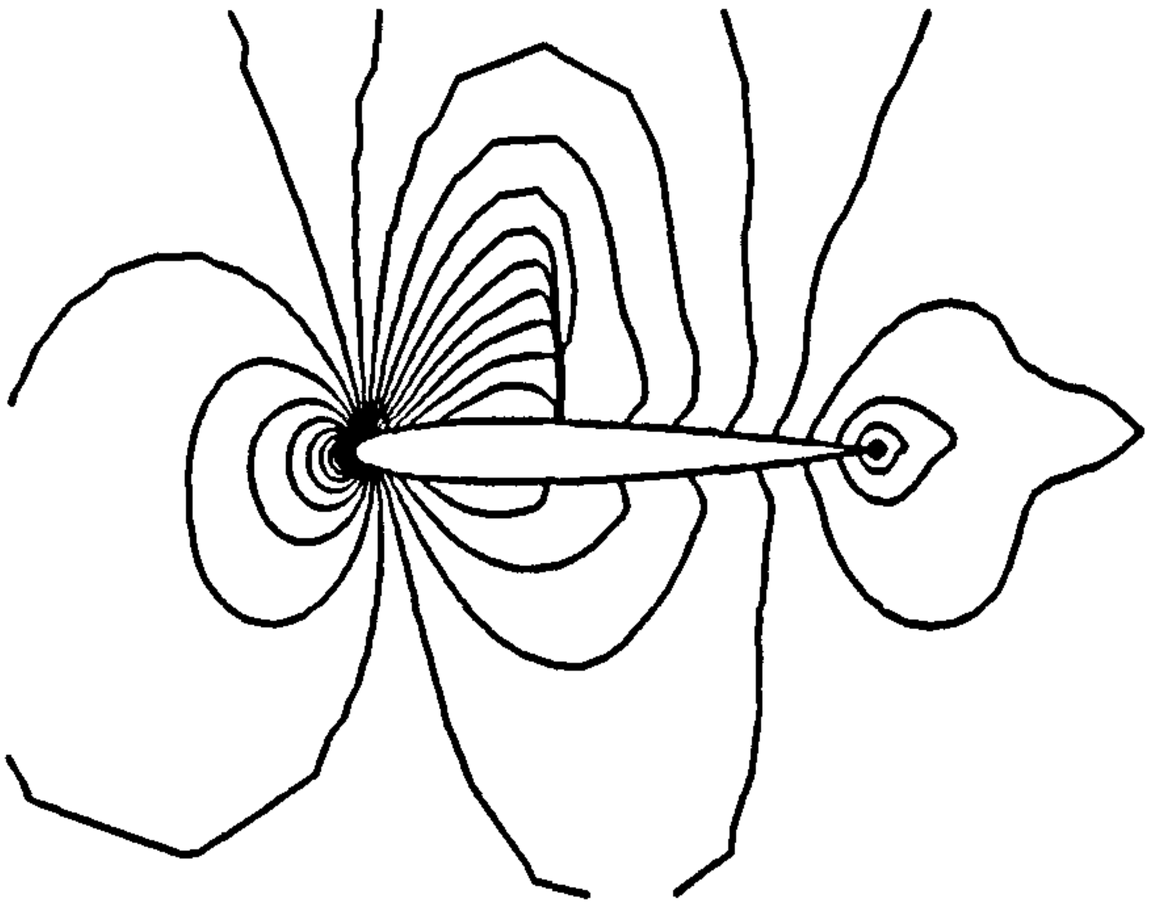
Fig. 17. Meshes, pressure and Mach number contours of the transonic flow over NACA 0012 airfoil ($M_\infty = 0.8$ and $\alpha = 1.25^o$).



Fine mesh

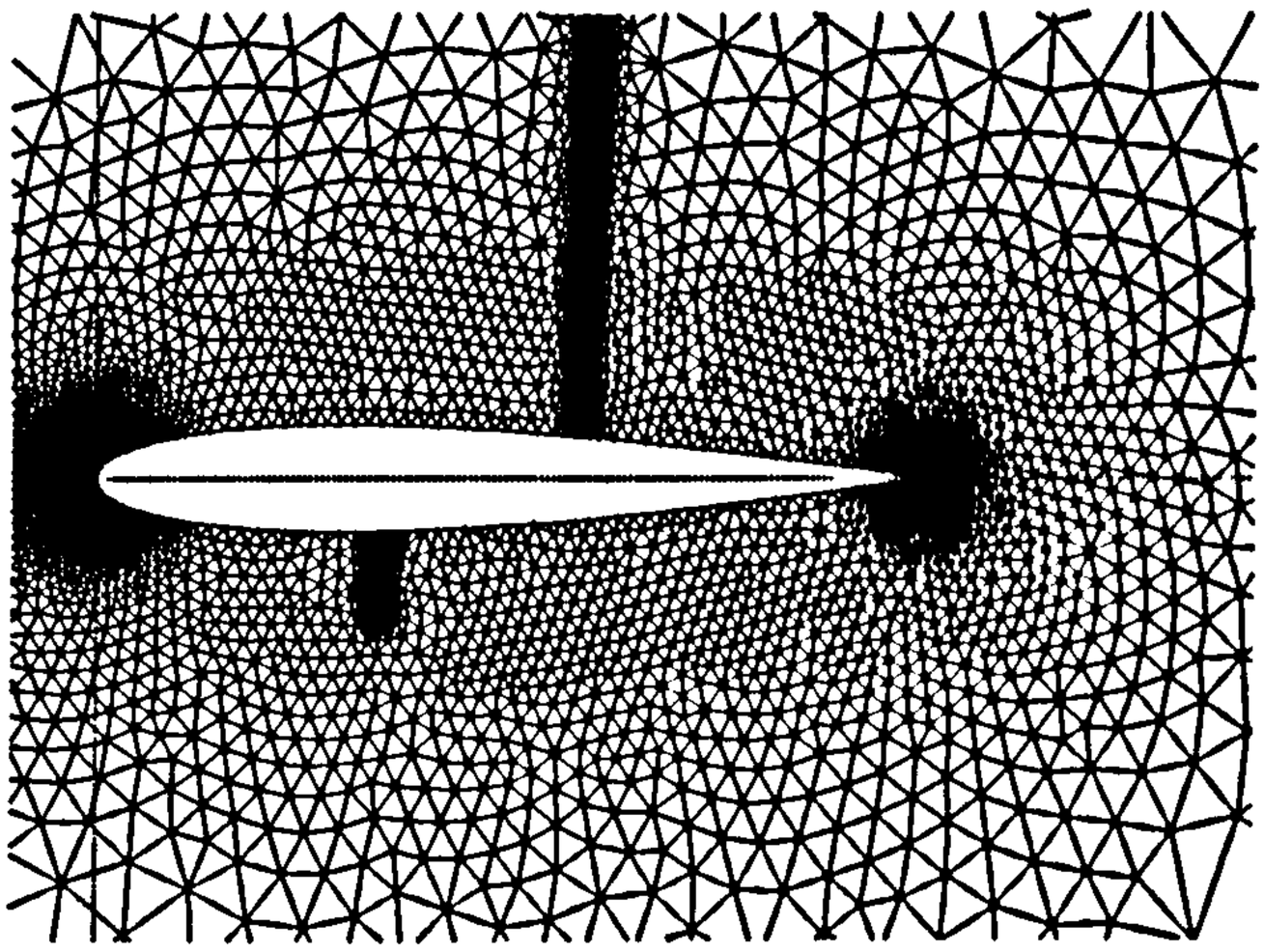


Pressure contours on fine mesh

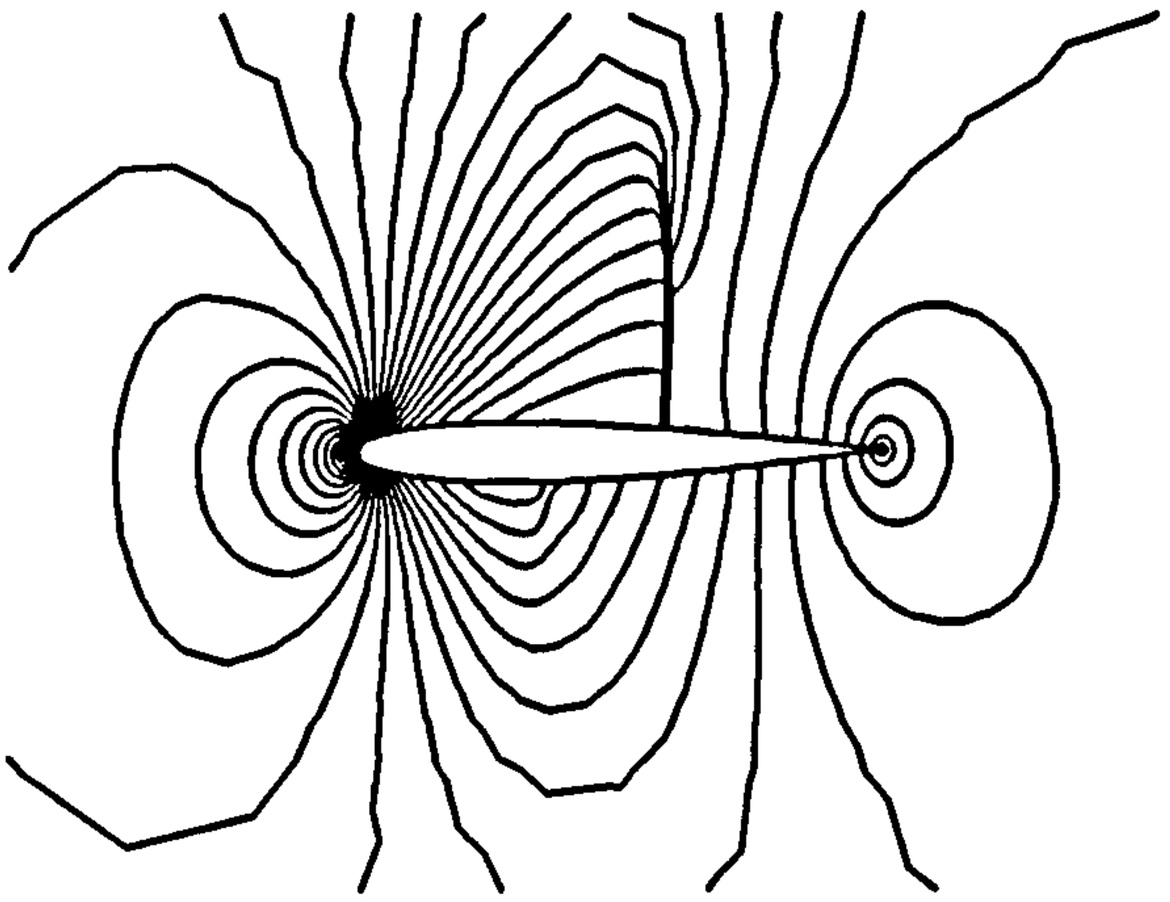


Mach number contours on fine mesh

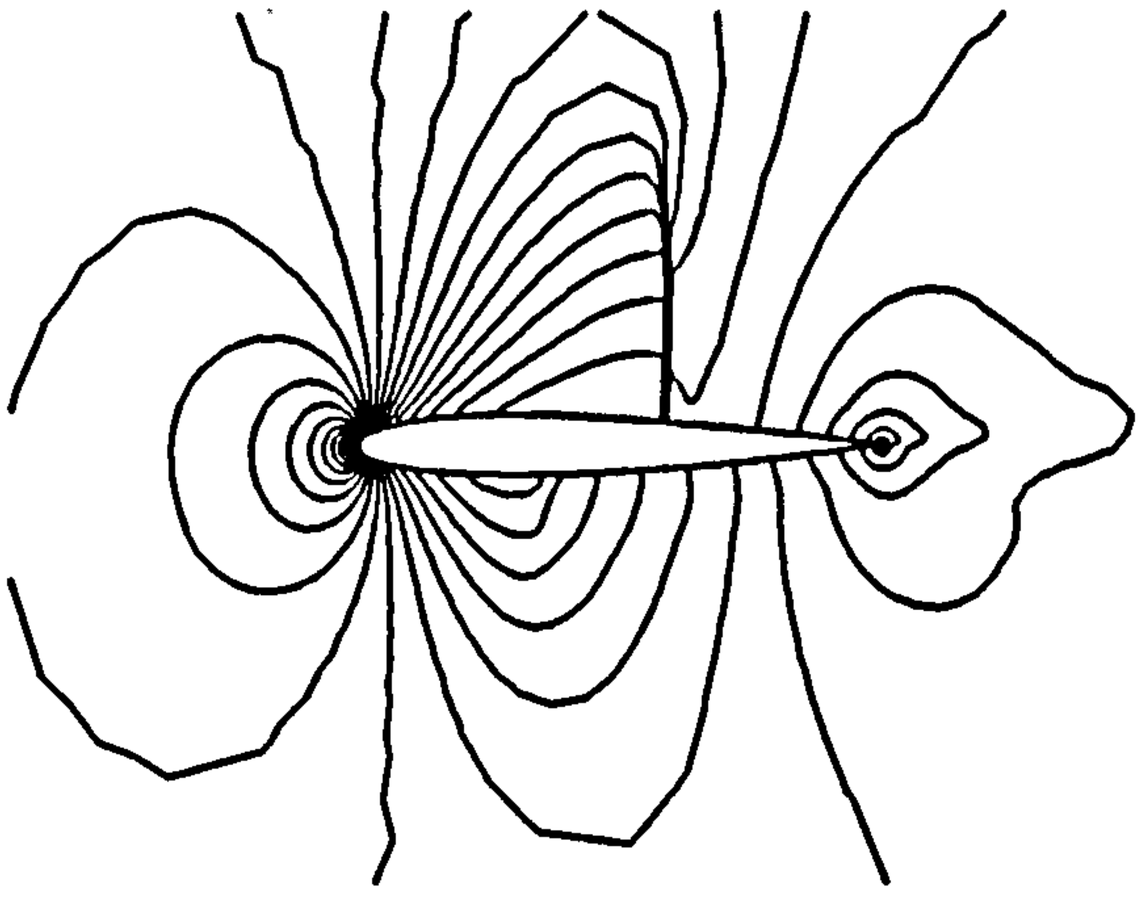
Fig. 18. Adaptive mesh, pressure and Mach number contours of the transonic flow over NACA 0012 airfoil ($M_\infty = 0.75$ and $\alpha = 2^\circ$)



Fine mesh

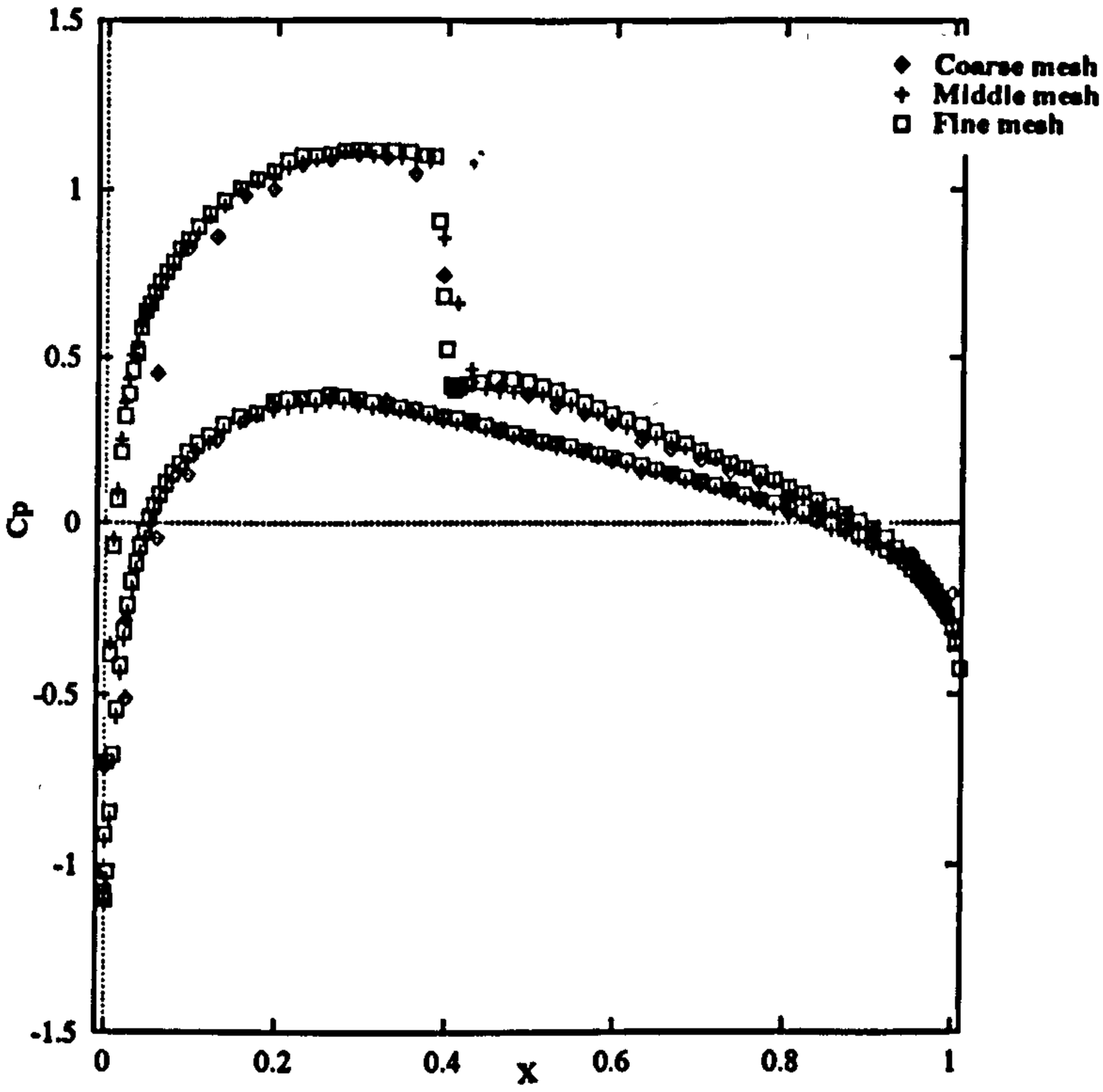


Pressure contours on fine mesh

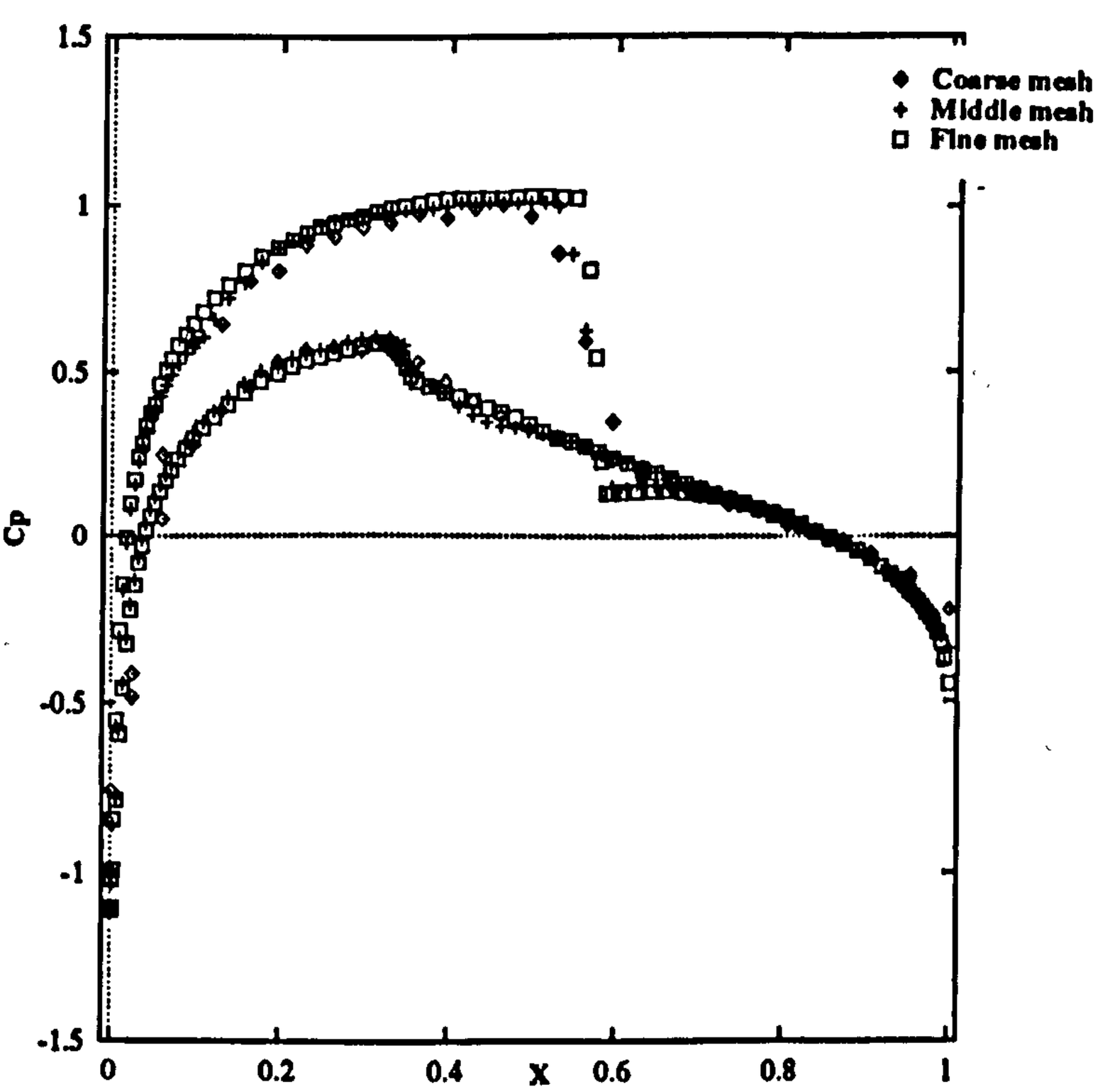


Mach number contours on fine mesh

Fig. 19. Adaptive mesh, pressure and Mach number contours of the transonic flow over NACA 0012 airfoil ($M_\infty = 0.8$ and $\alpha = 1.25^o$)

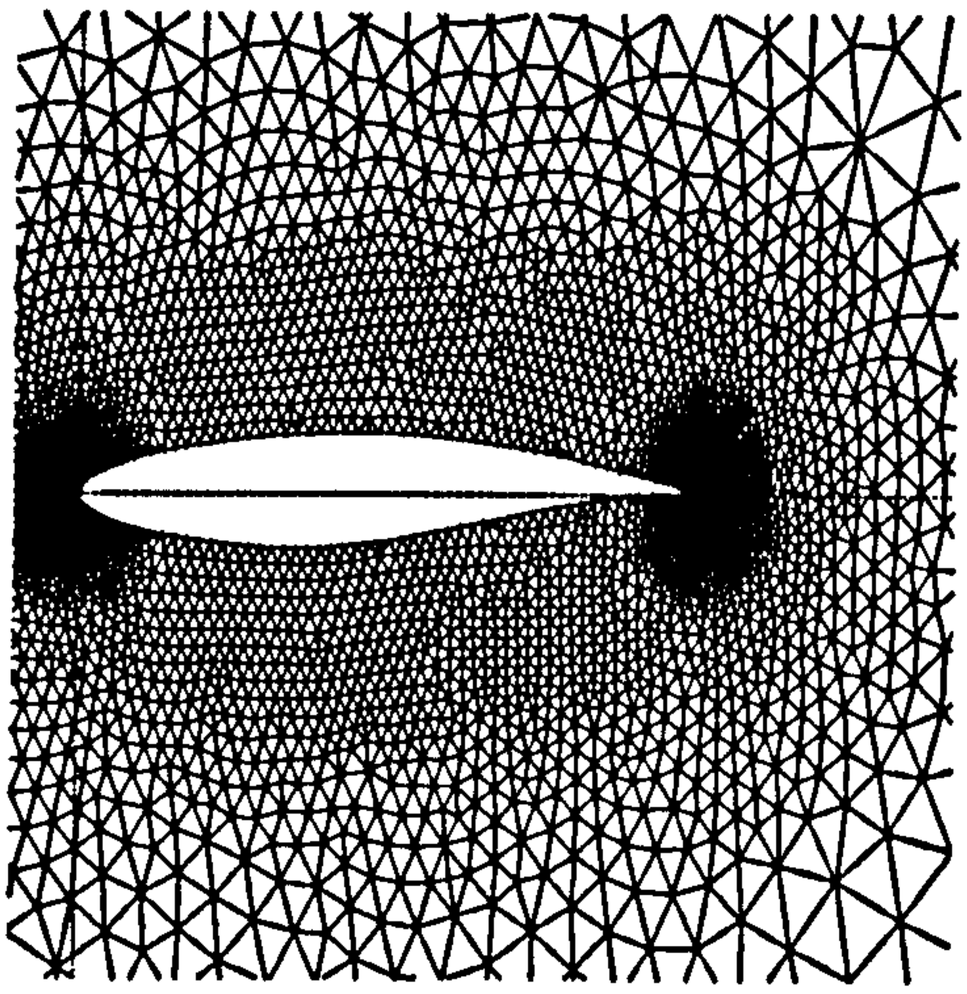


Pressure distribution (Mach=0.75 α=2 degree)

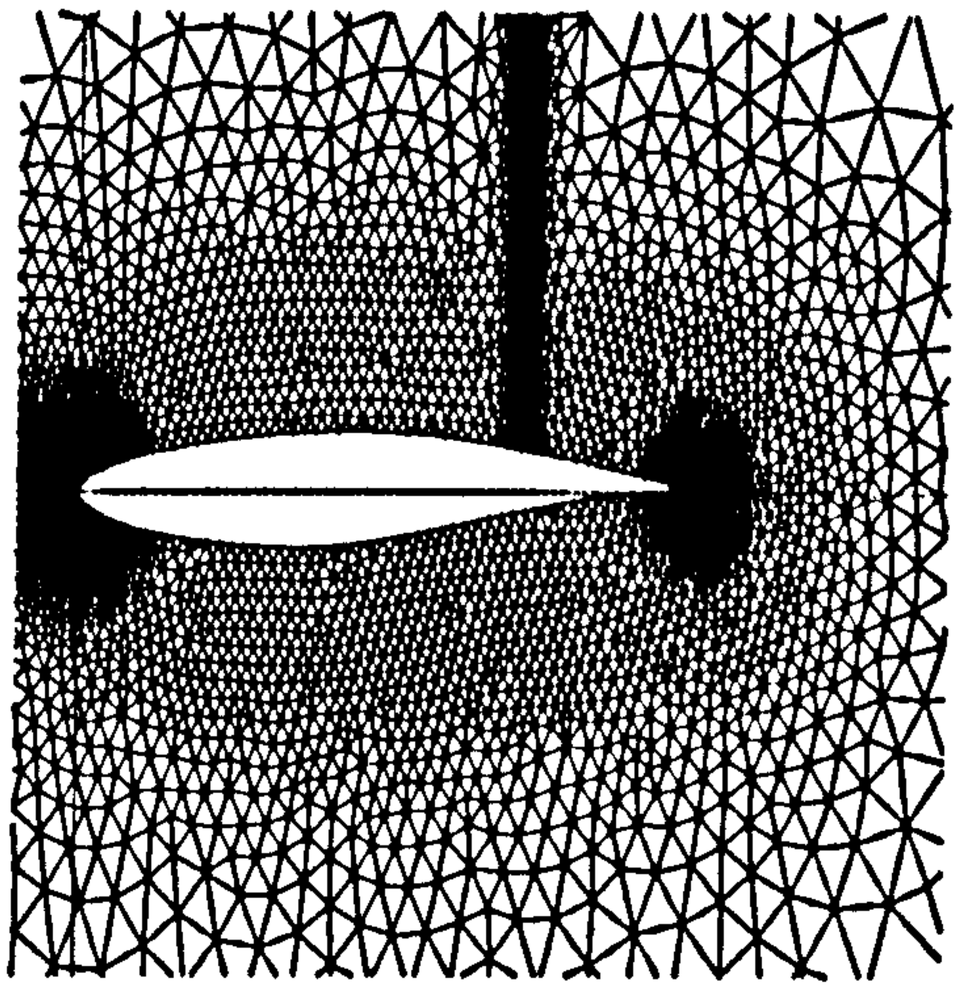


Pressure distribution (Mach=0.80 α=1.25 degree)

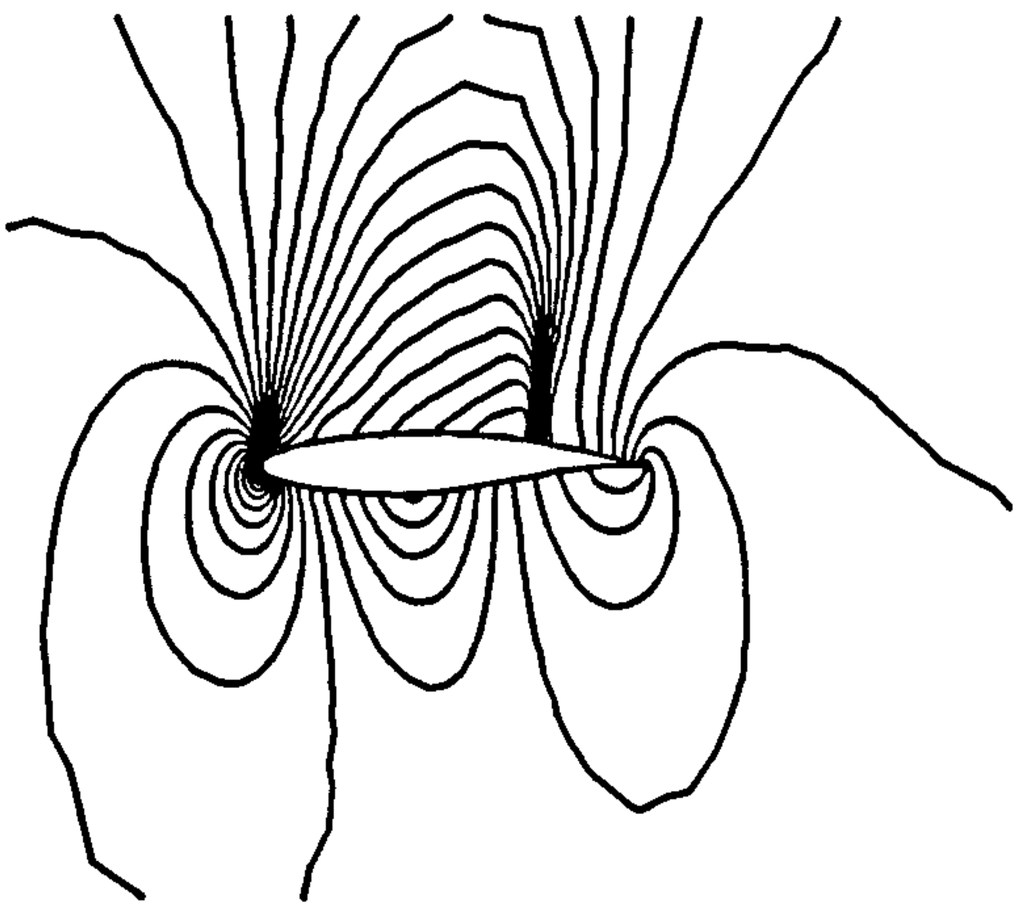
Fig. 20. Comparisons of pressure distributions on coarse and adaptive meshes.



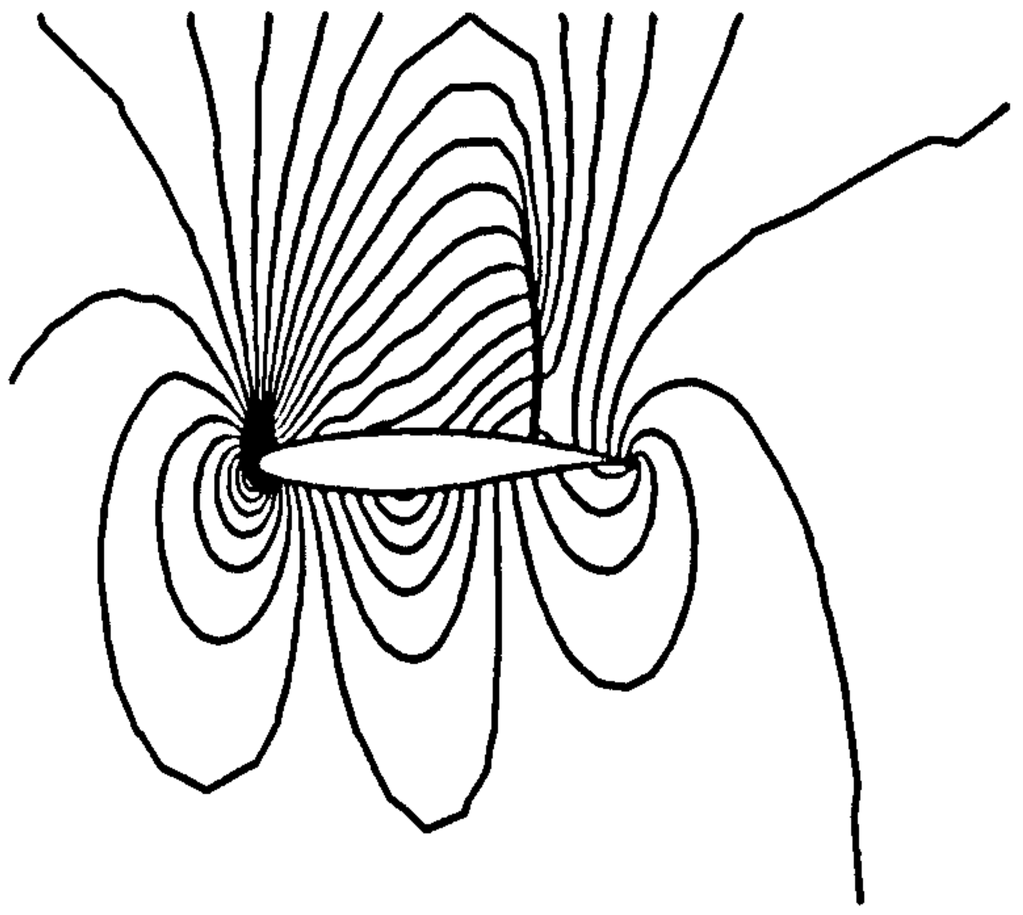
Coarse mesh



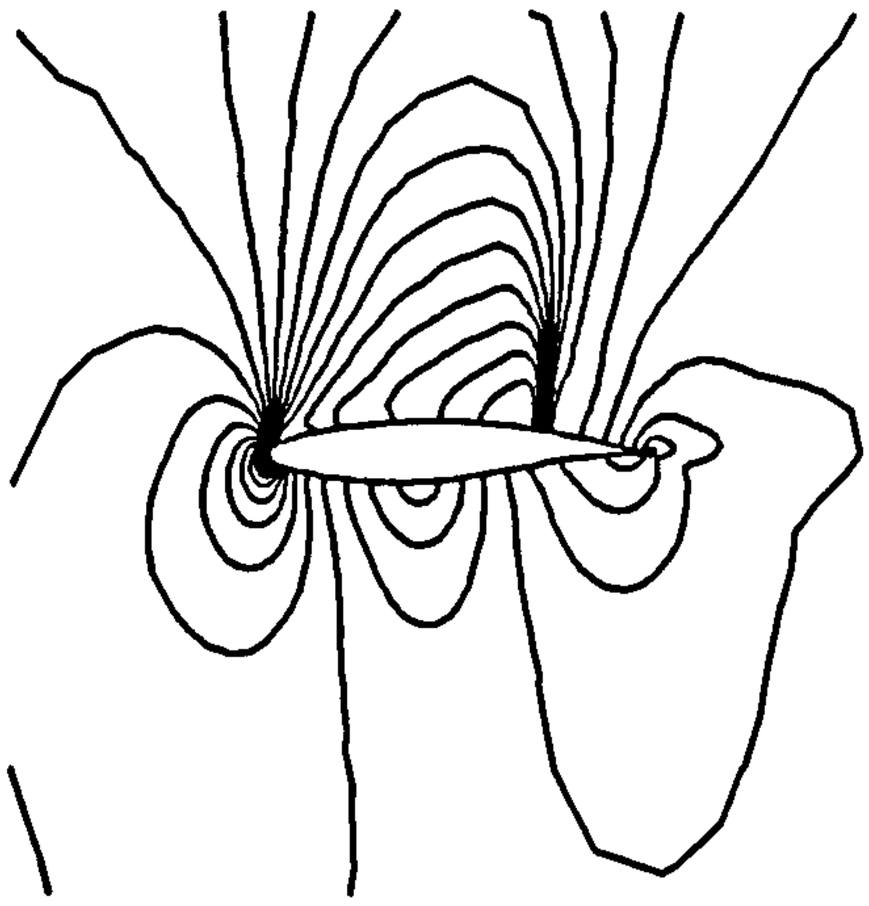
Fine mesh



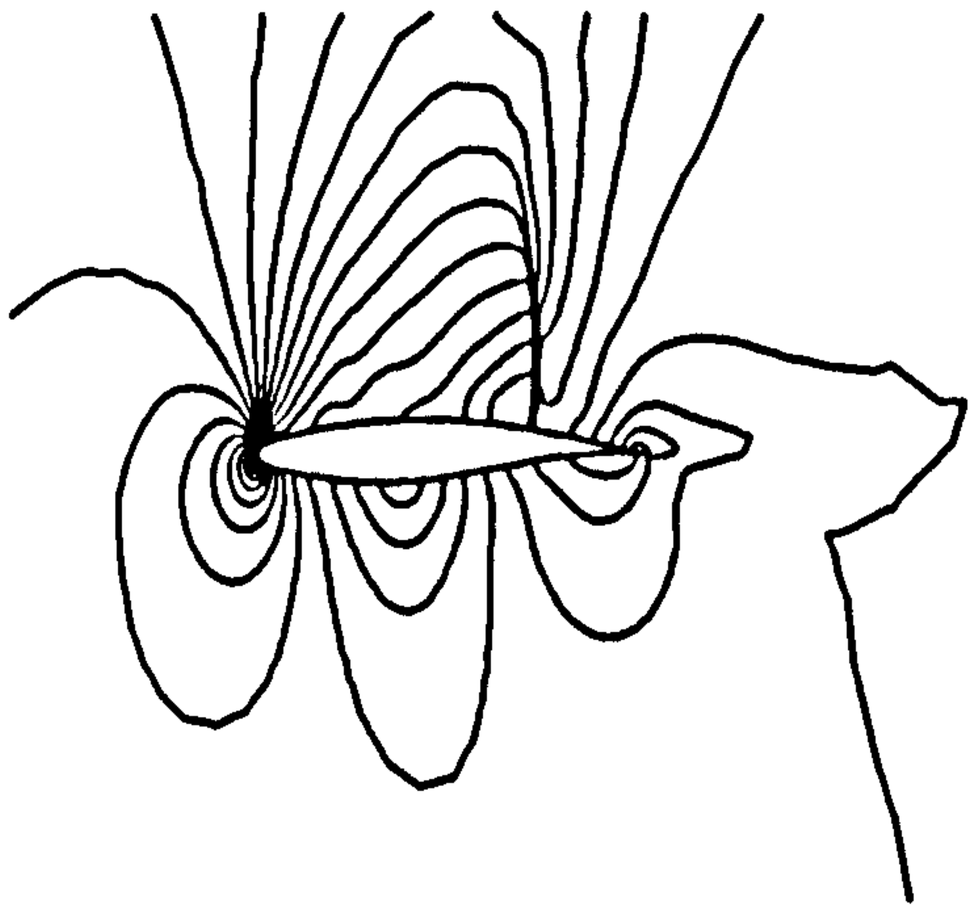
Pressure contours on coarse mesh



Pressure contours on fine mesh



Mach number contours on coarse mesh



Mach number contours on fine mesh

Fig. 21. Meshes, pressure and Mach number contours of the transonic flow over RAE 2822 airfoil ($M_\infty = 0.75$ and $\alpha = 3^o$)

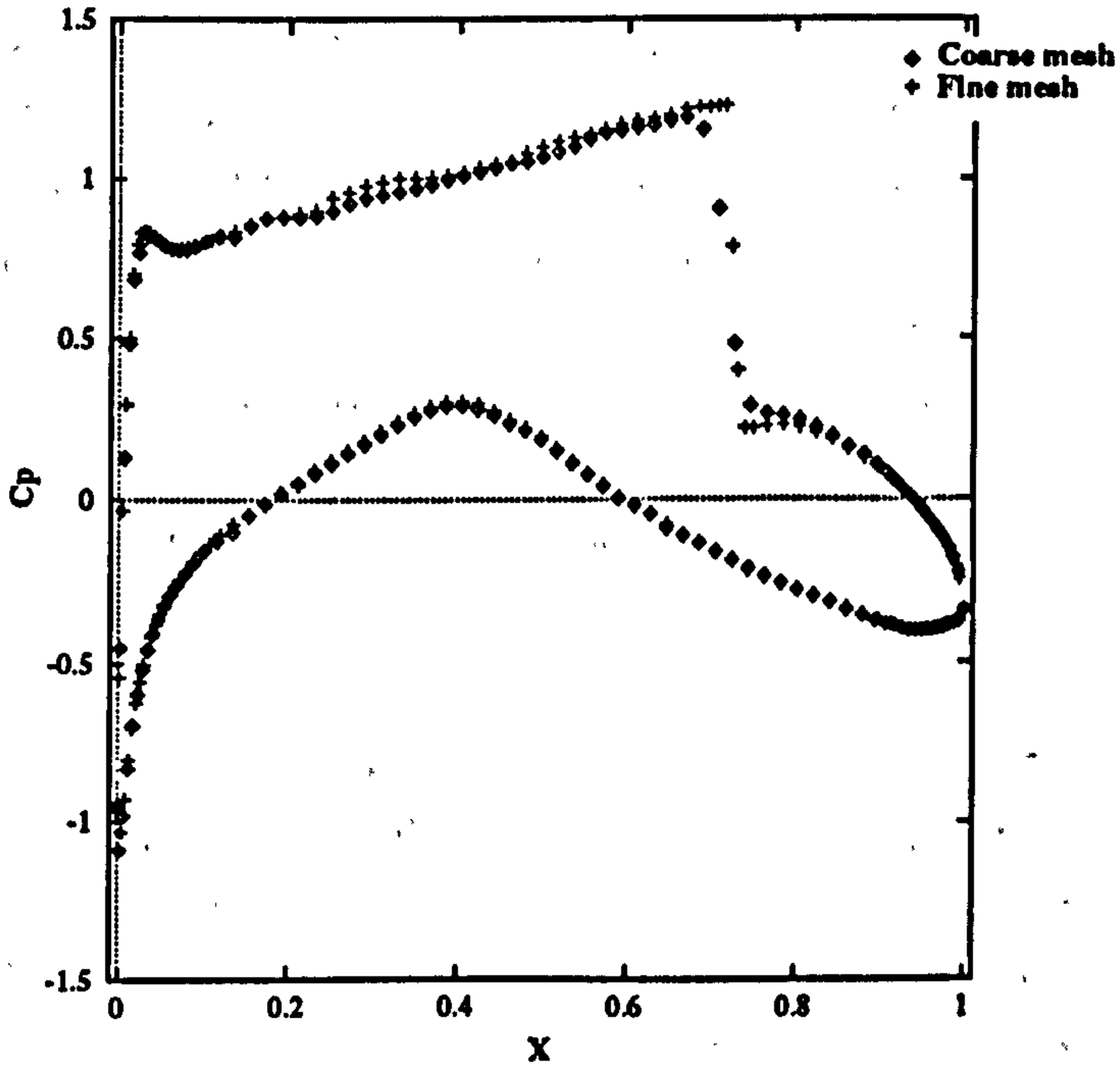


Fig. 22. Comparison of pressure distributions on coarse and adaptive meshes under flow conditions of $M_\infty = 0.75$ and $\alpha = 3^o$

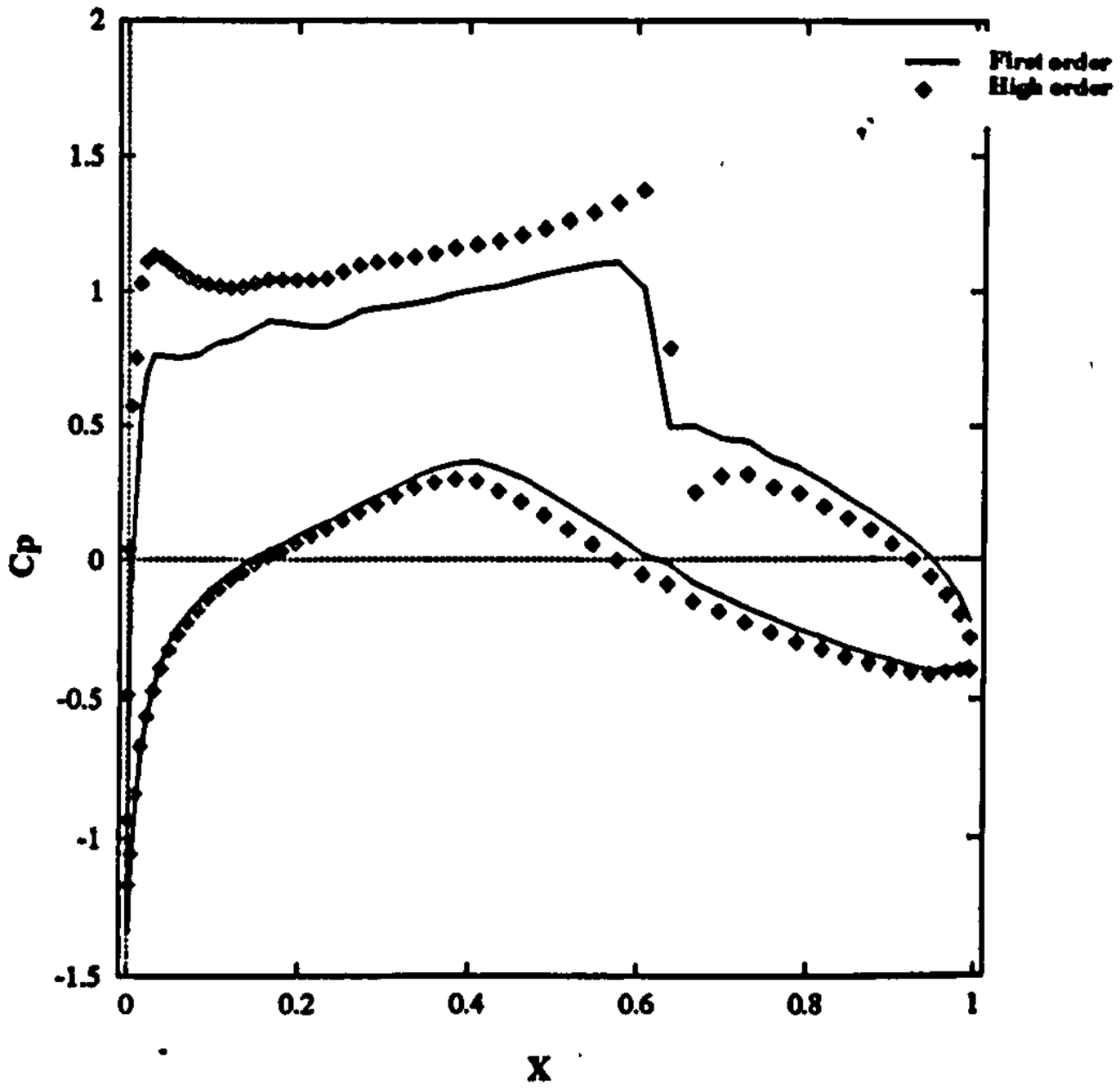
CHAPTER 4

HIGH ORDER RESOLUTION

4.1. Introduction

It has long been known in the computational fluid dynamics (CFD) community that the use of high order resolution can deal with practical flow problems at higher accuracy, while use of the first order scheme often cannot do so. An example of the latter is given in Figure 23 which illustrate the results of surface pressure distributions and convergence history of the transonic flow over the RAE 2822 airfoil under the conditions of Mach number 0.729 and incidence 2.31°. These results are given by the Euler flow solver (PGS-Roe code) described in the previous chapter, which is an upwind cell-centred finite volume method on unstructured grids with first order scheme considered. Although the residual has been reduced rapidly to -5 orders, the pressure is still unreasonable because of (1) the value near the upper surface of leading edge is not fully recovered and (2) the size of the jump at the shock wave position is not sufficiently high. The same problem is also revealed by Hirsch et al in their recent paper [86]. So it is concluded that the high order scheme must be used in CFD computation.

Concerning upwinding on structured grids, many high order reconstruction methods have been proposed and successfully applied. Among the popular ones used are the Flux Corrected Transport (FCT) scheme of Boris and Book [18], the Total Variation Diminishing (TVD) scheme of Harten [19] and the Monotonic Upstream-centered Scheme for Conservation Laws (MUSCL) of van Leer [21]. In recent years with the extension of upwind scheme to unstructured grids, more interest is focussing on how to reconstruct the high order scheme on such grids. The initial thinking is to apply the same idea as that used on structured grids directly. Some researchers have done so. Of them Batina [29] employed the MUSCL approach to reconstruct high order schemes on triangular grids when using an upwind cell-centred finite volume method, as did Frink [31] and Venkatakrishnan and Barth [32]. In Chapter 3 we have discussed the MUSCL approach and successfully applied it to supersonic and hypersonic flow computations. Alternatively, Barth and Jespersen [33] proposed a novel upwind scheme for the solution of the Euler equation on unstructured grids by departing from the one dimensional approach for satisfying the monotone principle. In multi-dimensions, they employed a monotonicity principle similar to van Leer [21] for structured grids, namely that the reconstructed distributions on the control vol-



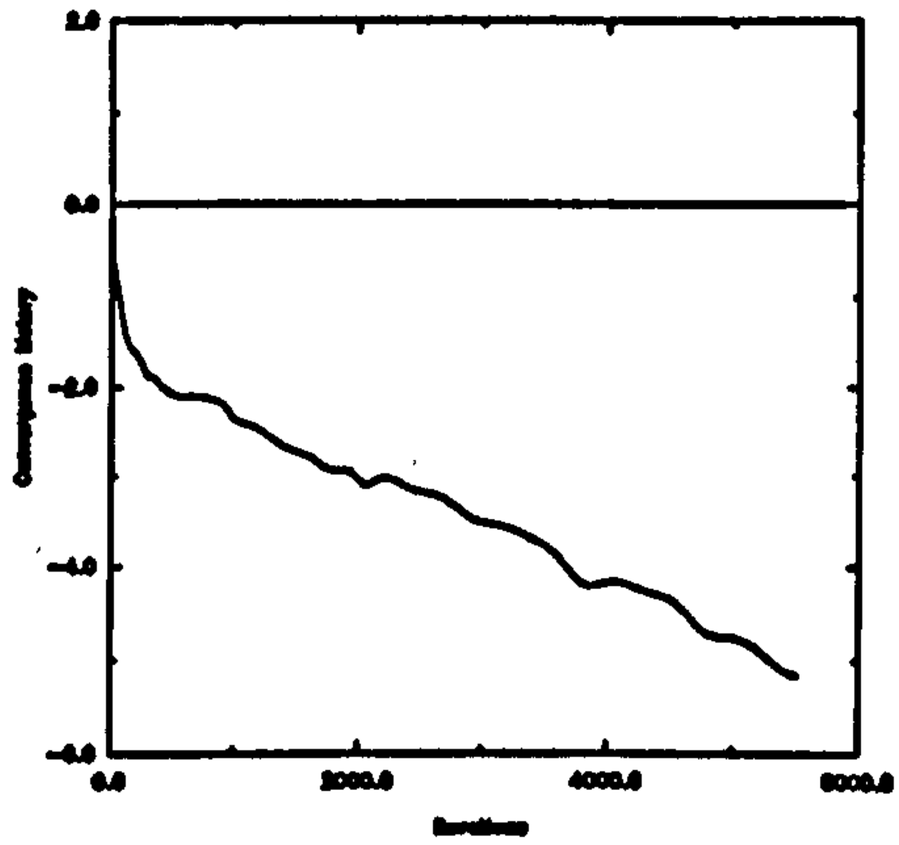


Fig. 23. Pressure distributions and convergence history of RAE2822 airfoil with the first order scheme

ume cannot extend over or under the values on all its neighbouring control volumes. To satisfy this, they introduced a min-mod type limiter to guarantee the smooth oscillation-free solution on unstructured grids. However many researchers found in their applications that the nondifferentiable limiter function such as the min-mod type limiter of Barth and Jespersen (B-J) generally leads to convergence difficulties. The B-J limiter had once been used in the calculation of supersonic corner flow in chapter 3. From the results in figure 11, it has also been found that the convergence stalls after two orders of reduction in the residual when using the variable recovery method (VAR) with the B-J limiter. Recently Venkatakrishnan [34] analysed this convergence problem theoretically and proposed a new limiter which has worked quite well in the test case considered. A modified version of the original van Albada limiter [84] was also discussed in [34]. Bishop and Noack [35] suggested the replacement of the min-mod type limiter function of B-J with a smooth second-degree polynomial function, as did Venkatakrishnan in [34], but further imposing an additional constraint condition of first derivative continuity. Rosendale [36] described another approach of applying a multi-dimensional generalized form of the van Albada limiter on triangular meshes and thought that it might be an inexpensive approach to the Essentially Non Oscilatory (ENO) scheme, which is still a very complex and high-cost scheme for unstructured grids. Besides the limiter, another problem connected with the high

order reconstruction procedure is the computation of gradients at the reference points of the control volume. Normally this is chosen as the centre points in the cell-centred scheme. Generally the Green-Gauss integral method is the common choice. Yet some researchers also found that the least-square reconstruction [39] based on the minimum energy principle is promising.

The methodology of flow solver employed in this chapter is the same as that described in Chapter 3. Section 4.2 describes a general form of the high order scheme on unstructured grids. Section 4.3 focusses on two gradient computation methods, i.e. the Green-Gauss integral and the least-square. In section 4.4 the limiters mentioned above will be described individually. The effects of gradient computation method on accuracy and convergence history, together with limiters is described in section 4.5 by the test case of the inviscid transonic flow over a supercritical airfoil RAE 2822. Finally the conclusions from this chapter are given in section 4.6.

4.2. General form of high order scheme on unstructured grid

In our point implicit iteration scheme described in the previous chapter the variables at the cell centre point of the control volume (element) are normally used as the first order approximation using the values at both sides of the interface to compute the fluxes across the interfaces in the Riemann solver. As we have shown in above section that this approach will produce unreasonable results by such approximation, the high order scheme needs to be considered.

In general the high order accuracy of variables over an arbitrary control volume can be achieved by the Taylor series expansion. In the 2D case with only second order approximations considered, such processes will result in

$$Q(x,y) = Q(x_o,y_o) + \nabla Q\mid_{x_o,y_o} \Delta\vec{r} + O\left(\Delta x^2, \Delta y^2\right) \tag{4.1}$$

where variable $Q(x,y)$ means the value of second order reconstruction, $Q(x_o,y_o)$ represents the value at the reference point (x_o,y_o), $\nabla Q\mid_{x_o,y_o}$ means the gradient at the reference point (x_o,y_o), $\Delta\vec{r} = \vec{r} - \vec{r_o}$ means the position vector of the point $\vec{r}(x,y)$ with respect to the reference point $\vec{r_o}(x_o,y_o)$ and $O(\Delta x^2, \Delta y^2)$ means the neglected higher order items.

Normally the result of the above linear reconstruction may exhibit nonphysical oscillations in the form of over- or under-shoots near the flow discontinuity regions. To prevent this a limiter Φ must be applied, which results in the final formula as

$$Q(x,y) = Q(x_o,y_o) + \Phi\nabla Q\mid_{x_o,y_o} \Delta\vec{r} + O\left(\Delta x^2, \Delta y^2\right) \tag{4.2}$$

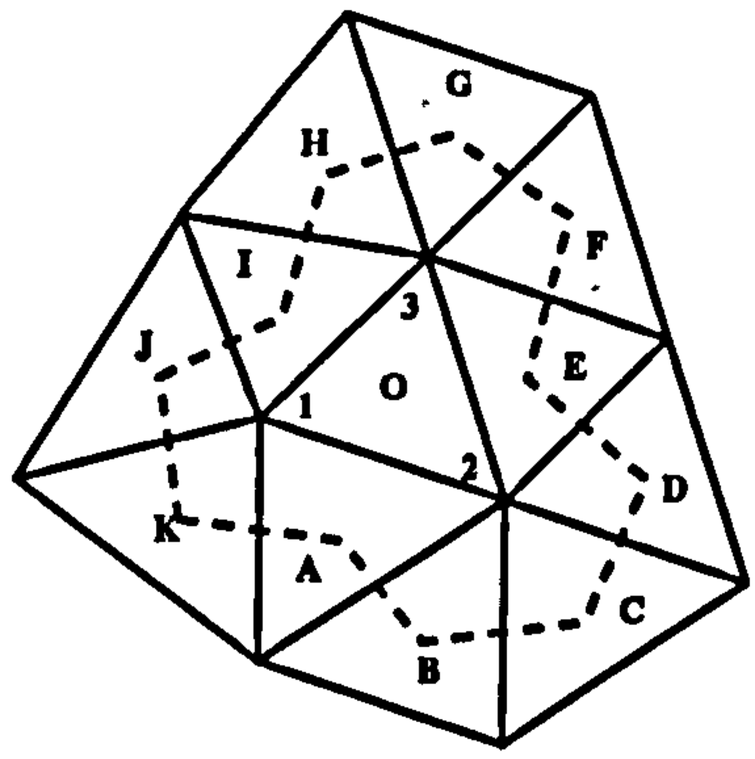


Fig. 24. Green-Gauss integral routine (A-B-···-K) used by Barth-Jespersen

Clearly equation (4.2) implies two aspects. One is calculation of the gradient at the reference points (e.g. the central point of the cell). Another one is how to define the limiter Φ. We will discuss them individually in the following sections.

4.3. Methods of gradient value computation

Two commonly used gradient value computation methods used for high order reconstruction on unstructured grids are the Green-Gauss (G-G) integral method and the least-square (L_2) method.

4.3.1. Green-Gauss integral method

The Green-Gauss integral method is based on the following formula

$$\iint_{\Omega} \nabla Q d\Omega = \oint_{\Gamma} \vec{Q}\vec{n} d\Gamma \tag{4.3}$$

In this case we assume the piecewise distribution of gradient ∇Q on Ω with the reference point (x_o, y_o) . Then we have

$$\nabla Q|_{x_o,y_o} = \frac{1}{\Omega} \oint_{\Gamma} \vec{Q}\vec{n} d\Gamma \tag{4.4}$$

In theory any closed routine around the reference point (x_o, y_o) can be selected as the integral path. In [33] Barth and Jespersen suggested a large route (see closed line of A-B-C-···-K in figure 24) as their integral path. It is clear that the more information considered to compute the gradient the more accuracy will be produced. However it will use more CPU time and memory. Thus there are balancing factors.

Here we consider all information in neighbouring elements in an attempt to keep the cost as low as possible. Similar to Frink [31] we first use the inverse-distance

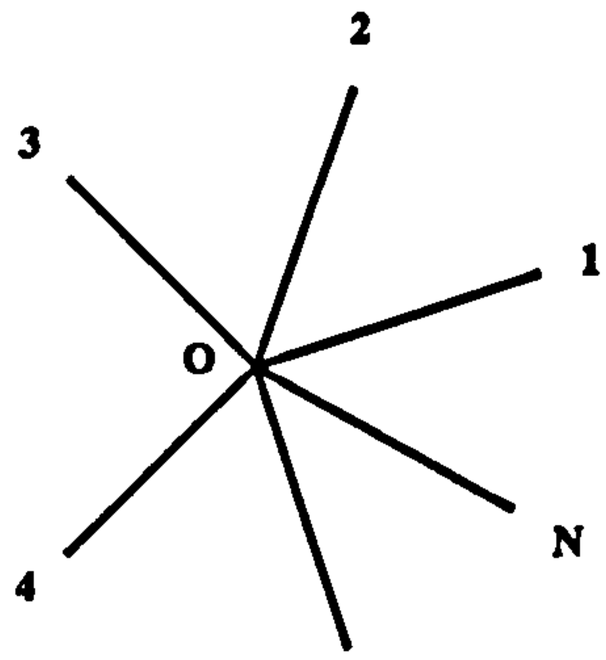


Fig. 25. Relations of current element and neighbouring elements used in least-square method

weighted method to compute the value at the node points with information from all neighbouring elements. After this the integral is evaluated using equation (4.4) along all edges of the current element. Taking the element "*o*" of figure 24 for example, we first find the values at its nodes 1,2 and 3 using

$$Q_i = \frac{\sum_{j=1}^{J(i)} Q_j/r_j}{\sum_{j=1}^{J(i)} 1/r_j} \qquad i = 1,2,3 \tag{4.5}$$

where $J(i)$ means all elements sharing the node i. Then the integral is evaluated by

$$\nabla Q|_{x_o,y_o} = \frac{1}{\Omega}\sum_{s_e=1}^{3} Q^*_{s_e}\delta s_e \tag{4.6}$$

$$Q^*_{s_e} = 0.5(Q_i + Q_{i+1}) \quad i = 1,2,3 \tag{4.7}$$

4.3.2. Least-square method

The least-square (L_2) method is another one commonly used to calculate the gradients, in which all neighbouring variables can be assumed to behave linearly. This method is based on the minimum-energy principle [87].

By applying equation (4.1) on all neighbouring elements (see figure 25 as an example) the variables at each cell-centre point of neighbouring elements may be expressed as

$$Q_i = Q_o + \frac{\partial Q}{\partial x}|_o \Delta x_i + \frac{\partial Q}{\partial y}|_o \Delta y_i \; (i = 1,2,\ldots,n) \tag{4.8}$$

It can also be written in an $N \times 2$ system of equations as

$$Ax = b \tag{4.9}$$

where

$$A = \begin{pmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \vdots & \vdots \\ \Delta x_n & \Delta y_n \end{pmatrix} \quad x = \begin{pmatrix} \frac{\partial Q}{\partial x} \\ \frac{\partial Q}{\partial y} \end{pmatrix} \quad b = \begin{pmatrix} \Delta Q_1 \\ \Delta Q_2 \\ \vdots \\ \Delta Q_n \end{pmatrix} \tag{4.10}$$

Clearly this is an overdetermined system of linear equations which can be solved to obtain the gradients at the cell-centered point of the current element. Here the Gram-Schmidt process [88] is used in which the system of equations is solved by decomposing the matrix A into a product of an orthogonal matrix Q and an upper triangular matrix R, i.e.

$$A = QR \tag{4.11}$$

so that the solution is obtained by

$$x = R^{-1}Q^T b \tag{4.12}$$

The resulting formulas for calculating the gradients at point (x_o, y_o) are given by

$$\frac{\partial Q}{\partial x}\,|_o = \sum_{i=1}^{N} w_i^x\,(Q_i - Q_o)\,; \quad \frac{\partial Q}{\partial y}\,|_o = \sum_{i=1}^{N} w_i^y\,(Q_i - Q_o) \tag{4.13}$$

where the summation is over all neighbouring elements and the weights are given by

$$w_i^x = \frac{x_i - x_o}{r_{11}^2} - \frac{r_{12}}{r_{11}r_{22}^2}\left[(y_i - y_o) - (x_i - x_o)\frac{r_{12}}{r_{11}}\right] \tag{4.14}$$

$$w_i^y = \frac{1}{r_{22}^2}\left[(y_i - y_o) - (x_i - x_o)\frac{r_{12}}{r_{11}}\right] \tag{4.15}$$

where

$$r_{11} = \left[\sum_{i=1}^{N}(x_i - x_o)^2\right]^{1/2} \tag{4.16}$$

$$r_{12} = \frac{\sum_{i=1}^{N}(x_i - x_o)(y_i - y_o)}{r_{11}} \tag{4.17}$$

$$r_{22} = \left[\sum_{i=1}^{N}(y_i - y_o)^2 - r_{12}^2\right]^{1/2} \tag{4.18}$$

It must be noted that all formulas above yield an unweighted least-square pro-

cedure in which all the variables of neighbours are considered equally.

4.4. Limiters

In the following subsection we will discuss the limiters mentioned in the introduction section to this chapter.

4.4.1. Barth-Jesperson limiter

The B-J limiter follows the monotonicity principle in multiple dimensions, in that the reconstructured distributions everywhere in the control volume must be bounded by the value of all the neighbours. This can be expressed mathematically as

$$\Phi_i = \begin{cases} 1 & \text{if } Q_i = Q_o \\ \min(1, \frac{Q^{max}-Q_o}{Q_i-Q_o}) & \text{if } Q_i > Q_o \\ \min(1, \frac{Q^{min}-Q_o}{Q_i-Q_o}) & \text{if } Q_i < Q_o \end{cases} \qquad (i = 1, 2, \ldots, n) \qquad (4.19)$$

where Q^{max} means the maximum value of all the neighbours, Q^{min} means the minimum value of all the neighbours and Q_i is the reconstructed value at node i of the current element by equation (4.1) without the limiter.

The final limiter of the element is defined as $\Phi = \min(\Phi_1, \Phi_2, \ldots, \Phi_n)$.

As the B-J limiter strictly follows the monotonicity principle it will produce smooth oscillation-free solutions even on very stretched unstructured grids. However as a min-mod limiter function is used it will lead to convergence difficulties. Several researchers reporting on use of the B-J limiter, noted that the convergence stalled after one or two orders decrease in residual even though the lift coefficient had already converged and reasonable pressure distributions achieved. This behaviour will also be illustrated in this work.

4.4.2. Venkatakrishnan limiter

Recently Venkatakrishnan analysed this convergence difficulty theoretically and proposed a new limiter (later we shall call it the Venkat limiter in short) which improves the convergence situation at some expense to monotonicity.

Insteady of using the min-mod function min(1,r), Venkatakrishnan uses the following second-degree polynomial function expressed as

$$\phi(r) = \frac{r^2 + 2r}{r^2 + r + 2} \qquad (4.20)$$

Based on this the Venkatakrishnan limiter can be written as

$$\Phi_i = \begin{cases} 1 & \text{if } Q_i = Q_o \\ \phi(\frac{Q^{max}-Q_o}{Q_i-Q_o}) & \text{if } Q_i > Q_o \\ \phi(\frac{Q^{min}-Q_o}{Q_i-Q_o}) & \text{if } Q_i < Q_o \end{cases} \qquad (i = 1, 2, \ldots, n) \tag{4.21}$$

where Q^{max}, Q^{min} and Q_i are defined as before. The limiter function ϕ is defined as above.

In order to undergo transition smoothly from the application of the limiter in regions of high gradient to the removal of it in near-constant regions of the solution, a modification must be made on the limiter function resulting in

$$\phi(\frac{\Delta_+}{\Delta_-}) = \frac{1}{\Delta_-}\left[\frac{(\Delta_+^2 + \epsilon^2)\Delta_- + 2\Delta_-^2\Delta_+}{\Delta_+^2 + 2\Delta_-^2 + \Delta_-\Delta_+ + \epsilon^2}\right] \tag{4.22}$$

where $\Delta_+ = Q^{max} - Q_o$ if $Q_i > Q_o$ or $\Delta_+ = Q^{min} - Q_o$ if $Q_i < Q_o$. $\Delta_- = Q_i - Q_o$. ϵ^2 is taken to be $(k\Delta h)^3$.

Here k is a constant. A value of zero for k implies that the limiter is still active even in the near-constant regions, whereas a very large value for k implies effectively no limiter at all. Δh is taken as the distance between a node point and the centre point of the control volume. In practical application Δ_- should be replaced by $\text{sign}(\Delta_-)(|\Delta_-| + 10^{-12})$ to avoid division by a very small value of Δ_-. The value of k depends on the test case and flow conditions. It can be defined through tests.

4.4.3. van Albada limiter

The original van Albada limiter takes the same definition of equation (4.21) but including the following form of the limiter function ϕ

$$\phi(r) = \frac{r^2 + r}{r^2 + 1} \tag{4.23}$$

where $r = \Delta_+/\Delta_-$, $\Delta_+ = Q^{max}(or Q^{min}) - Q_o$, $\Delta_- = Q_i - Q_o$

Also in order to avoid clipping smooth extrema the following modification should be made

$$\phi(\frac{\Delta_+}{\Delta_-}) = \frac{1}{\Delta_-}\left[\frac{(\Delta_+^2 + \epsilon^2)\Delta_- + (\Delta_-^2 + \epsilon^2)\Delta_+}{\Delta_+^2 + \Delta_-^2 + 2\epsilon^2}\right] \tag{4.24}$$

where ϵ^2 is defined the same as that in the Venkat limiter. And Δ_- is also replaced by $\text{sign}(\Delta_-)(|\Delta_-| + 10^{-12})$ to avoid the zero division.

4.4.4. Bishop-Noack limiter

Firstly the B-N limiter uses a smooth approximation that is a ratio of the second-degree polynomials instead of the min(1,r) function. The resulting limiter function must satify the conditions: $\phi(0) = 0$, $\phi(2) = 1$ (which is the second-order accuracy constraint) and $\phi(\infty) = 1$ with the constraints $\phi(r) \leq min(1, r)$ for $0 \leq r < 2$ and $\phi(r) \geq 1$ for $r \geq 2$.

The form of limiter function must be taken as

$$\phi(r) = \frac{r^2 + 2r}{r^2 + r + 2} \tag{4.25}$$

which is the same as that adopted by Venkatakrishnan.

Secondly in the B-N limiter, an additional constraint in the first derivative of continuity, namely $\phi'(2) = 0$, is imposed. The limiter function that satisfies all these conditions is

$$\phi(r) = \frac{4s}{s^2 + 4} \qquad \text{with } s = \min(2, r) \tag{4.26}$$

Finally to de-activate the limiter in near-constant regions requires the following modification

$$\phi(\frac{\Delta_+}{\Delta_-}) = \frac{2\tilde{\Delta}_+\tilde{\Delta}_- + \tilde{\epsilon}^2}{\tilde{\Delta}_+^2 + \tilde{\Delta}_-^2 + \tilde{\epsilon}^2} \tag{4.27}$$

in which $\tilde{\Delta}_- = 2\Delta_-$, $\tilde{\Delta}_+ = \min(2\Delta_-, \Delta_+)$ and $\tilde{\epsilon}^2 = \alpha^4 \Delta h^3$ applied to avoid application of the limiter in near zero gradient regions. The grid spacing Δh represents the distance between the cell centre and the node points. Then $\alpha \geq \epsilon$ is used to avoid division by zero where ϵ is a small number.

4.4.5. General van Albada limiter

By re-writing the generalized form of the van Albada limiter on triangular meshes Rosendale derived the following formula to evaluate the gradients

$$\nabla Q|_{x_o, y_o} = w_1 \nabla Q_1^n + w_2 \nabla Q_2^n + w_3 \nabla Q_3^n \tag{4.28}$$

where ∇Q_1^n, ∇Q_2^n and ∇Q_3^n are the gradients at node points 1,2,3 of the element (see figure 24). Suppose it is already computed by the method described in the previous section. Suitable weights are w_1, w_2, w_3, which obey

$$w_1 + w_2 + w_3 = 1 \tag{4.29}$$

$$w_1, w_2, w_3 \in [0, 1] \tag{4.30}$$

To preserve the original van Albada properties of not clipping extrema, the particular choice may be

$$w_1 = \frac{a_2 a_3 + \epsilon^2}{a_1 a_2 + a_2 a_3 + a_3 a_1 + 3\epsilon^2} \tag{4.31}$$

$$w_2 = \frac{a_3 a_1 + \epsilon^2}{a_1 a_2 + a_2 a_3 + a_3 a_1 + 3\epsilon^2} \tag{4.32}$$

$$w_3 = \frac{a_1 a_2 + \epsilon^2}{a_1 a_2 + a_2 a_3 + a_3 a_1 + 3\epsilon^2} \tag{4.33}$$

where $a_1 = \| \nabla Q_1^n \|^2$; $a_2 = \| \nabla Q_2^n \|^2$; $a_3 = \| \nabla Q_3^n \|^2$, and ϵ is a small number.

4.5. Comparisons

In this section all limiters and gradient computation methods described above will be compared using the test case of the transonic inviscid flow over the RAE 2822 aerofoil.

The mesh employed is a regular unstructured mesh produced by halving the structured mesh (129×33) (figure 26). The airfoil is in a flow of Mach number 0.729 and incidence of 2.31°. The normal spacing of the first grid near the wall takes 10^{-3} chords which is sufficient for inviscid flow simulation.

4.5.1. Comparison of the computation of the gradient

Here we present the comparison of methods for calculating gradient. The limiters considered are confined to those of B-J and Venkat.

Figure 27 shows very good agreements in pressure distributions using the B-J and Venkat limiters with both the Green-Gauss and the least-square methods. Small differences are found near the upper surface of leading edges illustrated by the detailed picture.

Figure 28 illustrates the convergence history of the residual. It is found using the B-J limiter with both G-G and L_2 methods the residuals are stalled after nearly two orders reduction in residual. However the residual is reduced to -3 by the G-G method and even -4.2 by the L_2 method when the Venkatakrishnan limiter used. It improves as the iteration continues.

Figures 29 give the history of the lift coefficients. Both the G-G and the L_2 method give nearly the same result but small differences are also found in the detailed C_l picture. The G-G method results in a high value of 0.88 while the L_2 method results

in a low value of 0.865.

4.5.2. Comparison of limiters

Comparison is made in this section between first and high order schemes and also with the high order scheme with different limiters.

From figures 30, 31 and 32 we can find that (1) pressure distributions are more reasonable when using the high order scheme rather than the first order scheme; (2) the residual reduces rapidly when the first order scheme is used, while with the use of the high order scheme it is found more difficult to converge the residual; (3) a higher lift coefficient is obtained using the high order scheme compared with the first order scheme.

Figure 30 illustrates the comparison of the pressure distribution for the first order scheme and the high order schemes with limiters. All the high order schemes gave nearly the same C_p results except in the shock wave regions where the limiter is active. Different limiters result in different values of C_p in the shock wave region. Without a limiter strong oscillations appear. Use of limiters improves this situation.

Figure 31 compares the convergence history of the first order scheme with the high order one with different limiters. With the first order scheme, the residual decreases rapidly below -5 order, while the high order scheme results in convergence difficulties. From the detailed picture it is found that: (1) without the limiter the residual can be reduced to near -3.0 and will be reduced further if the iteration continues; (2) with the B-J limiter the residual oscillates around the value of -2.2 and appears not to decrease further; (3) the Venkatakrishnan limiter (with k=5 used in present test case) results in a residual of -3.2 in the same number of iterations and decrease further as the iteration continues; (4) the residual using the van Albada limiter (with k=5 used) also stalled at -2.5, with only a little improvement using the B-J limiter; (5) the B-N limiter did not work well in present test case, the residual with its limiter only reaches -1.8 and does not decrease further; (6) the general van Albada limiter results in nearly the same convergence result as that using the B-J limiter.

Although in respective references all authors published very successful application of their limiters on the test case they selected, the results by the author did not give the promised favourable result, especially with regard to convergence history. Only the Venkatakrishnan limiter, in constrast to others, appears favourable.

Figures 32 gives the lift coefficient history. By using the first order scheme the lift coefficient (C_l) is nearly 20% lower than that using the high order scheme. The high order scheme with different limters gives nearly the same value of C_l but small difference still remain when compared in detail (right picture). The B-J limiter,

Venkat limiter and van Albada limiter result in the higher C_l value of nearly 0.88. Without the limiter the C_l value only reaches 0.875. Both the B-N and the general van Albada limiters give the value of 0.86. In constrast the first order scheme results in only a value of C_l of 0.71.

It should be mentioned in all the above computations, the gradient value is calculated by the Green-Gauss method.

4.6. Conclusions

The following conclusions can be made through above researches:

(1) Generally the high order scheme should be used in the flow solver, otherwise unreasonable results are produced;

(2) With the limiters considered in this chapter, a similar pressure distribution is obtained except in the shock wave region where the limiter is active. Converged results of lift coefficient are achieved with little difference between methods;

(3) The residual is very sensitive to the limiter used. Most limiters considered resulted in difficulty of convergence. Only the Venkatakrishnan limiter (with k=5 used) is superior to others resulting a residual decreasing below -3 for a similar number of iterations;

(4) The selection of the computation of gradient has little influence on the value of C_p and C_l but more influence on the convergence history. The L_2 method seems better than the G-G method in the present test case, by which the residual can be decreased below -4 when using the Venkatakrishnan limiter;

(5) The quality of limiter is problem-dependent. In our experience some limiters did not show as good a performance as in their original publications.

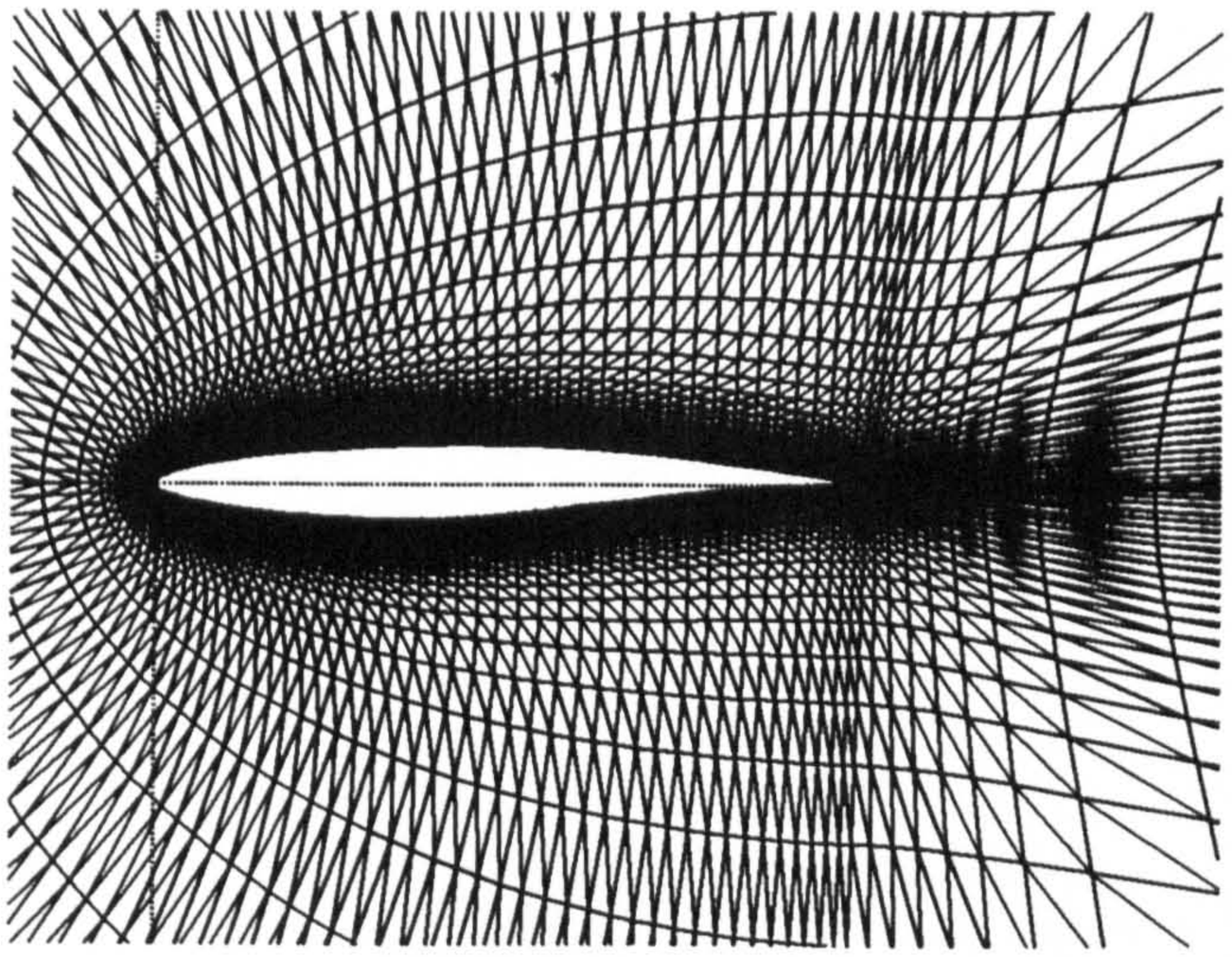


Fig. 26. Regular unstructured grids around RAE2822 airfoil

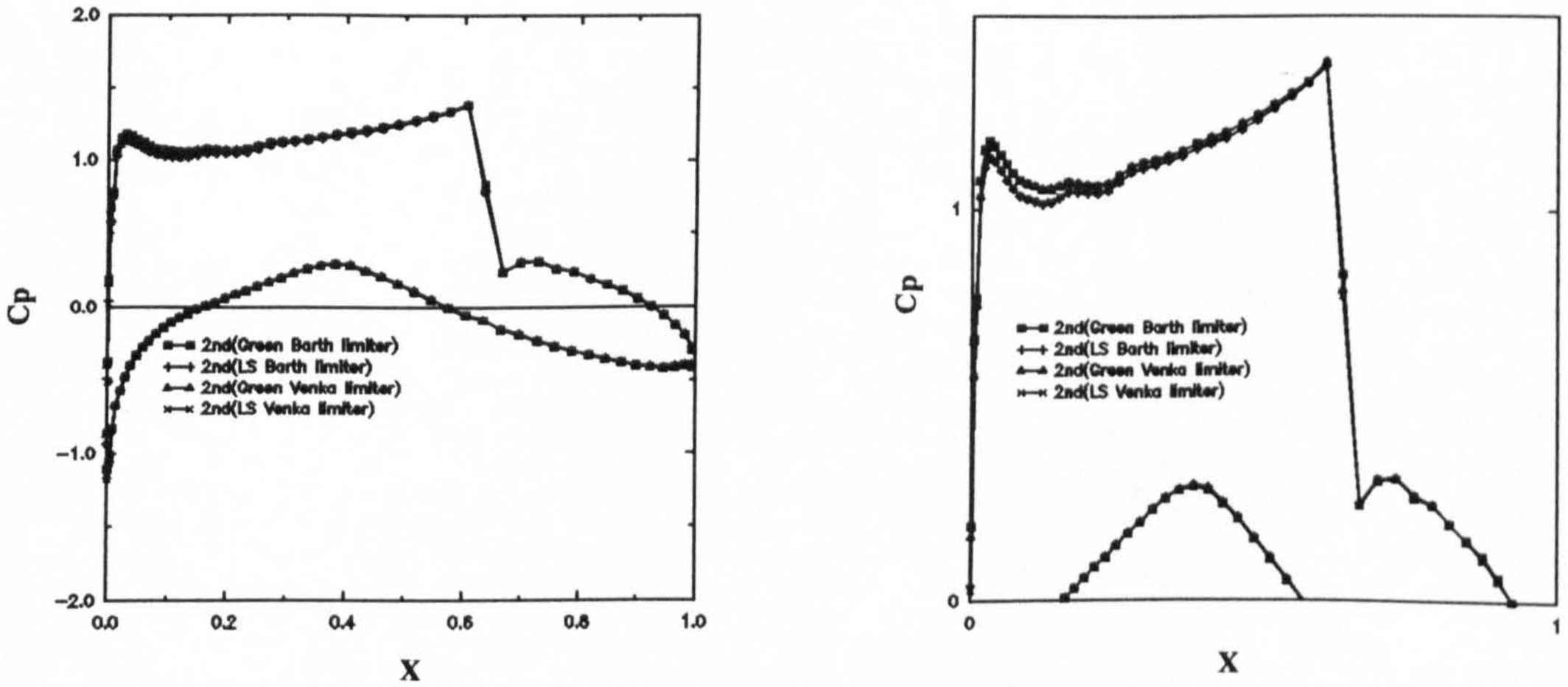


Fig. 27. Comparison of pressure distributions of RAE2822 airfoil with different gradient computation methods

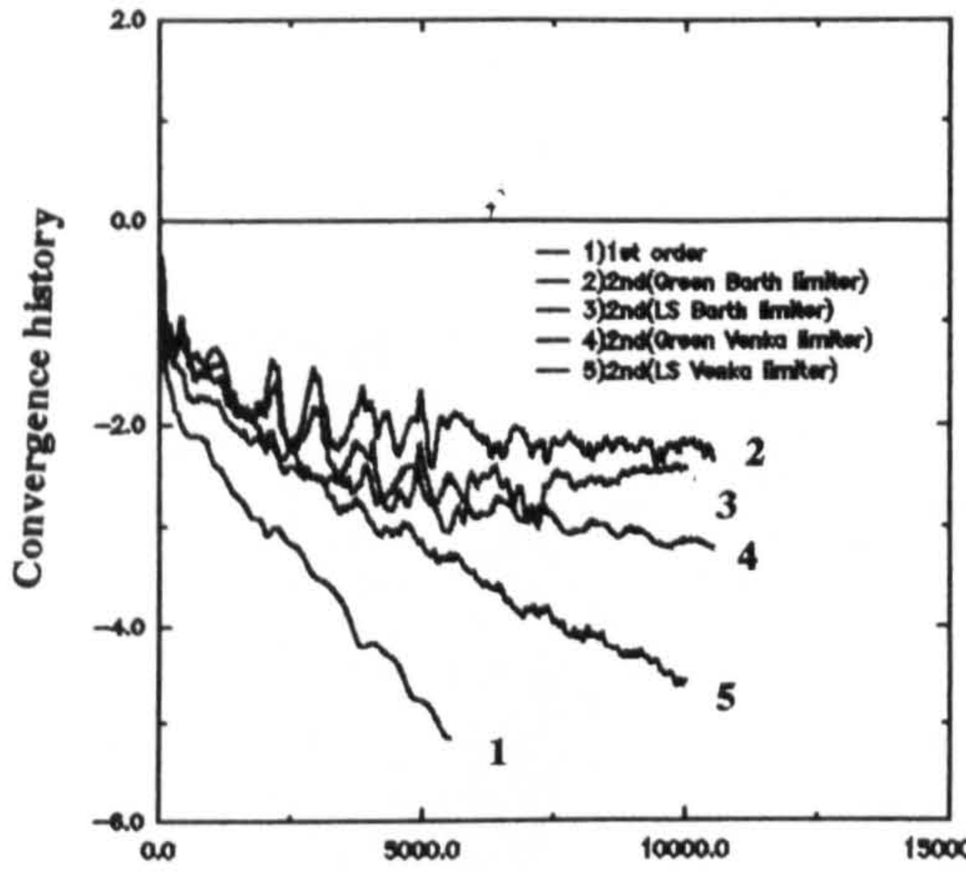


Fig. 28. Comparison of convergence history of RAE2822 airfoil with different gradient computation methods

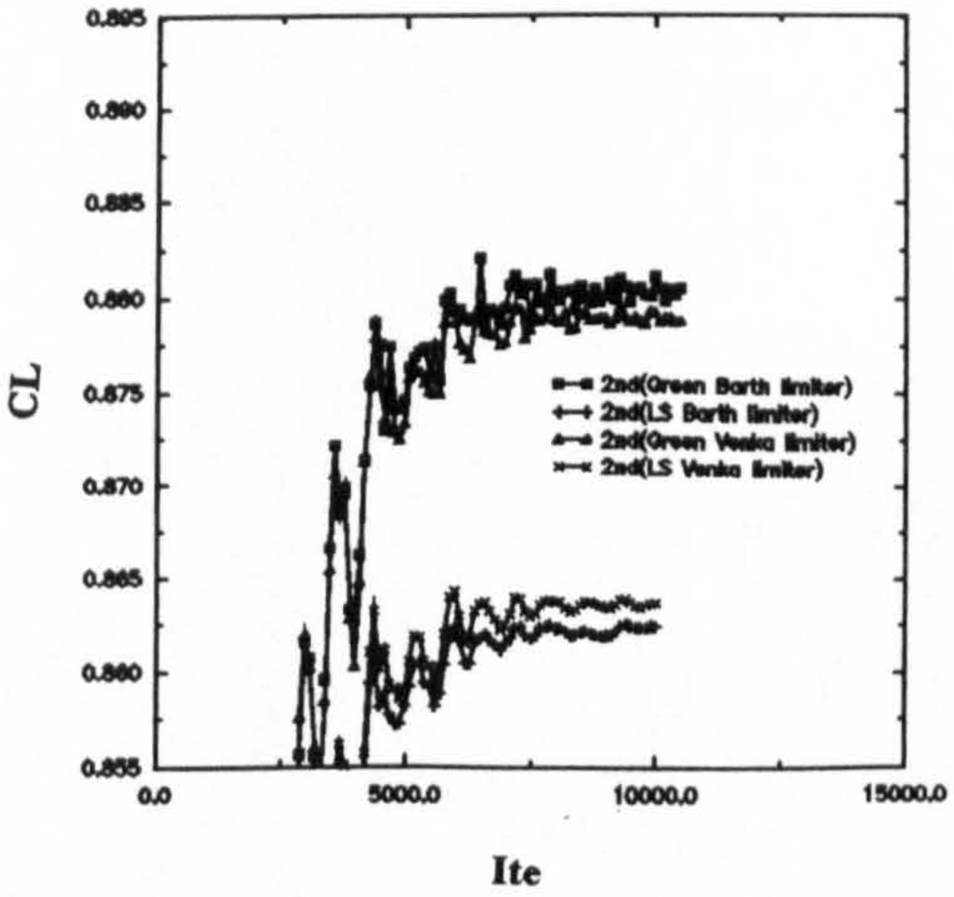


Fig. 29. Comparison of lift coefficient history of RAE2822 airfoil with gradient computation methods

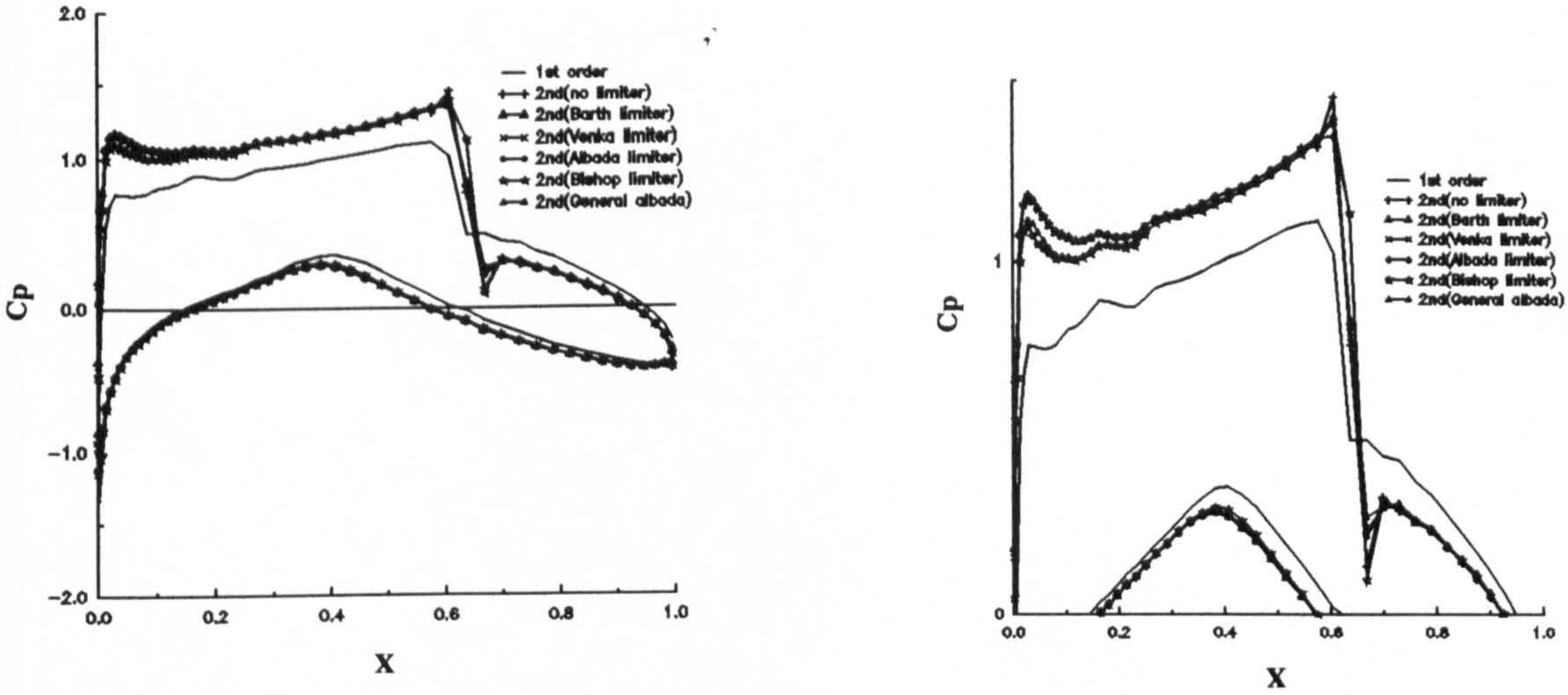


Fig. 30. Comparison of pressure distributions of RAE2822 airfoil with limiters

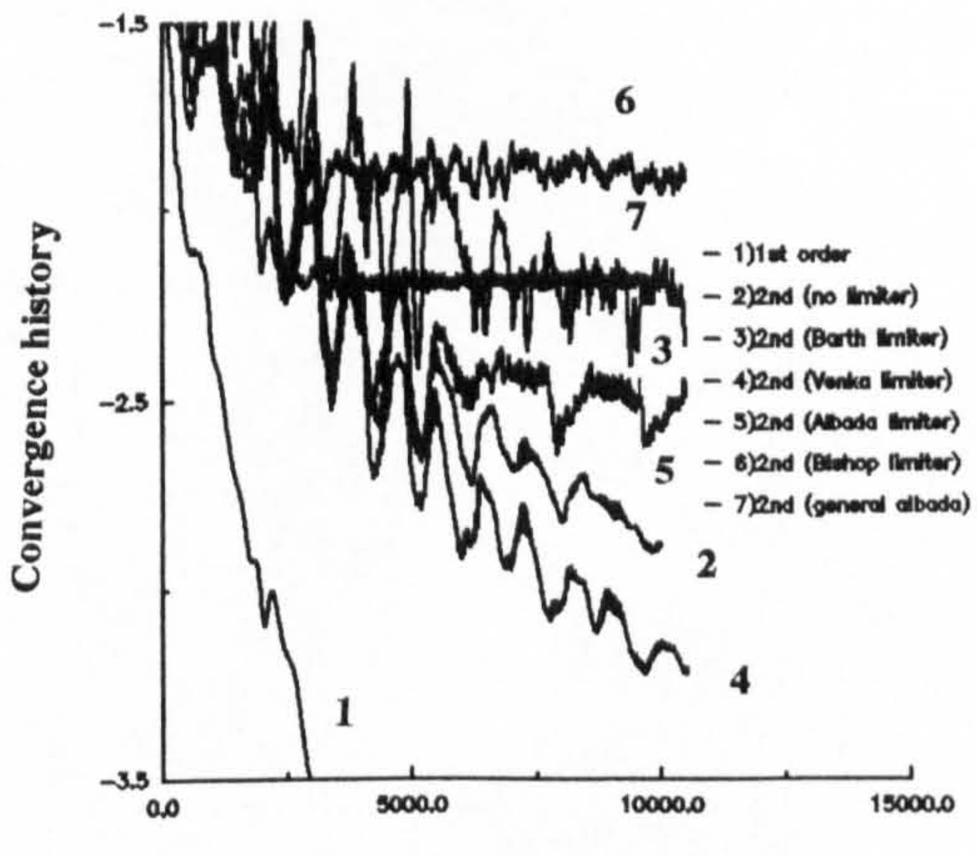


Fig. 31. Comparison of convergence history of RAE2822 airfoil with limiters

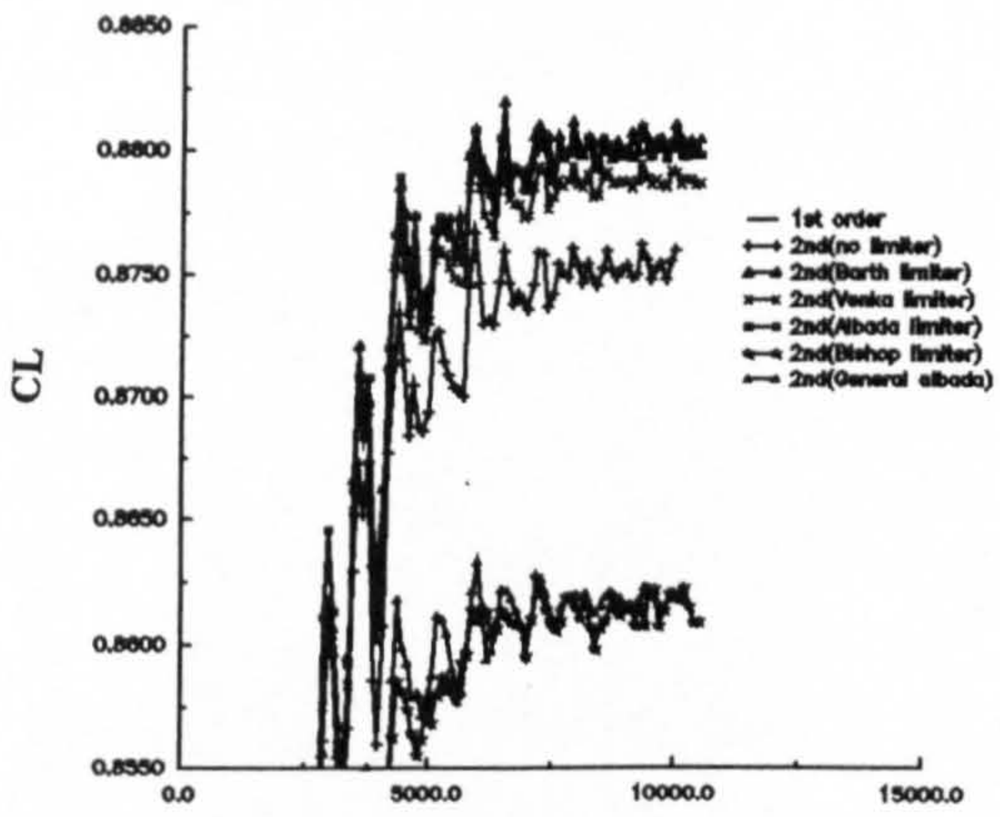


Fig. 32. Comparison of lift coefficient history of RAE2822 airfoil with limiters

CHAPTER 5

NUMERICAL SOLUTION OF TWO-DIMENSIONAL NAVIER-STOKES EQUATION

5.1. Introduction

Compared to the solution of the Euler equations, the solution of the Navier-Stokes equations has additional difficulties when using unstructured grids. The problem is the simulation of boundary layer flow, where the variables in the normal direction change rapidly compared to those in the tangential direction. The present situation is that the methods most used to generate unstructured grids, e.g. Delauney triangulation and the advancing front technique, by their nature, do not lend themselves to generating the highly stretched elements which are required in viscous flow. However in most approaches on structured grids this problem can be overcome using cells with aspect ratios typically of the order of thousands. It would appear that, similar to structured grids, suitable unstructured grids used in the simulation of viscous flows would have to include elements with very high aspect ratios.

In the application of unstructured grids to viscous flow simulation, most of the effort is concentrated on the construction of unstructured grids with a suitably stretched grid distribution in the boundary layer flow region and the development of flow solution algorithms capable of producing accurate results. Investigations have been carried out in developing such techniques for using unstructured grid for viscous flow. Three categories of recent approaches for mesh construction can be identified. The first category is the technique which attempts to modify a generated grid by element transformation and use of an intermediate mapping space. Mavriplis [54] [55] proposed the generation of unstructured triangular meshes based on a Delaunay triangulation, which is performed in a locally stretched space in order to obtain high-aspect-ratio triangles in the boundary layer and wake regions. Vilsmeier and Hanel [56] have also considered this approach and applied an intermediate mapping procedure to obtain strongly anisotropic grids. Although they are successful in the 2D case the extension of such an approach in 3D cases has not proved successful for general complex configurations. The second technique is to construct viscous grids using an intermediate stage which involves a modified form of standard unstructured grid generation. In the paper of Weatherill et al [59] two methods, named "Node Attraction" and "Advancing Normals", were described in detail. Both are designed to construct a viscous mesh based on an unstructured mesh generated by the Delauney triangula-

tion or advancing front techniques. Structured grids in near-wall regions are formed by pulling nodes close to a solid wall and re-ranking them. A similar approach is also advocated by Pirzadeh [58] using the advancing layer concept to generate the structured grid near the wall and unstructured grid in the far-field. The third approach is to create a "skin" around a shape within which the structured grid is developed as in Holmes and Connell [60]. Alternatively as in our report [89], a method of such a "skin" strategy is proposed to construct a hybrid mesh with a structured grid in the near-wall regions and an unstructured one in other regions. The difference is that the structured mesh is firstly constructed using an appropriate efficient and reliable code (e.g. the EAGLE grid generator [90]). Then an unstructured mesh is generated outside the "skin" boundary using the advancing front technique [50], and finally an intermediate layer is used to connect the two types of meshes. It should be pointed out that a structured mesh, embedded within a globally unstructured mesh, will primarily ease the viscous flow computation, especially in the implemention of the turbulence model (e.g. the Baldwin-Lomax algebraic model), without any need of an additional reference structured grid as mentioned in [54] [55].

Based on the successful work on the Euler flow solver outlined in the last chapter and the hybrid grid generation method of [89], this chapter deals with the development of a Navier-Stokes flow solver. In a similar fashion to the Euler flow solver, a cell-centred finite-volume method is used to discretize the Navier-Stokes equations. Roe's approximate Riemann solver is applied for the inviscid flux computations at each cell interface, assuming a local 1-D Riemann problem in its normal direction. A linear reconstruction using gradient estimates and limiters is employed to obtain two high order Riemann states at the sides of the cell interface. For viscous fluxes a finite volume formulation for the gradient at each cell interface, which can be regarded as a second order central scheme, is developed. Therefore the difference between the first and high order schemes lies in the discretization of the inviscid terms. An implicit point Gauss-Seidel scheme with Roe's approximate Riemann solver is used for the time integration. The test case of a laminar viscous flow over NACA 0012 airfoil is chosen first to validate the proposed Navier-Stokes solver on several kinds of grid topologies including the hybrid grid. Then the Baldwin-Lomax algebraic turbulence model is implemented within the NS flow solver and the computational procedure of predicting the high-Reynolds number turbulent flow over RAE 2822 airfoil on the hybrid grid is considered.

Section 5.2 gives a brief description of the hybrid grid generation technique. Section 5.3 gives the finite volume descritization of the Navier-Stokes equation. Section 5.4 describes the central-difference scheme for viscous terms. Section 5.5 illustrates

the iteration procedure. Section 5.6 deals with boundary conditions for Navier-Stokes equations. Section 5.7 gives the results of NS flow solver on the laminar test case. Section 5.8 discusses the Baldwin-Lomax algebraic turbulence model and its implementation on hybrid grids with the NS flow solver. Section 5.9 gives the results of turbulence flow of NACA 0012 and RAE 2822 airfoils. Conclusions for this chapter are given in section 5.10.

5.2. Hybrid viscous grid generation

It has been mentioned in the introductory section to this chapter of the difficulties appearing when dealing with viscous flow on a purely unstructured grid and that structured grids have particular advantages in dealing with viscous flow. It is then believed that the best approach is to construct a grid with the combination of both structured and unstructured grids thus retaining the advantages of both methods and suppressing the disadvantages. In reference [89] we proposed an efficient strategy, called the "skin" method, to construct a hybrid grid based on both structured and unstructured generators. The idea is simple and straightforward. Firstly the structured grid with high aspect ratio grids is generated in the near-wall region where viscous flow regions are expected in order to fulfill the requirements of viscous flow computation. This will also ease the implementation of viscous flow computations, particularly for turbulent flow cases. Secondly an irregular unstructured grid is constructed in the far-wall region, where some complex flow phenomena, such as shock waves, might appear. For such regions with high gradient values techniques such as mesh adaption need be considered. For unstructured grids, adaption is easy to implement. Thirdly two types of grids are merged on their interface resulting in an unified hybrid grid. Reference [89] describes the procedures of the hybrid grid generation method and gives some results of Euler solver validation. The same methodology will be used here to deal with the viscous flow cases.

5.3. Navier-Stokes equation in integral form and finite volume discretization

The procedure, described below, is applicable for general unstructured grids composed of either triangular or quadrilateral elements or a combination of both. Similar to that described in section 3.2, considered on a single control volume Ω_e, equation (2.18) can be written in its integral form as

$$\iint_{\Omega_e} \frac{\partial U}{\partial t} d\Omega + \iint_{\Omega_e} \left(\frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} \right) d\Omega = \iint_{\Omega_e} \left(\frac{\partial G_1}{\partial x} + \frac{\partial G_2}{\partial y} \right) d\Omega \qquad (5.1)$$

By using the divergence theorem, we have

$$\iint_{\Omega_e} \frac{\partial U}{\partial t} d\Omega + \oint_{\Gamma_e} F_n d\Gamma = \oint_{\Gamma_e} G_n d\Gamma \tag{5.2}$$

where Ω_e, Γ_e and F_n are defined as before. Viscous flux is defined as $G_n = \vec{G}_i \cdot \vec{n}_i$, in which $\vec{G}_i = (G_1, G_2)$ denotes the viscous vector fluxes and $\vec{n}_i = (n_x, n_y)$ denotes the unit vector outward normal to the boundary Γ_e of control volume Ω_e.

In a cell-centred discretization we assume a piecewise constant distribution of the unknowns U_e on each element Ω_e, so the discretized form of the above integral equation can be approximated as

$$\frac{\Delta U_e}{\Delta t_e} \Omega_e + F^I = G^V \tag{5.3}$$

where symbols are defined as before except those specifically noted and G^V denotes the viscous contribution as

$$G^V = \sum_{s_e} \tilde{G}_n \delta s_e \tag{5.4}$$

where $\tilde{G}_n$ denotes the approximate viscous fluxes through the side s_e shared by elements "e" and "r".

5.4. Central-difference scheme for the viscous terms

Differing from the inviscid terms, which are discretized using the upwinding scheme, the viscous terms are discretized by a central-difference type scheme.

The definition of the viscous terms is given in equation (2.21), for a cell side "s", the numerical viscous flux is calculated using the average value of the variables in the left element "e" and right element "r" , that is

$$u_s = 0.5(u_e + u_r) \tag{5.5}$$

$$v_s = 0.5(v_e + v_r) \tag{5.6}$$

$$T_s = 0.5(T_e + T_r) \tag{5.7}$$

The required values for μ is obtained by using T_s in equation (2.15). Hence the

viscous contributions at side "s" are

$$G_1^V = \begin{pmatrix} 0 \\ (\tau_{xx})_s \\ (\tau_{xy})_s \\ u_s(\tau_{xx})_s + v_s(\tau_{xy})_s - (q_x)_s \end{pmatrix} \tag{5.8}$$

$$G_2^V = \begin{pmatrix} 0 \\ (\tau_{yx})_s \\ (\tau_{yy})_s \\ u_s(\tau_{yx})_s + v_s(\tau_{yy})_s - (q_y)_s \end{pmatrix} \tag{5.9}$$

The normal viscous fluxes with respect to a side with outward normal vector are therefore written as

$$(G^V(1))_n = 0 \tag{5.10}$$

$$(G^V(2))_n = (\tau_{xx})_s n_x + (\tau_{xy})_s n_y \tag{5.11}$$

$$(G^V(3))_n = (\tau_{yx})_s n_x + (\tau_{yy})_s n_y \tag{5.12}$$

$$(G^V(4))_n = [u_s(\tau_{xx})_s + v_s(\tau_{xy})_s - (q_x)_s]n_x + [u_s(\tau_{yx})_s + v_s(\tau_{yy})_s - (q_y)_s]n_y \tag{5.13}$$

where subscripts denote an evaluation at the sides.

Referring to equations (2.22), (2.23), (2.24), (2.25), (2.26) and (2.27) the viscous stresses are determined from

$$(\tau_{xx})_s = \frac{1}{Re_\infty}[2\mu_s(\frac{\partial u}{\partial x})_s - \frac{2}{3}\mu_s((\frac{\partial u}{\partial x})_s + (\frac{\partial v}{\partial y})_s)] \tag{5.14}$$

$$(\tau_{xy})_s = \frac{1}{Re_\infty}[\mu_s((\frac{\partial u}{\partial y})_s + (\frac{\partial v}{\partial x})_s)] \tag{5.15}$$

$$(\tau_{yx})_s = (\tau_{xy})_s \tag{5.16}$$

$$(\tau_{yy})_s = \frac{1}{Re_\infty}[2\mu_s(\frac{\partial v}{\partial y})_s - \frac{2}{3}\mu_s((\frac{\partial u}{\partial x})_s + (\frac{\partial v}{\partial y})_s)] \tag{5.17}$$

and the heat fluxes are

$$(q_x)_s = -\frac{\mu_s}{(\gamma-1)M_\infty^2 Re_\infty P_r}(\frac{\partial T}{\partial x})_s \tag{5.18}$$

$$(q_y)_s = -\frac{\mu_s}{(\gamma-1)M_\infty^2 Re_\infty P_r}(\frac{\partial T}{\partial y})_s \tag{5.19}$$

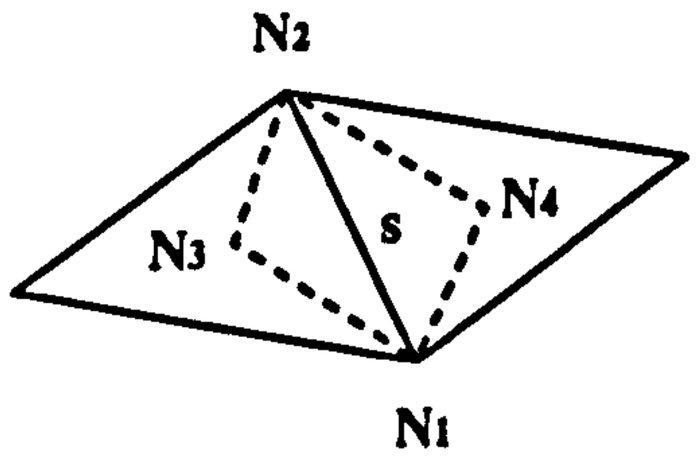


Fig. 33. Integration path for Green's theorem

It is clear that the evaluation of the viscous contributions requires a knowledge of the first derivatives of quantities, such as the velocity components (u,v) and the temperature T.

5.4.1. Calculation of first derivatives

(1) Arithematic average

The first derivatives can be obtained by the Green-Gauss integral formula along the path including side "s" (figure 33).

Based on Green's theorem, one can obtain the gradient from

$$\iint_{\Omega} \frac{\partial f}{\partial x} d\Omega = \oint_{\Gamma} f n_x d\Gamma \tag{5.20}$$

(Here we take a scalar variable f with respect to x at an element side "s" for example.)

Assuming a constant distribution of the gradient over this domain ($N_1 \rightarrow N_4 \rightarrow N_2 \rightarrow N_3 \rightarrow N_1$), the left side becomes $\left(\frac{\partial f}{\partial x}\right)_s \Omega$ and Ω denotes the area of the integral domain.

The right side integration is evaluated along the path "s" , which can be represented as 4- subpaths. Assuming each variable is constant along each subpath and is replaced by an average value, i.e.

$$\begin{aligned} \oint_{\Gamma} f n_x d\Gamma \;=\; & \int_{N_1}^{N_4} (f n_x)_{14} ds + \int_{N_4}^{N_2} (f n_x)_{42} ds \\ + \; & \int_{N_2}^{N_3} (f n_x)_{23} ds + \int_{N_3}^{N_1} (f n_x)_{31} ds \end{aligned} \tag{5.21}$$

By assuming $f_{ij} = 0.5(f_i + f_j)$, thus the gradient $\partial/\partial x$ is completely determined by writing

$$\begin{aligned} \left(\frac{\partial f}{\partial x}\right)_s = \frac{1}{2\Omega} \quad [\quad & (f_1 + f_4)(y_4 - y_1) + (f_4 + f_2)(y_2 - y_4) \\ + \quad & (f_2 + f_3)(y_3 - y_2) + (f_3 + f_1)(y_1 - y_3)] \end{aligned} \tag{5.22}$$

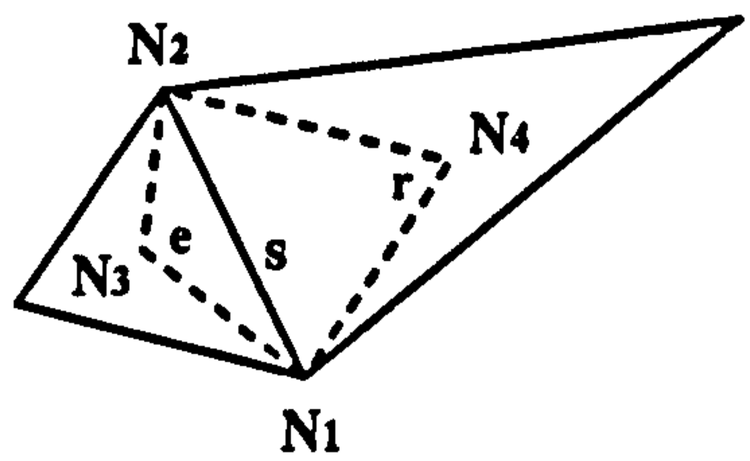


Fig. 34. Notations for Weighted Average Method

For the gradient $\partial/\partial y$ using a similar procedure, we have

$$\left(\frac{\partial f}{\partial y}\right)_s = \frac{1}{2\Omega} \quad [\quad (f_1+f_4)(x_1-x_4)+(f_4+f_2)(x_4-x_2)$$
$$+ \quad (f_2+f_3)(x_2-x_3)+(f_3+f_1)(x_3-x_1)] \qquad (5.23)$$

In the above expression, it requires the knowledge of the value of variables at node points N_3 and N_4. Simply from averaging the flow variables in all elements surrounding the node, results in

$$f_i = \frac{1}{idim(Node)} \sum_{k=1}^{idim(Node)} f_k \qquad (5.24)$$

A more accurate estimation of the node value through a weighted average is described by equation (3.87) in chapter 3.

Unfortunately, for a strongly stretched grid, in a region where the area of an element changes suddenly, there will arise errors when using the above formulation. Hence we have attempted to use the weighted average method below instead of the arithematic average method used above.

(2) Weighted average

A more accurate estimate of the gradient at side "s" can be obtained by a weighted average (see figure 34) as described by

$$\nabla f_s = w_e \nabla f_e + w_r \nabla f_r \qquad (5.25)$$

where f represents the physical quantity (i.e. u, v or T); and w_e is the area ratio of the right triangle ($N_1 \rightarrow N_4 \rightarrow N_2$) to the quadrilateral ($N_1 \rightarrow N_4 \rightarrow N_2 \rightarrow N_3$), and w_r is the area ratio of the left triangle ($N_1 \rightarrow N_2 \rightarrow N_3$) to the quadrilateral ($N_1 \rightarrow N_4 \rightarrow N_2 \rightarrow N_3$), ∇f_e and ∇f_r can be evaluated separately by the Green's theorem along its integral path, i.e. for the left side integral path this is ($N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N1$), and for right side integral path this is ($N_1 \rightarrow N_4 \rightarrow N_2 \rightarrow N_1$).

5.5. Iteration algorithms

5.5.1. Explicit scheme

In a similar fashion to that described in Chapter 3 an explicit time stepping scheme results from an evaluation of the forms in equation (5.3) at time level n. Hence, the formulation of inviscid flux using Roe's scheme and viscous flux using the central-difference scheme will take the form

$$\Delta U_e = -\frac{\Delta t_e}{2\Omega_e}\sum_{s_e}[F_e^n + F_r^n - \mid A_{roe}^n \mid (U_r^n - U_e^n) + G_s^n]\,\delta s_e \tag{5.26}$$

Here G_s^n is calculated at each side "s" of the element. Local time stepping is used to update the calculation.

The allowable time step for the numerical solution of the Navier-Stokes equations can be obtained by analogy with the advection-diffusion equation as

$$(\delta t)_s \leq \frac{h_s}{(\lambda_{max})_s + \frac{4\mu_s}{Re_\infty \rho_s h_s}} \tag{5.27}$$

where the definition of h_s is the same as in section 3.4 and μ_s and ρ_s are determined as the average of the values at the interface "s" with two elements "e" and "r" at both sides.

As in the Euler equations, the local time step for an element is determined from the minimum of the time steps calculated on all its sides.

5.5.2. Point implicit scheme

If the flux contributions are evaluated at time t^{n+1}, equation (5.3) will lead to the fully implicit scheme as

$$\frac{\Delta U_e}{\Delta t_e}\Omega_e = -R(U_e^{n+1}) \tag{5.28}$$

By using Roe's scheme for the inviscid flux and central scheme for viscous flux, the right hand side term in equation (5.28) can be expressed as

$$R(U_e^{n+1}) = \sum_{s_e}\left\{\frac{1}{2}[F_e^{n+1} + F_r^{n+1} - \mid A_{roe}^{n+1} \mid (U_r^{n+1} - U_e^{n+1})] + G_s^{n+1}\right\}\delta s_e \tag{5.29}$$

Linearization of the values of unknowns and the inviscid fluxes using the same procedure as in Chapter 3 and for the viscous terms we have

$$G_s^{n+1} = G_s^n + B_s^n \Delta U_e \tag{5.30}$$

where B_s^n is the Jacobian matrix of the transformation.

Replacing the above linearization into the general finite volume formulation and re-arranging the terms results in a point implicit time integration scheme as

$$\left[LHS_{inv}^n - \frac{\Delta t_e}{\Omega_e}\sum_{s_e} B_{s_e}^n \delta s_e\right]\Delta U_e = RHS_{inv}^n + \frac{\Delta t_e}{\Omega_e}\sum_{s_e} G_{s_e}^n \delta s_e \tag{5.31}$$

where LHS_{inv} and RHS_{inv} denote the inviscid contributions to the left hand side and right hand side respectively.

The items in matrix B_s^n can be derived by the method called the variational recovery process [42] [91]. They are

$$B_{11} = 0 \qquad B_{12} = 0 \qquad B_{13} = 0 \qquad B_{14} = 0 \tag{5.32}$$

$$B_{21} = \frac{\phi_{1s}u_{1e}^n + \phi_{2s}u_{2e}^n}{\rho_e^n} \tag{5.33}$$

$$B_{22} = -\frac{\phi_{1s}}{\rho_e^n} \tag{5.34}$$

$$B_{23} = -\frac{\phi_{2s}}{\rho_e^n} \tag{5.35}$$

$$B_{24} = 0 \tag{5.36}$$

$$B_{31} = \frac{\phi_{3s}u_{1e}^n + \phi_{4s}u_{2e}^n}{\rho_e^n} \tag{5.37}$$

$$B_{32} = -\frac{\phi_{3s}}{\rho_e^n} \tag{5.38}$$

$$B_{33} = -\frac{\phi_{4s}}{\rho_e^n} \tag{5.39}$$

$$B_{34} = 0 \tag{5.40}$$

$$B_{41} = -\gamma\phi_{7s}\frac{(u_{1e}^2 + u_{2e}^2)^n - \rho_e^n}{(\rho_e^n)^2} + \frac{\phi_{5s}u_{1e}^n}{\rho_e^n} + \frac{\phi_{6s}u_{2e}^n}{\rho_e^n} \tag{5.41}$$

$$B_{42} = -\frac{\phi_{5s}}{\rho_e^n} + \frac{\gamma\phi_{7s}u_{1e}^n}{\rho_e^n} \tag{5.42}$$

$$B_{43} = -\frac{\phi_{6s}}{\rho_e^n} + \frac{\gamma\phi_{7s}u_{2e}^n}{\rho_e^n} \tag{5.43}$$

$$B_{44} = -\frac{\gamma\phi_{7s}}{\rho_e^n} \tag{5.44}$$

where

$$\phi_{1s} = \alpha(4n_1b_{se}/3 + n_2c_{se}) \tag{5.45}$$

$$\phi_{2s} = \alpha(-2n_1c_{se}/3 + n_2b_{se}) \tag{5.46}$$

$$\phi_{3s} = \alpha(n_1c_{se}/3 - 2n_2b_{se}/3) \tag{5.47}$$

$$\phi_{4s} = \alpha(n_1b_{se}/3 + 4n_2c_{se}/3) \tag{5.48}$$

$$\phi_{5s} = u_{1s}\phi_{1s} + u_{2s}\phi_{3s} \tag{5.49}$$

$$\phi_{6s} = u_{1s}\phi_{2s} + u_{2s}\phi_{4s} \tag{5.50}$$

$$\phi_{7s} = \alpha(n_1 b_{se} + n_2 c_{se})/Pr \tag{5.51}$$

$$\alpha = \mu_s/Re \tag{5.52}$$

$$b_{se} = \frac{(y_{N_2} - y_{N_1}) + (y_{N_3} - y_{N_4})(\frac{1}{idim(N_1)} - \frac{1}{idim(N_2)})}{2\Omega} \tag{5.53}$$

$$c_{se} = \frac{(x_{N_1} - x_{N_2}) + (x_{N_4} - x_{N_3})(\frac{1}{idim(N_1)} - \frac{1}{idim(N_2)})}{2\Omega} \tag{5.54}$$

here $idim(N_1)$ and $idim(N_2)$ are the number of elements around the node N_1 and N_2.

Since the inviscid and viscous terms on the right hand side of equation (5.31) are evaluated at time level n, the procedure is equivalent to a point Gauss-Jacobi iteration. It also can use the most recent values to determine the viscous contributions as well as the inviscid terms to the right hand side and result in the point Gauss-Seidel scheme. For a PGS scheme, equation (5.30) is replaced by

$$G_s^{n+1} = G_s^* + B_s^* \Delta U_e \tag{5.55}$$

and equation (5.31) will become

$$\left[LHS^*_{inv} - \frac{\Delta t_e}{\Omega_e}\sum_{s_e} B^*_{s_e}\delta s_e\right]\Delta U_e = RHS^*_{inv} + \frac{\Delta t_e}{\Omega_e}\sum_{s_e} G^*_{s_e}\delta s_e \tag{5.56}$$

where, as before, an asterisk represents an evaluation using the latest available values. Apart from using the latest values of variables, details of treatment will be similar to that of the PGJ iteration, i.e. the resulting equations can be obtained by substituting the superscript "n" with "*" for the elements surrounding the current element in equation (5.31).

To reduce the computational work two simplifications are made. Firstly, high order accuracy is kept only on the right hand side of equation (5.56) while the first order scheme is used for the inviscid Jacobian computation. Secondly, instead of $B^*_{s_e}$, $B^n_{s_e}$ is used to avoid the complexity. Thus the iteration solution algorithm is actually point Gauss-Seidel for the inviscid terms and point Gauss-Jacobi for the viscous terms.

5.6. Boundary conditions for Navier-Stokes equations

The formulation of the exterior boundary is similar to that given for the Euler equations. For the inner boundary, i.e. the solid wall, the boundary condition specific

to the Navier-Stokes equations is the no-slip wall condition which means the relative velocity between the fluid and the solid wall is zero. Assuming a fixed wall, all the velocity components at the wall are taken to be zero. For an isothermal wall, the temperature is fixed at the wall temperature. For an adiabatic wall, the heat flux is zero. In this case the temperature at the boundary is taken to be the same as the temperature at the adjacent element inside the domain. For the pressure, the boundary layer assumption $\frac{\partial p}{\partial n}$, is employed. Other variables, in particular the density, can be determined from the equation of state.

5.7. Validation of NS flow solver for laminar cases

The approach described in the previous sections is applied to compute external transonic viscous laminar flows. A NACA 0012 airfoil in a flow of Mach number 0.85 at zero angle of attack and with a Reynolds number of 500 is investigated. This case(A5) is one of several typical test cases for validating laminar Navier-Stokes algorithms suggested in [92]. During the present study several grid topologies are used. The outer boundary is set to 10 chords away from the airfoil. The no-slip boundary condition is used on the airfoil surface and a one-dimensional characteristic analysis is applied to the far-field boundary. All these computations were performed on a SG Indy workstation using the PGS-Roe code with the high order scheme and the Venkatakrishnan limiter.

Table III gives the results of lift coefficient, drag coefficient, number of iterations and the maximum residuals reached on each grid topology. Considering the most interesting results, i.e. C_D values in laminar viscous flow computation, the values are all distributed between 0.2136 to 0.2401 in the present calculations. Conclusions can be made from the C_D results on same test case given in table IV reproduced from [92]. Apart from two author's C_D values of 0.0964 and 0.179, which seem considerally different from the results of others, the remainder are all located in a band from 0.2176 to 0.2420. Thus the present results appear to be in very good agreement with those given by most of the authors published in [92].

The reference numbers in table IV refer to the following author(s):

(2) Angrand (INRIA,France);

(3) Bristeau, Glowinski, Mantel, Periaux and Pouletty (AMD/BA,INRIA,France);

(4) Cambier(ONERA,France);

(5) Grasso, Jameson and Martinelli (Italy, Uni. of Princeton, USA);

(6) Haase (Dornier Gmbh,FRG);

(7) Kalfon, Volpert and Brocard (ONERA-CERT, France);

(8) Kordulla (DFVLR,FRG);

Grid Topology	C_L	C_D	Ite	Log(res)
Regular Unstructured Grid (161 × 33) NE=10240 NP=5313 Edge=15522	4.44e-5	0.2138	4510	-3.456
Structured Grid (161 × 33) NE=5120 NP=5313 Edge=10402	1.06e-5	0.2238	4510	-3.35
Mixed stru/untru Grid (161 × 33) NE=9120 NP=5313 Edge=14402	0.38e-5	0.2136	4510	-3.35
Regular Unstructured Grid (221 × 33) NE=14080 NP=7293 Edge=21342	6.0e-5	0.2383	4510	-4.01
Structured Grid (221 × 33) NE=7040 NP=7293 Edge=14302	4.33e-5	0.2401	4510	-4.057
Mixed stru/unstru Grid (221 × 33) NE=12540 NP=7293 Edge=19802	3.48e-5	0.2384	4510	-3.92
AFT Unstructured Grid (half space) NE=7200 NP=3821 Edge=11021	0.0	0.2285	3000	-3.58
Hybrid Grid NE=15652 NP=9481 Edge=24813	0.38e-5	0.2247	4510	-3.776

Table III. Results of validating NS flow solver on test case (A5)

Ref. No.	2	3	4	5	6	7	8	9	10	11
C_L	1.e-5	1.e-5	7.e-4	0.0	1.e-5	0.0	4.e-4	4.e-5	0.0	1.e-4
C_D	0.2184	0.179	0.2221	0.0964	0.2176	0.23	0.242	0.2199	0.2261	0.2181

Table IV. Other computational results on test case (A5)

(9) Muller, Berglind and Rizza (FFA,Sweden);
(10) Satofuka,Morinishi and Nishida (Kyoto Inst. of Tech. Japan);
(11) Secretan, Dhatt and Nguyen (Uni. Laval, Canada).

Figure 35 gives the near airfoil meshes of the regular unstructured, structured and mixed structured/unstructured grids. Figure 36 gives pictures of the unstructured grid generated by the AFT and the hybrid grid generated by the author's "skin" method [89].

Figure 37 illustrates the results of the flow vector field, pressure and Mach number contours on the mixed structured/unstructured grid of 221×33 with a spacing of 0.005 chord length at the airfoil surface and 161 points on the airfoil surface. From the map of the velocity vector field (upper), the velocity profiles and the development of the viscous wake flow can be seen clearly. The two lower pictures give the pressure and Mach number contours respectively.

After reviewing the validation on the mixed grids, results on the irregular unstructured grid generated by the AFT method are now considered. As the present test case is a symmetric airfoil with a symmetric flow condition, only the flow over half the airfoil need be considered. For the AFT generator a minimum spacing of 0.015 chord length is used which produces 7201 elements in the flow domain (half space). Figure 38 gives the results of the velocity vector field, pressure and Mach number contours. Because the mesh distribution is irregular thus no clear velocity profile can be seen at a first glance. But by examining the results more closely, it can be seen that the velocity is reduced to zero on approach to the surface of the airfoil, due to the applied no-slip condition.

Finally a hybrid grid around the NACA 0012 airfoil is generated using the method developed in [89]. Validation of the Navier-Stokes solver is performed on such a mesh with a structured grid in the near-wall region and unstructured grid in the far-wall region. Figure 39 shows the results. It can be seen clearly that velocity profiles are produced along the airfoil surface and the viscous flow develops well in the wake region.

Figure 40 gives the comparison of the distribution of the skin friction coefficient distributions on mixed and hybrid grids. Good agreements can be seen for both grid topologies.

5.8. Turbulence flow and Baldwin-Lomax algebraic model

Up to now the Navier-Stokes flow solver is tested in laminar flow case. For turbulent flow the turbulence model must be considered. The most widely used turbulence models currently are either the field-equation or algebraic types. The use of

field-equation turbulence models (e.g. one-equation model such as Baldwin-Barth model [63] [87], Spalart-Allmaras model [45] [64] and two-equation models such as the $k-\epsilon$ model [65] [93]) are in principle more general than their algebraic counterparts (such as Baldwin-Lomax model [94] [95]) and appear well suitable for use on unstructured grids. The additional field equations may be discretized and solved on the unstructured grid in the same manner as the governing equations. However the solution of additional field equation can be quite expensive, especially in the thin boundary-layer regions near the wall, where the equation can become very stiff. Algebraic turbulence models, on the other hand, are relatively inexpensive to compute and have demonstrated generally superior accuracy and reliability for limited classes of problems, such as high-Reynolds number attached flow over streamlined bodies. However such models typically require information concerning the distance from the wall. Turbulence length scales, which are related to the local boundary layer and wake thickness, are determined by scanning the appropriate flow values along specified streamwise stations. In the context of a structured grid, grid points occur at regular streamwise locations, hence the implementation of algebraic models on such grids is straightforward. On the unstructured grid with irregular distributions of grid, such implementation is not so direct. Reference [94] proposed a method of using two different grids, i.e. a global unstructured grid and a local background structured grid, to calculate the turbulent flows using the Baldwin-Lomax algebraic turbulence model [61]. Here the same idea is used for the computation of turbulent flows over 2D airfoils. Different from [94], a hybrid grid is considered in this work. The structured grid in the near wall region is selected as the local background grid during the implementation. Hence no interpolation of variable values between global grid and background grid are needed. Thus no interpolation error will be produced.

The turbulence model to be integrated into the present Navier-Stokes flow solver is a slightly modified version of the algebraic Bladwin-Lomax model [61]. The two-layer equilibrium model, patterned after that of Cebeci [96], defines the eddy viscosity μ_t as a function of the minimum layer eddy viscosity. However the introduction of a hyperbolic tangent function produces a smoother more desirable eddy viscosity distribution, namely

$$\mu_t = \mu_{t,outer} tanh\left(\frac{\mu_{t,inner}}{\mu_{t,outer}}\right) \tag{5.57}$$

In the inner region, the Prandtl-van Driest formulation is used by replacing, for Navier-Stokes applications, the normal derivative of the velocity profile by the

absolute values of the vorticity as

$$\mu_{t,inner} = \rho l^2 |\omega| \tag{5.58}$$

where

$$\begin{aligned} l &= \kappa y(1 - e^{-\frac{y^+}{A^+}}) \qquad (5.59)\\ \kappa &= 0.4 \\ y^+ &= \frac{(\rho_{wall}|\tau_{wall}|)^{0.5}}{\mu_{wall}} y \\ A^+ &= 26.0 \end{aligned}$$

The expression contained within the parentheses is the van Driest damping factor, whilst $|\omega|$ is the magnitude of the vorticity which for two dimensional flows reads

$$|\omega| = |\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}| \tag{5.60}$$

κ is the von Karman constant and y is the normal distance from the wall.

For the outer layer, alternate expressions for the eddy-viscosity are proposed as

$$\mu_{t,outer} = KC_{cp}F_{wake}F_{Kleb} \tag{5.61}$$

$$F_{wake} = min(y_{max}F_{max}, \frac{C_{wk}y_{max}u_{diff}^2}{F_{max}}) \tag{5.62}$$

where $C_{cp} = 1.6$ and $C_{wk} = 0.25$.

It should be noted here that the original value for $C_{wk} = 0.25$, defined in the Baldwin-Lomax 1978 paper, has been changed for some computations to $C_{wk} = 1.0$. The higher values yield considerably stronger interactions between shock and boundary layer.

The Clauser parameter, K, is generally assigned to be a constant, i.e. $K = 0.0168$, although it varies slightly in the low-momentum Reynolds number range.

The smallest values for F_{wake} in equation (5.62) have to be taken. The quantities F_{max} and y_{max} are determined from the function $F = y|\omega|(1 - e^{-\frac{y^+}{A^+}})$.

In wakes the exponential term is set to zero. The quantity F_{max} is the maximum value of F that occurs in the velocity profile and consequently y_{max} defines the y-location where F equals F_{max}. However as y_{max} is the characteristic length in the outer boundary layer, peak values of F in the near wall region are ignored.

Also in the equation (5.62), u_{diff} is the difference between maximum and mini-

mum velocity in the profile

$$u_{diff} = u_{max} - u_{min} \tag{5.63}$$

where u_{min} is always taken to be zero except in wake regions.
Furthermore the Klebanoff intermittency factor F_{Kleb} is given by

$$F_{Kleb} = \frac{1}{1 + 5.5(\frac{yC_{Kleb}}{y_{max}})^6} \tag{5.64}$$

with $C_{Kleb} = 0.3$.

The algebraic turbulence models described here are two-layer eddy-viscosity models. This means that according to the eddy-viscosity concept, in terms of the Navier-Stokes equations for laminar flow, the molecular viscosity μ is replaced by $\mu = \mu_l + \mu_t$ while in heat-flux terms $\frac{\mu}{Pr}$ is replaced by $\frac{\mu}{Pr} = \frac{\mu_l}{Pr_l} + \frac{\mu_t}{Pr_t}$ with the Prandtl numbers chosen to be $Pr_l = 0.72$ for the laminar flow and $Pr_t = 0.9$ for the turbulent flow.

The second coefficient of viscosity λ is also replaced by $\lambda = -\frac{2}{3}(\mu_l + \mu_t)$. However some researchers still leave the second coefficient of viscosity unchanged.

5.9. Validation for turbulence cases

The implementation of the algebraic B-L turbulence model described in the previous sections is combined with the PGS-Roe code to simulate two-dimensional turbulent flows. The high order scheme with the Venkatakrishnan limiter is also considered. Three examples of turbulent airfoil flows are tested.

The first example is a subsonic flow over the NACA 0012 airfoil with the flow conditions of Mach number of 0.5, zero incidence and Reynolds number of 2.89×10^6. The calculation is carried out on the mixed structured/unstructured grid of 221×33 with 161 points on the airfoil surface. The mixed grid is constructed in this way. After the generation of a structured C-type grid over the airfoil using the EAGLE grid generator, the inner 20 layers of structured grids are fixed and the outer 13 layers of structured grids are converted to the regular unstructured grid by halving them. In the calculation, the turbulence model is only considered on the background reference grid, i.e. the inner layers with a structured grid. Figure 41 gives the computational result of the developed NS-turbulence solver which compared very well with the experimental data given in [97].

The second example considered is also a subsonic flow over a NACA 0012 airfoil with the flow conditions of Mach number of 0.502, incidence of 1.77^o and Reynolds number of 2.91×10^6. The mixed grid adopted is the same as that used in the first example. Figure 42 gives the comparison of present computational results with the

experiment data of reference [97]. Good agreement is shown.

The third example tested is a transonic flow over the RAE 2822 airfoil. The selected test example is case 9 which is widely used for the CFD code validation. The experimental condition is a Mach number of 0.730, incidence of 3.19° and Reynolds number of 6.5×10^6. A slightly different flow condition is used in the computation, i.e. Mach number of 0.729, incidence of 2.31° and Reynolds number of 6.5×10^6, as most researchers have used [65] [94]. Figure 43 illustrates the hybrid grid used in the calculation, which is constructured by the "skin" method proposed in [89]. The hybrid grid contains 6599 elements, 4937 nodes and 104 points on the airfoil surface. The local structured C-type background grid has 20 layers within which the turbulence model is considered. Beside the hybrid grid, other grid topologies such as structured and mixed structured/unstructured grids mentioned in laminar flow validation section have also been adopted in the present calculations. Figure 44 gives the pressure distributions obtained by our NS-turbulence code on structured, mixed and hybrid grids. Excellent agreement of the computational results with experiments have been achieved. Figure 45 shows the skin friction distributions along the chord of the airfoil. Comparisons are made of the computational results with the experiments.

5.10. Conclusions

From the study above the following conclusions can be made:

(1) The viscous laminar and turbulent flows have been successfully simulated by the Navier-Stokes equations on an unstructured grid;

(2) An efficient hybrid grid generation method named the "skin" method is proposed and successfully applied in the viscous flow simulation;

(3) Discretization of inviscid fluxes for the Navier-Stokes equation were performed by use of a high order upwind cell-centred finite volume method with Roe's approximate Riemann solver, while the viscous term was discretized through the central-difference scheme. At each time step the linear system of equations which arises through the linearization of the fluxes is approximately solved with a point implicit Gauss-Seidel iteration algorithm;

(4) The procedure of implementation of the algebraic Baldwin-Lomax turbulence model is discussed and successful computation is achieved by the application on turbulent flows over NACA 0012 and RAE 2822 airfoils on the mixed and hybrid grids.

Up to now a complete framework has been built up within which the method of the high order upwind cell-centred finite volume scheme of Navier-Stokes equations with the implementation of algebraic turbulence model on unstructured grid has been developed. However compared to those on structured grid, the CFD code based on

unstructured grid still requires the challenge of more memory requirements and large CPU time consumption. With the appearance of parallel computers and the parallel computing technique these two problems can now be overcome. In the next two chapters some parallel issues connected with the flow simulation on unstructured grids will be discussed. Some initial test results will be given. The prospects in this field will be shown.

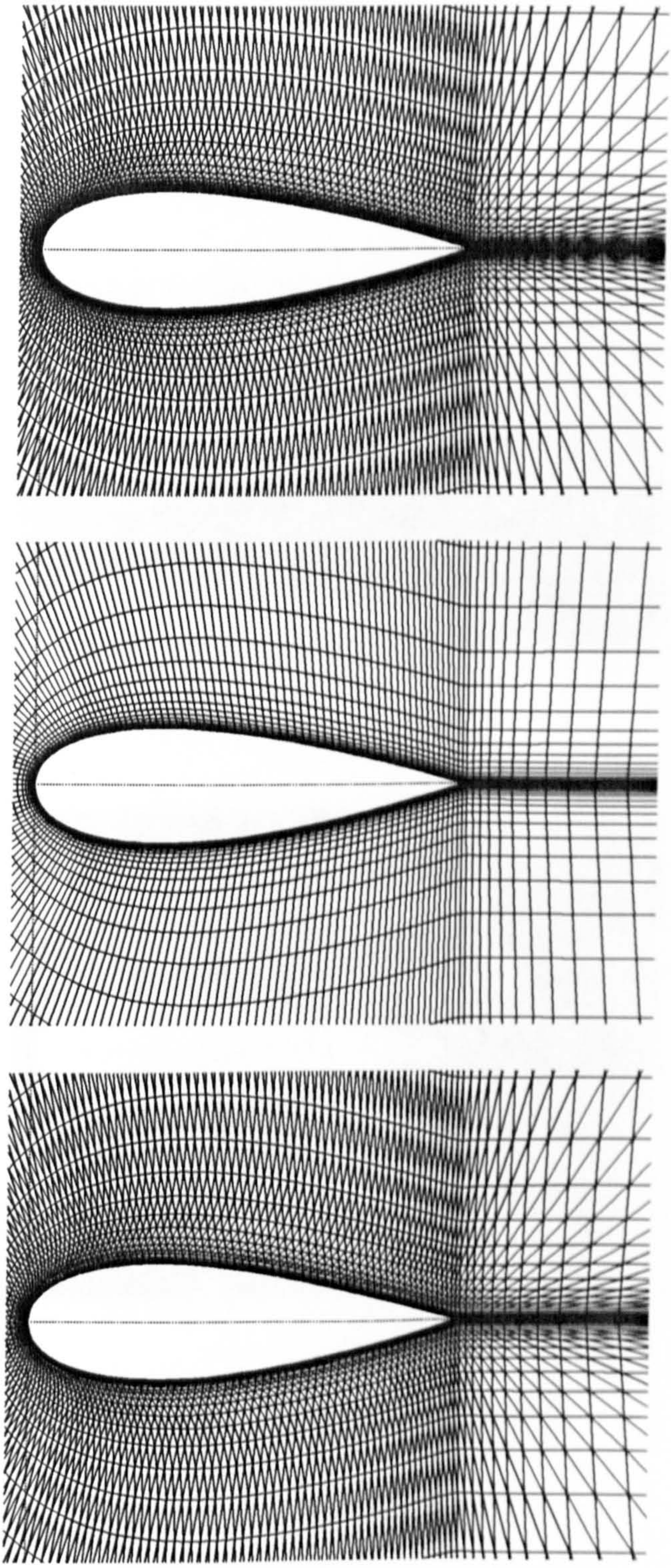


Fig. 35. Regular unstructured, structured and mixed structured/unstructured grids

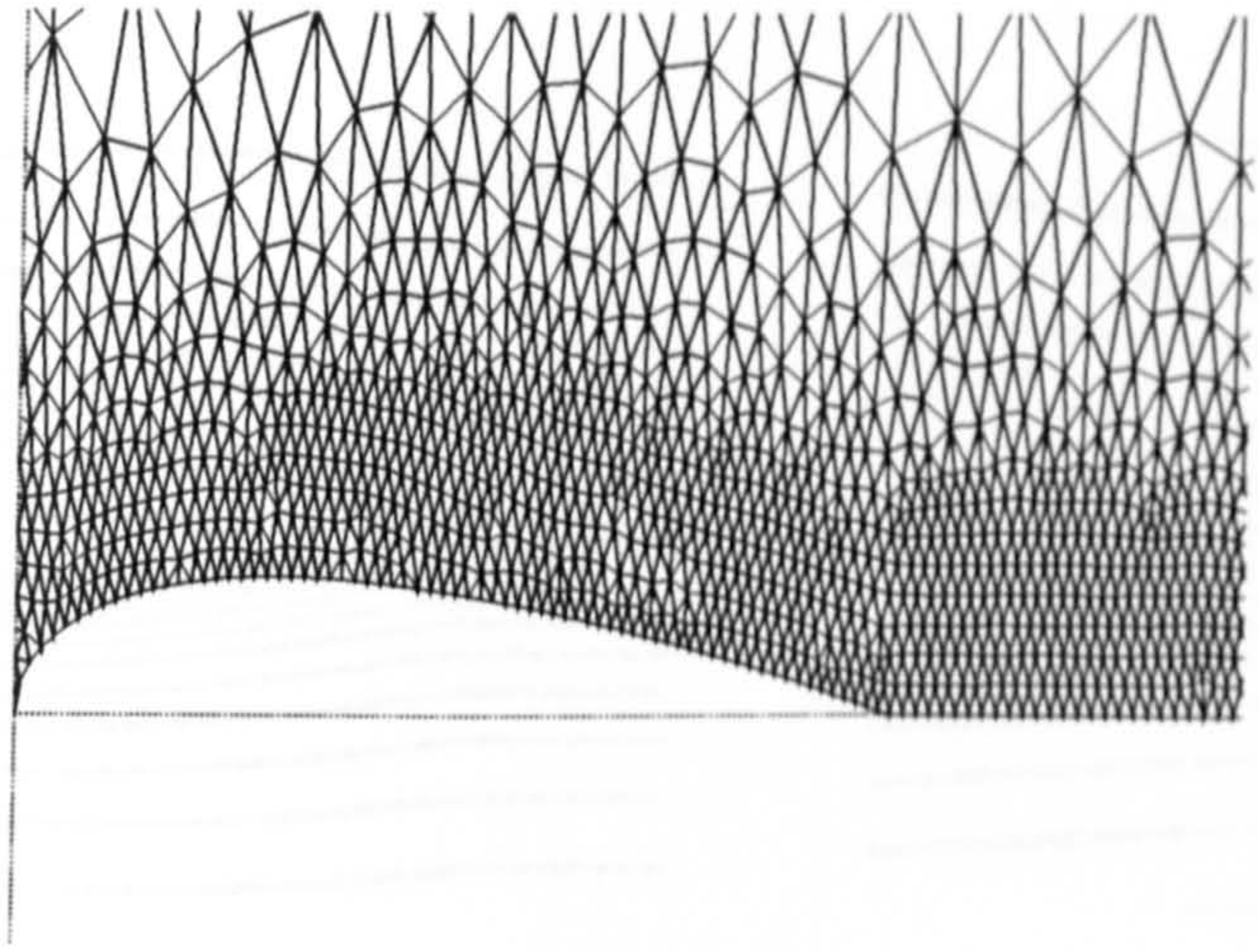


Fig. 36. Unstructured and hybrid grids

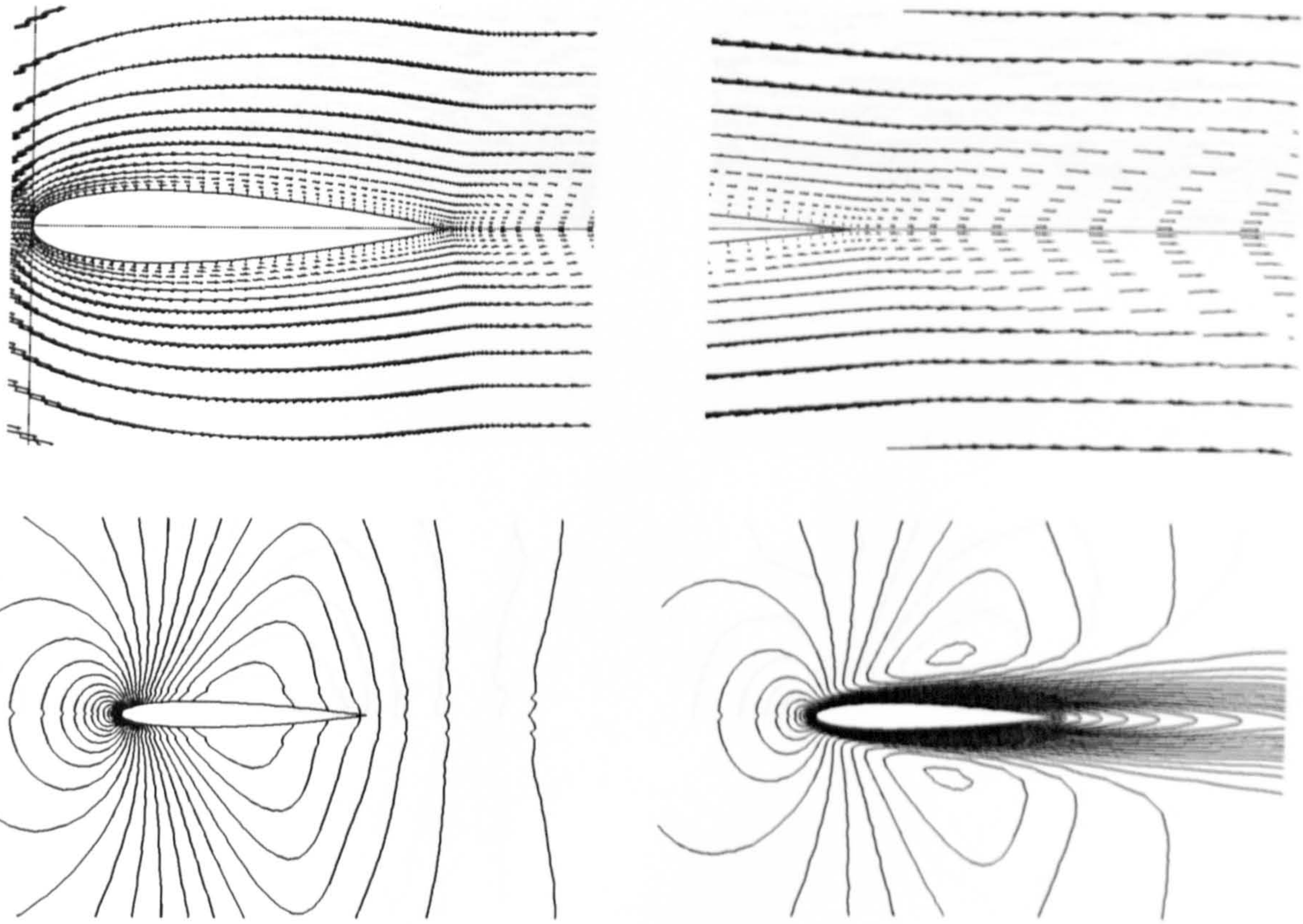


Fig. 37. Flow vector fields, pressure and Mach number contours of test case (A5) on mixed grids

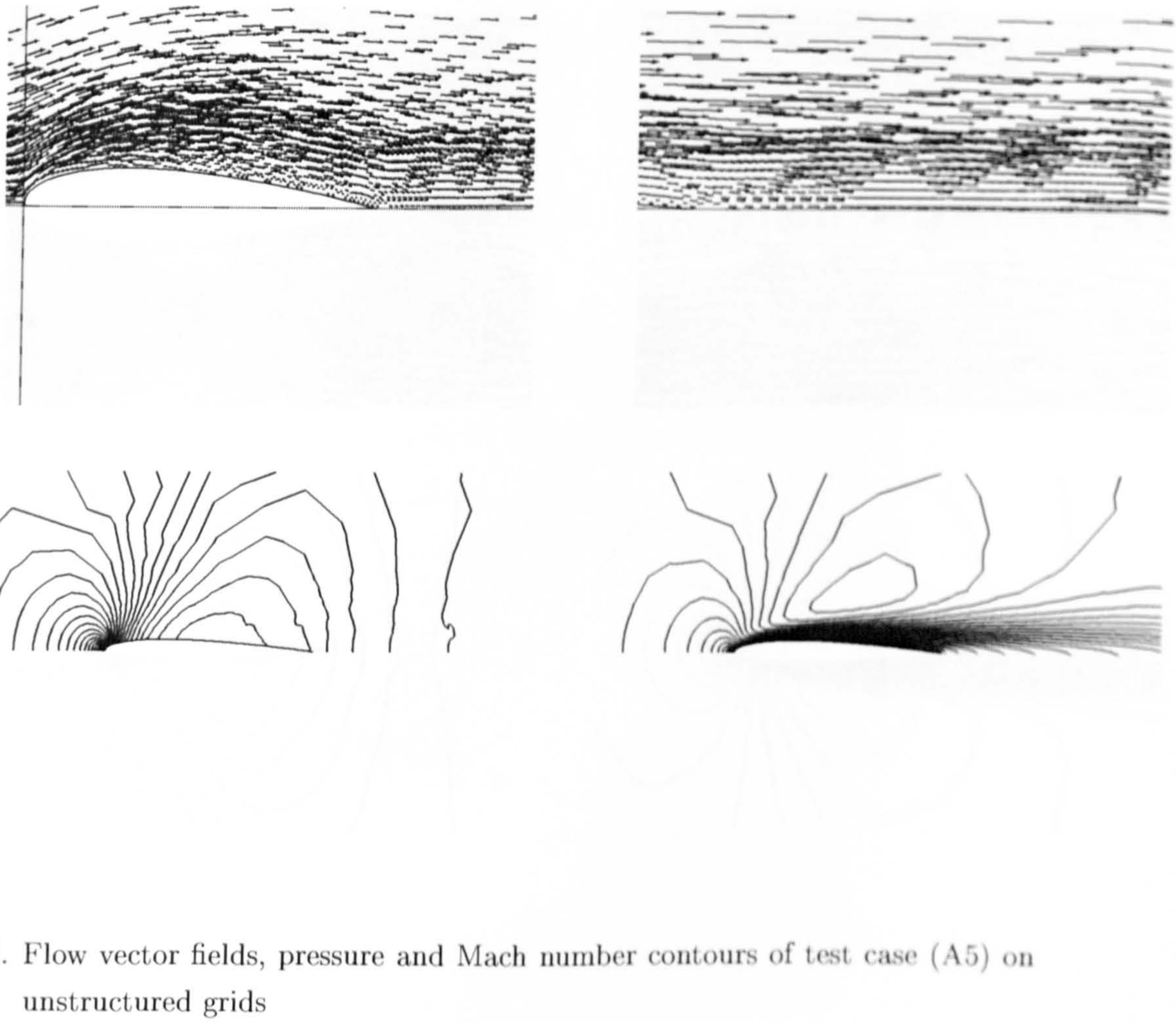


Fig. 38. Flow vector fields, pressure and Mach number contours of test case (A5) on unstructured grids

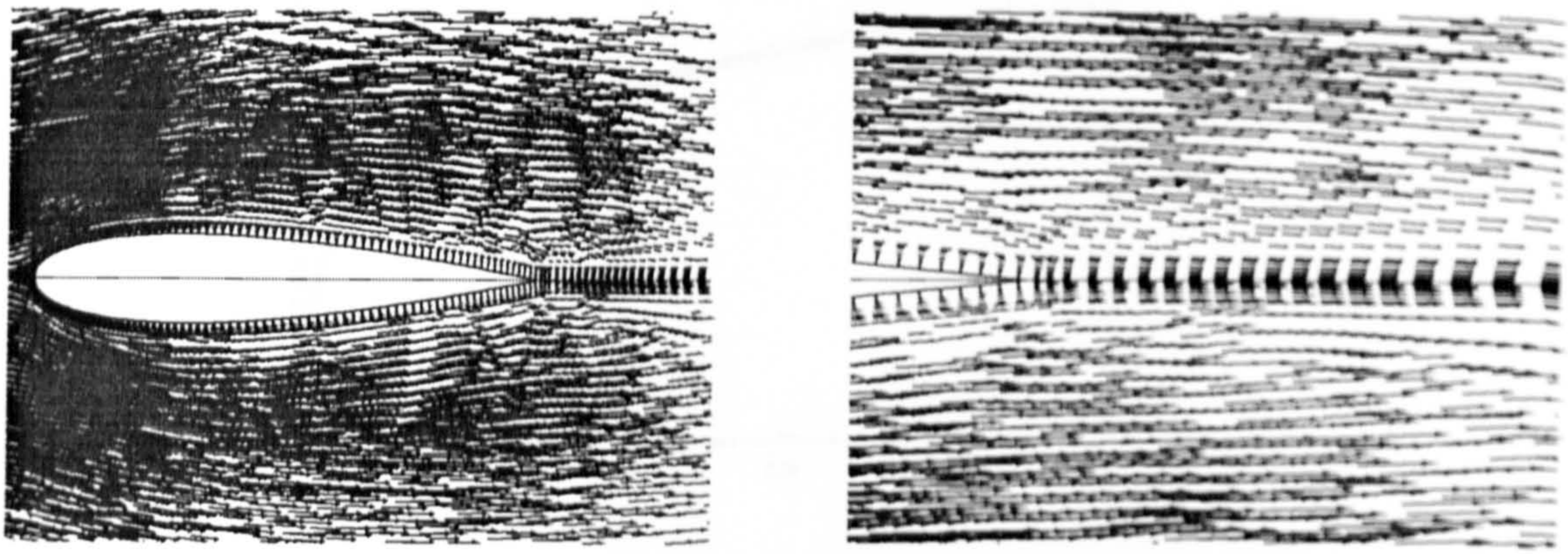


Fig. 39. Flow vector fields, pressure and Mach number contours of test case (A5) on hybrid grids

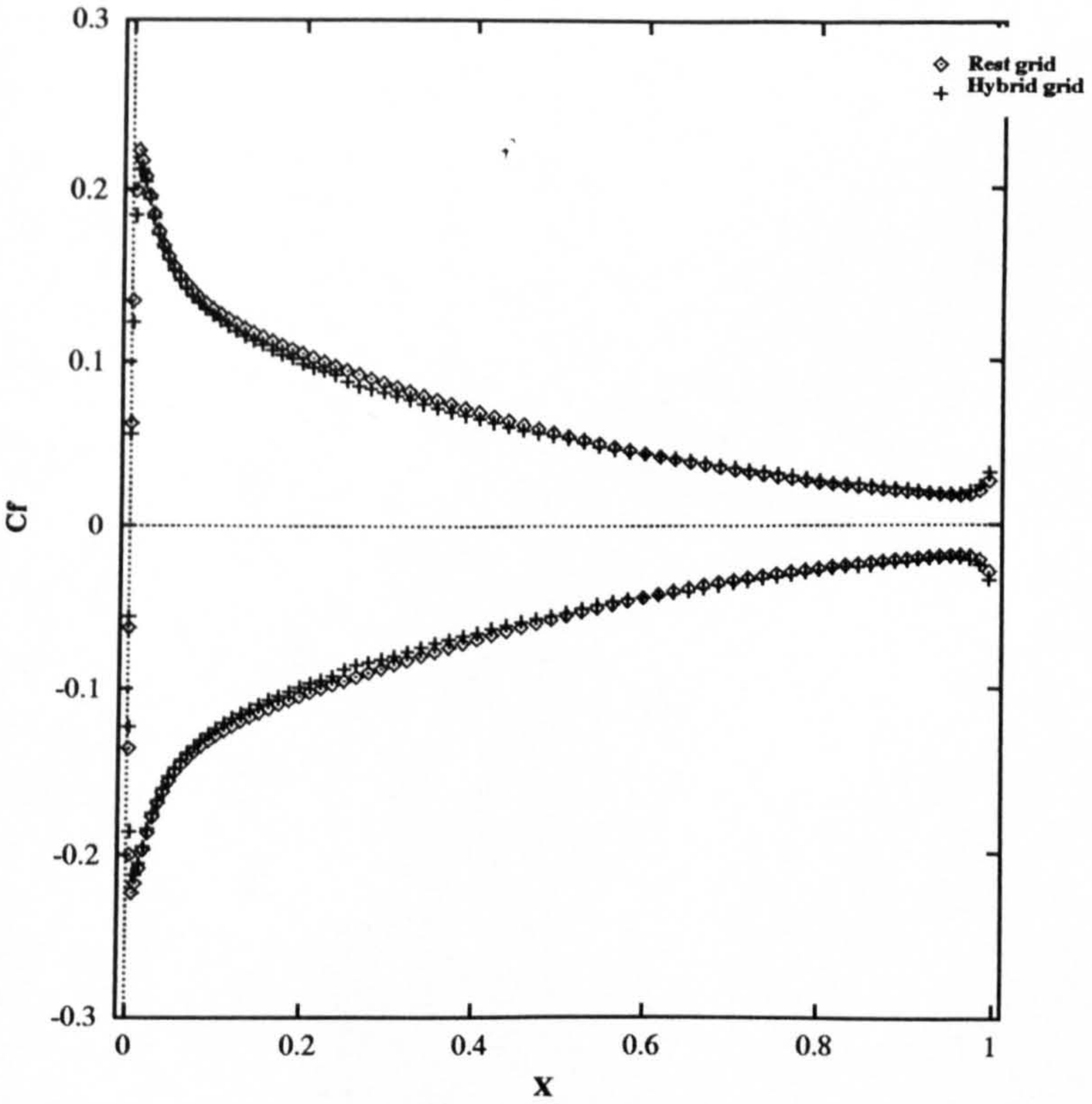


Fig. 40. Comparison of skin friction coefficients

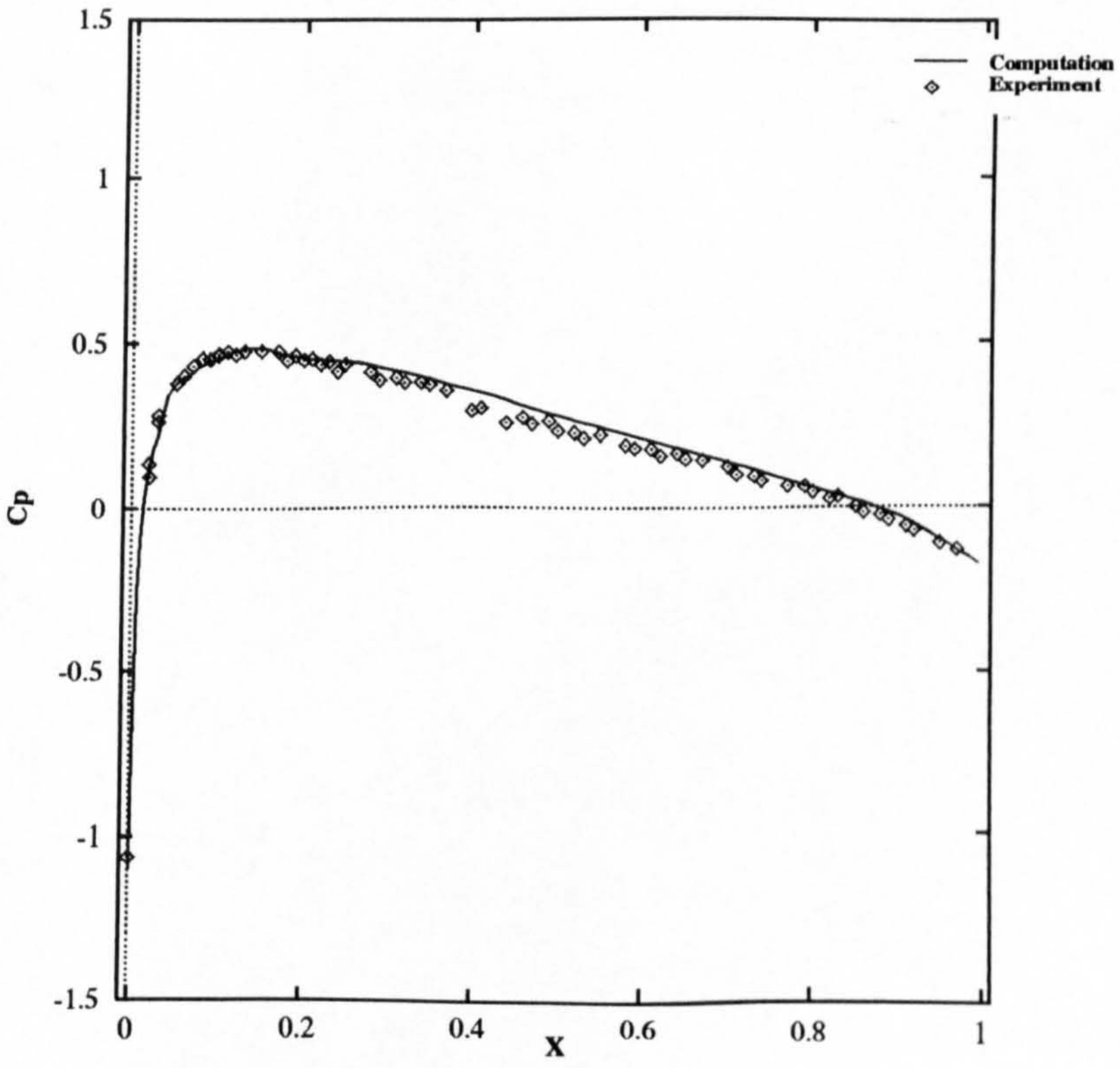


Fig. 41. Pressure distributions for turbulent flow over NACA 0012 airfoil at $M_\infty = 0.5$, $\alpha = 0$ and $Re = 2.89 \times 10^6$

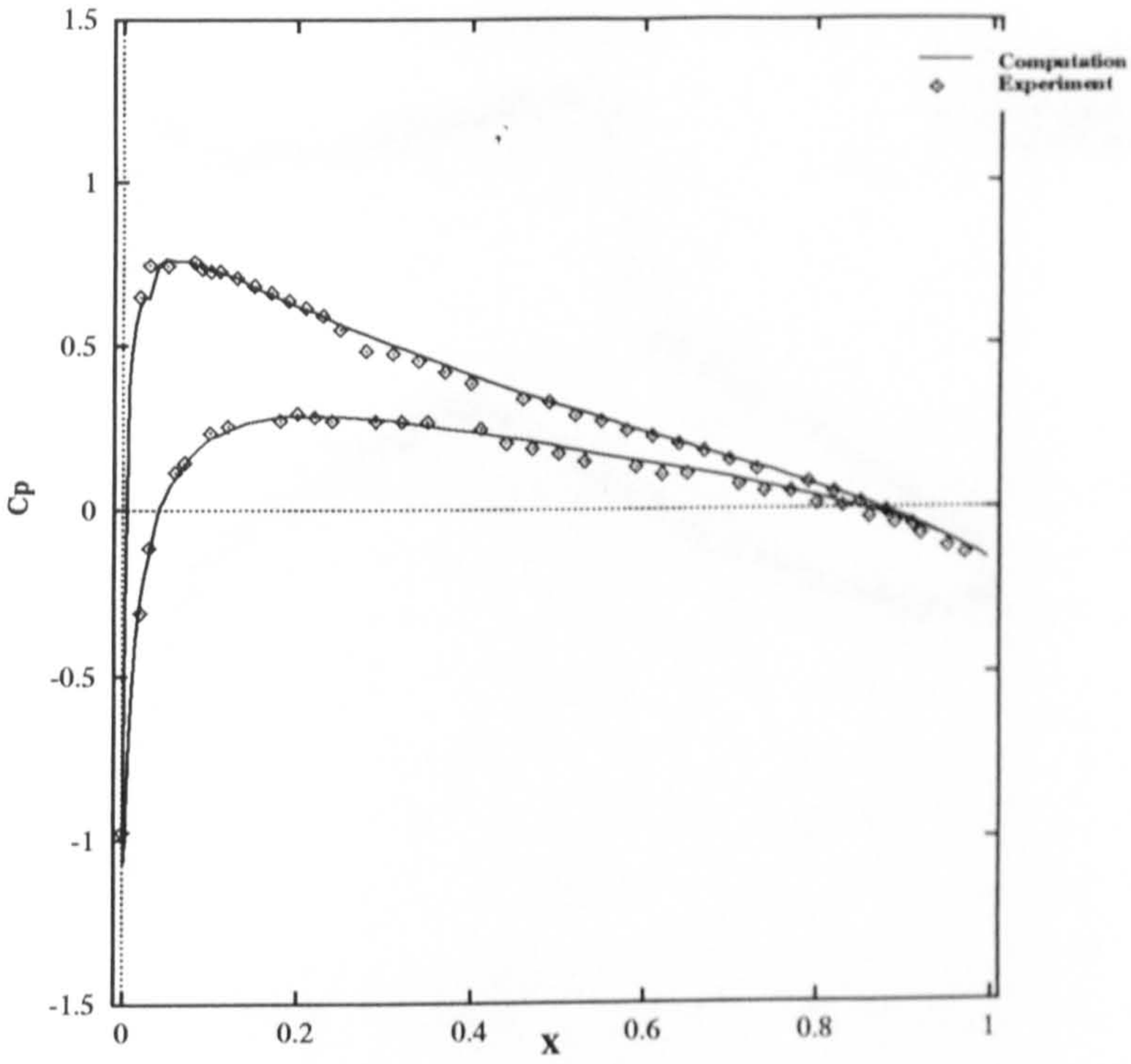


Fig. 42. Pressure distributions for turbulent flow over NACA 0012 airfoil at $M_\infty = 0.502$, $\alpha = 1.77^o$ and $Re = 2.91 \times 10^6$

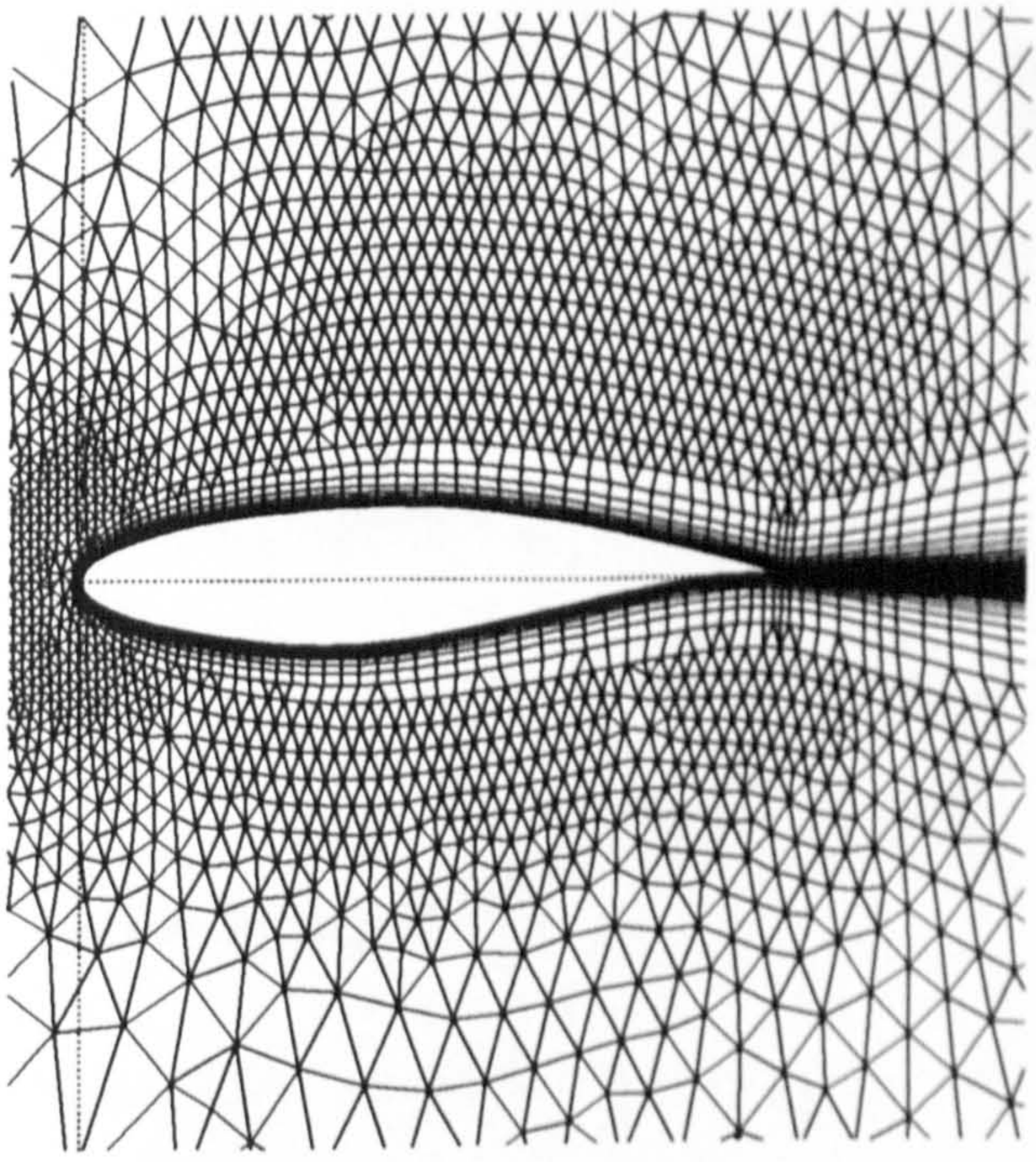


Fig. 43. Hybrid grids over RAE 2822 airfoil

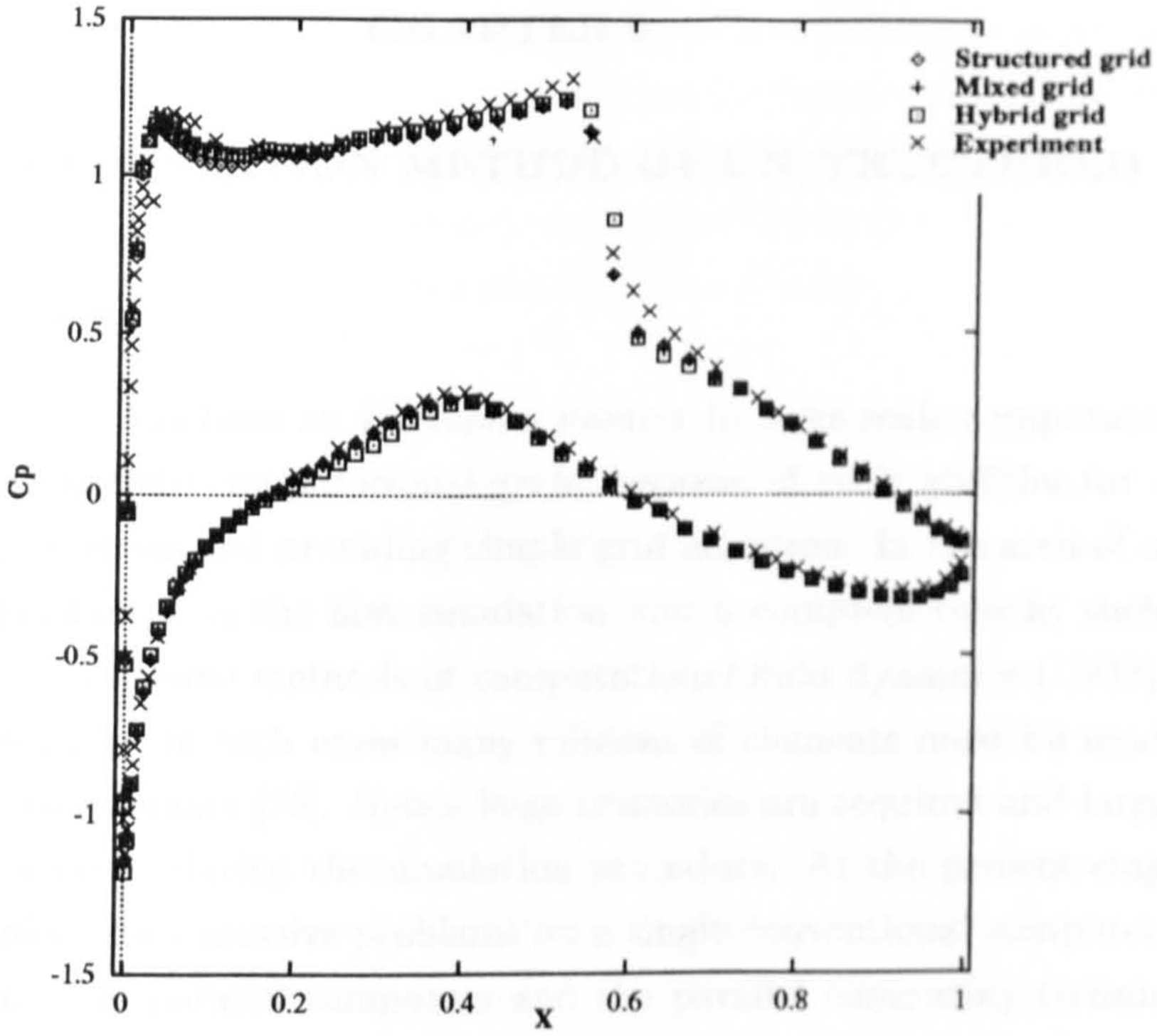


Fig. 44. Pressure distributions for turbulent flow over RAE 2822 airfoil (case 9) at $M_\infty = 0.729$, $\alpha = 2.31^o$ and $Re = 6.5 \times 10^6$

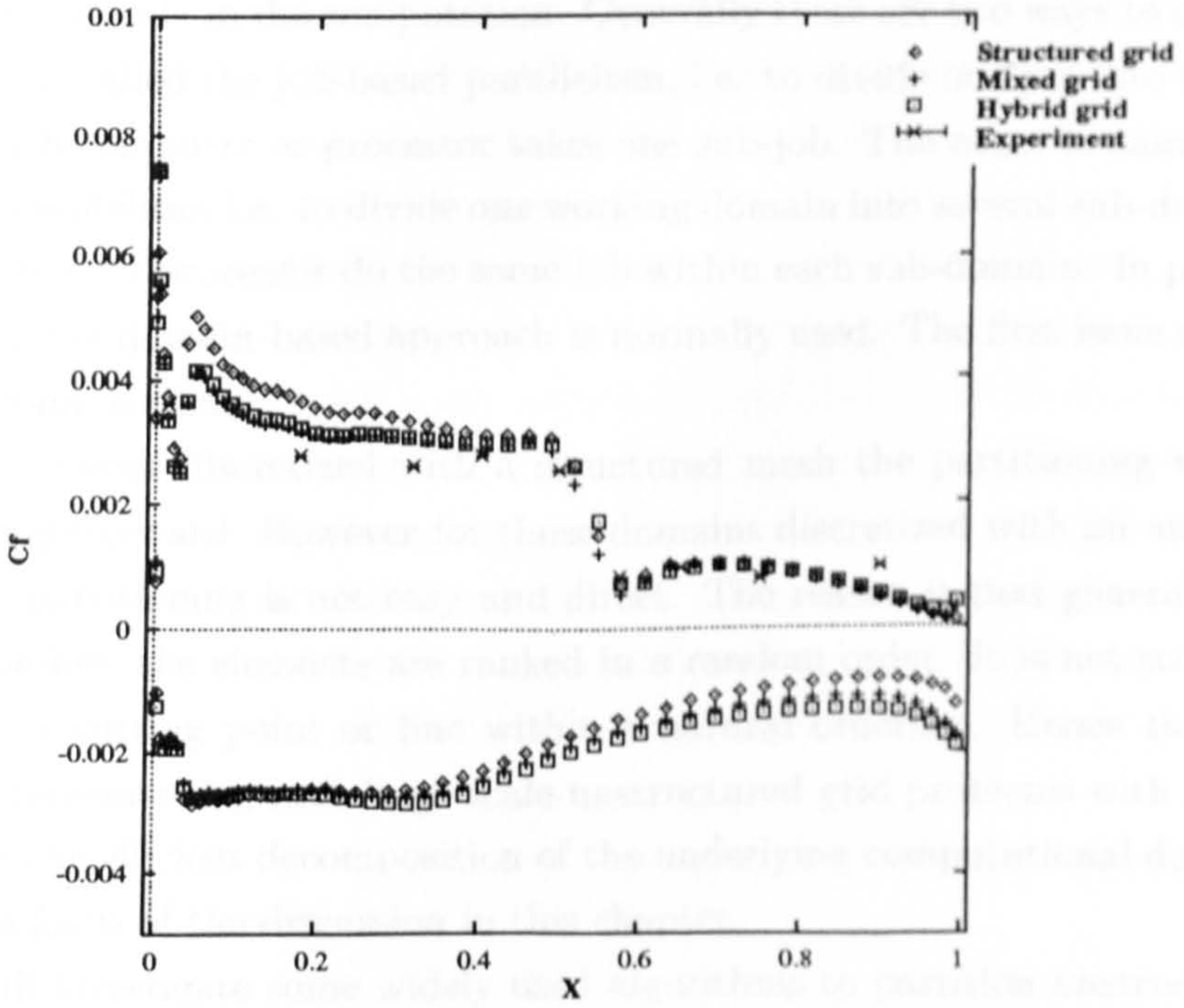


Fig. 45. Skin friction coefficients for turbulent flow over RAE 2822 airfoil (case 9) at $M_\infty = 0.729$, $\alpha = 2.31^o$ and $Re = 6.5 \times 10^6$

CHAPTER 6

DOMAIN DECOMPOSITION METHOD OF UNSTRUCTURED GRID

6.1. Introduction

In recent years there has been an increasing interest in large scale computations using unstructured discrete computational grids, because of their abilities for dealing with complex geometries and providing simple grid adaption. In the area of aerodynamics a typical example is the flow simulation over a complete vehicle, such as an aircraft, using finite volume methods of computational fluid dynamics (CFD) based on unstructured grids. In such cases many millions of elements must be used when discretizing the flow domain [75]. Hence huge memories are required and large CPU times will be consumed during the simulation procedure. At the present stage it is difficult to complete such massive problems on a single conventional computer.

With advances in parallel computers and the parallel computing technique, it becomes realistic to calculate such large scale problems based on supercomputers or even clusters of workstations by implementing a parallel computation procedure of the CFD code. The idea behind parallelism is simple. Many computers or processors are employed simultaneously in the computation. Generally there are two ways to achieve parallelism, one is called the job-based parallelism, i.e. to divide one job into several sub-jobs and each computer or processor takes one sub-job. The other is named the domain-based parallelism, i.e. to divide one working domain into several sub-domains with each computer or processor do the same job within each sub-domain. In parallel CFD computing the domain-based approach is normally used. The first issue will be the domain decomposition.

For those domains discretized with a structured mesh the partitioning will be simple and straightforward. However for those domains discretized with an unstructured mesh the partitioning is not easy and direct. The reason is that generally for unstructured meshes, the elements are ranked in a random order. It is not straightforward to find a cutting point or line within a natural ordering. Hence the first problem when implementing such large scale unstructured grid problems with parallel computing is the efficient decomposition of the underlying computational domain. This will be the focus of the discussion in this chapter.

Here we will investigate some widely used algorithms to partition unstructured domains. The first class of algorithms considered are recursive [66], i.e. the computational domain is subdivided by some strategy into two subdomains and then the

same strategy is applied to the subdomains in a similar way. In this way a partition into $p = 2^k$ subdomains is obtained after carrying out k of these recursive partitioning steps. These algorithms thus only differ by the partition strategy of a single domain into two subdomains. The three algorithms considered here are:

1) recursive coordinate (or angular) bisection (RCB or RAB);

2) recursive graph bisection (RGB);

3) recursive spectral bisection (RSB).

The RCB and RGB algorithms have been used by a number of researchers. In particular RCB is a very direct approach, which comes immediately into mind, when considering the partitioning problem. A similar method, named the RAB method, is also discussed when considering the ordering of angles instead of the ordering of coordinates. The third method named recursive spectral bisection (RSB), developed by Pothen, Simon and Liou [66], will also be discussed. The RSB method is based on the computation of an eigenvector of the Laplacian matrix associated with the graph. Compared to the RCB(RAB) and RGB algorithms, significant improvement is achieved by the RSB algorithm. However the cost of the spectral partitioning is still high (even using the Lanczos algorithm to compute the eigenvalue problem). Contributions to improve the efficiency of the RSB partitioning has been made by the proposed multilevel spectral bisection (MSB) [68]. Another class of partition algorithm is called multilevel graph partitioning (MGP). Karypis and Kumar [99] of Minnesota University have completed a series of researches on partitioning problems by mutilevel graph partitioning methods. The resulting package named Metis, version 2.0 is available in the public domain (http://www.cs.umn.edu/ karypis/metis/metis.html).

In section 6.2, we formulate a general framework for the partitioning problem based on some graph theory notation. In section 6.3, the three partitioning algorithms, RCB(RAB), RGB and RSB, are introduced and discussed firstly, and then the MGP method will be described later. Section 6.4 discusses in particular some preordering and smoothing techniques coupled with the procedure of the RCB(RAB) and RGB method to obtain a 'good' partitioning result. In section 6.5 we discuss another interesting method, i.e. the domain dividing technique (DDT). Section 6.6 gives some quantitative comparisons of algorithmic applications on 2D CFD problems. Section 6.7 offers conclusions.

6.2. The partitioning problem

So-called "efficient" partitionings are both dependent on the problem and the computer considered. Given N_P, the number of processors, one generally would like to partition the given problem into N_P sub-problems of about equal size (this process is

called load balancing) and at the same time minimize the amount of communication information needed between processors in a parallel computation. Minimizing the communication is a function of both the length of the boundary of the subdomains, as well as of the number of neighbouring subdomains. For an explicit algorithm, the achievement of good load balancing is probably more important than minimizing communication costs, whereas for an implicit algorithm with high communication requirements the situation may be the reverse.

In this work the main target is the implementation on a workstation cluster — a parallel computer system based on a distributed network in which each workstation represents one processor. The target application is to test the approach of explicit and point implicit two dimensional Euler/Navier-Stokes solvers on unstructured grids as described in the previous chapters. With this computer/application combination in mind the partitioning problem can be defined more precisely.

The partitioning problem can be considered as a generalization of the graph bisection problem, which is defined as follows: Given an undirected graph G, with the set of vertices V (either nodes or centre points of each element) and the set of edges E, $G = (V, E)$, partition $V = V_1 \cup V_2$,$V_1 \cap V_2 = 0$, such that

$$|E_e| = \{e|e \in E; e = (v_1, v_2); v_1 \in V_1; v_2 \in V_2\} \tag{6.1}$$

is minimized, subject to some constraint on the partition. Here we choose $|V_1| = |V_2|$, if $n = |V|$ is even and $|V_1| = |V_2| - 1$, if n is odd.

The assumption that the underlying problem can be expressed as an undirected graph is in no way restrictive. For example, for our target application, the upwind cell-centred finite-volume flow solver for the Euler/NS equations, the solution variables are associated with each element and the flux computation is performed at the edges of each non-overlapping control volume. Each edge connects a pair of control volumes. In the partitioning which we are planning to use, mesh triangles are assigned to a particular processor. Fluxes are computed by the individual processors associated with the triangles. Communication is required along the common edges, which are shared between the adjacent triangles residing in different processors. Hence for the purposes of establishing the partitioning of the problem, i.e., the assignment of triangles to different processors, we consider the *dual* graph. The triangles of the original mesh are vertices of the dual graph, and two triangles are considered to be adjacent vertices of the graph, if and only if they share an edge in the original mesh. A graph partitioning of this dual graph will thus yield an assignment of triangles to processors. In a similar way most general partitioning problems can be transformed to a graph partitioning problem. The approach used here is thus quite generally

1	Determine the longest expansion of the domain (x,y,or z direction)
2	Sort the vertices according to coordinates in the selected direction
3	Assign half of the vertices to each subdomain
4	Repeat recursively (divide and conquer)

Table V. Algorithm of recursive coordinate bisection (RCB)

applicable.

The relationship between the unstructured mesh and its dual graph is shown in Figure 46 and Figure 47.

6.3. Partitioning algorithms

The general idea behind the first three partitioning algorithms is to use a strategy to partition a domain into two subdomains, and then to apply the same algorithms recursively for k steps until $p = 2^k$ subdomains have been obtained. The algorithms thus only differ in the partitioning strategy for a single domain into two subdomains. The last algorithm described is named the multilevel graph partitioning (MGP) method.

6.3.1. Recursive coordinate bisection (RCB) or angular bisection (RAB)

This algorithm is very simple conceptually. It is based on the assumption that along with the set of vertices $V = (v_1, v_2, \ldots, v_n)$, there are also two or three dimensional coordinates available for the vertices. For each $v_i \in V$ we thus have an associated duple $v_i = (x_i, y_i)$ or triple $v_i = (x_i, y_i, z_i)$, depending on whether we have a two or three dimensional model. A simple bisection strategy for the domain is then to determine the coordinate direction of longest expansion of the domain. Without loss of generality, assuming that this is the x-direction, then all vertices are sorted according to their x-coordinate. Half of the vertices with small x-coordinates are assigned to one domain, the other half with the large x-coordinates are assigned to the second subdomain. The algorithm for RCB is summarized in table V.

Figure 50 gives an example of application of the RCB algorithm on an unstructured mesh used for the calculation of the flow over a NACA 0012 airfoil resulting in 8 partitioning.

Similar to RCB, an alternative method, named RAB, is also considered. The only difference is that it selects the angle ordering instead of the coordinate ordering.

The algorithm of RAB is the same as that of RCB except for the ordering procedure. Figure 52 gives the results of RAB applied to the unstructured mesh over an NACA 0012 airfoil shape resulting in 8 sub-domains.

6.3.2. Recursive graph bisection (RGB)

The weakness of both RCB and RAB is that the algorithm does not take full advantage of the connectivity information given by the graph. For efficiency, the main goal is to minimize the number of graph edges, which are connecting different subdomains. Thus instead of using the Euclidean distance between the vertex coordinates, a better way is to consider the graph distance between a vertex given by $d(v_i, v_j) = shortest\ path\ connecting\ v_i\ and\ v_j$. With this change one can define a new partitioning algorithm, which is called recursive graph bisection (RGB).

First two vertices of maximal or near maximal distance in the graph are determined. Then all other vertices are sorted in the order of increasing distance from one of the extremal vertices. Finally vertices are assigned to two sub-domains according to the graph distance. The only difficulty is the determination of the diameter (or at least of a pseudo-diameter) of the graph. However there exist some very good heuristic algorithms for that purpose. These algorithms are also quite well-known in the engineering structures community, since they can also be used for reducing the storage requirements of sparse matrices in envelope or skyline storage format. Here the reverse Cuthill-McKee (RCM) algorithm of SPARSPAK [100] is used.

The RCM algorithm first finds two pseudo-peripheral vertices in the graph (i.e. vertices which have a very large distance, but which are not necessarily the pair of vertices with maximum distance). Then starting from one of the vertices, the root vertex, a so-called level structure is constructed. The level structure is a convenient way of orginizing the vertices in the graph in sets of increasing distance from the root. Hence the level structure delivered by the RCM algorithm forms the basis for the recursive graph bisection algorithm. Half of the vertices, the ones which lie closer to the root are assigned to one subdomain, the remaining vertices to the other subdomain. If we start out with a connected graph then by construction it is guaranteed that at least one of the two subdomains (the one including the root) is connected. The algorithm of RCM is summarized in table VI.

Figures 48 and 49 illustrate non-zero entries of Laplacian matrix produced from natural ordering and RCM ordering. It shows that the band of the matrix is significantly reduced using RCM ordering.

By using RCM the algorithm of RGB can be summarized in table VII.

Figure 54 gives the results of the RGB algorithm for a NACA 0012 airfoil resulting

1	Find vertex with lowest degree. This is the ROOT vertex.
2	Find all neighbouring vertices connecting to the ROOT by incident edges. Order them by increasing vertex degree. This forms level 1.
3	Form level k by finding all neighbouring vertices of level k-1 which have not been previously ordered. Order these new vertices by increasing vertex degree.
4	If vertices remain, go to 3.

Table VI. Algorithm of reverse Cuthill-Mckee ordering

1	Use the RCM algorithm to compute a level structure
2	Sort vertices according to the RCM level structure
3	Assign half of the vertices to each sub-domain
4	Repeat recursively (divide and occupy)

Table VII. Algorithm of recursive graph bisection (RGB)

in 8 sub-domains.

6.3.3. Recursive spectial bisection (RSB)

The recursive spectral bisection algorithm (RSB) is derived from a graph bisection strategy developed by Pothen, Simon and Liou [66], which is based on the computation of a specific "second" eigenvalue of the Laplacian matrix associated with the graph G. The Laplacian matrix $L(G) = (l_{ij}), i, j = 1, \ldots, n$ is defined by

$$l_{ij} = \begin{cases} +1 & \text{if } (v_i, v_j) \in E \\ -deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

It is easily seen that

$$L(G) = -D + A \tag{6.3}$$

where A is the standard adjacency matrix of the graph

$$A_{ij} = \begin{cases} +1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

and D is the diagonal matrix with the entries equal to the degree of each vertex, $D_i = deg(v_i)$. From this definition, it is clear that the sum in each row of L is zero. Define an N-vector, $s = [1, 1, 1, ...]^T$. By construction we have that

$$Ls = 0 \tag{6.5}$$

This means that at least one eigenvalue is zero with s as an eigenvector. If G is connected then λ_2, the second largest eigenvalue, is negative. The magnitude of λ_2 is a measure of connectivity of the grpah. The eigenvector associated with λ_2 is called *Fielder* vector for short. The Lanczos algorithm [101] is used to calculate the *Fielder* vector. Detailed description of the RSB method can be found in [66].

The objective of the spectral partitioning is to divide any kind of mesh into two partitions with equal size such that the numbers of edges cut by the partition boundary is approximately minimized. In summary the recursive spectral bisection algorithm is given by table VIII.

The RSB has been applied to the NACA 0012 airfoil. The 8 partition is obtained in figure 57, which shows that the domains obtained from RSB are connected (even

1	Compute Fielder vector for graph using the Lanczos algorithm
2	Sort vertices according to the size of entries in the Fielder vector
3	Assign half of the vertices to each sub-domain
4	Repeat recursively (divide and occupy)

Table VIII. Algorithm of recursive spectral bisection (RSB)

though there is no theoretical proof for it), nicely rounded and compact compared to that given by the RCB and RGB method. The second example is a multi-element Subdoo (4 elements) airfoil. Figure 58 shows the unstructured grid over the airfoil, which includes 8018 elements and 4147 node points. The results of 8 partition is given in figure 59. Although the quality by the RSB method is good, it still suffers from the problem of more CPU time consumed because of the calculation of the *Fielder* vector. In [68] an improved algorithm named multilevel spectral bisection (MSB) is proposed. In the next sub-section a brief description will be given with another efficient method called multilevel graph partitioning, which runs even faster than the MSB code and the partitioning quality remains the same.

6.3.4. Multilevel graph partitioning (MGP)

Different from the recursive type schemes discussed above another class of graph partitioning algorithm is called the multilevel graph partitioning scheme, which reduces the size of the graph by collapsing vertices and edges, partitions the smaller graph and then refines it to construct a partition for the original graph. Surveys and details about the MGP algorithm can be found in [99]. Here only a brief description is given.

Compared to section 6.2 a more general definition of graph partitioning, i.e. the k-way graph partitioning problem is given as follows:

Given a graph $G = (V, E)$, with the set of vertices $|V| = n$ and the set of edges E, partition V into k subsets, $V_1, V_2, ..., V_k$ such that $V_i \cap V_j = 0$ for $i \neq j, |V_i| = n/k$, and $\cup_i V_i = V$, and the number of cut edges of E whose incident vertices belong to different subsets is minimized. By this definition those three schemes discussed above can be called 2-way partitioning or bisection.

The basic idea of the multilevel graph partitioning algorithm is simple. The graph G is first coarsened down to a few hundred vertices, a bisection of this much smaller graph is computed. and then this partition is projected back towards the original graph (fine graph), by periodically refining the partition. Since the finer graph has

1	read in the mesh file
2	list and number elements in 1-D natural ordering
3	select an ordering rule, i.e. the barycentre of element, the minimum coordinate value of its node, etc.
4	pre-order all elements according the rule selected.

Table IX. Algorithm of pre-ordering

more degrees of freedom, such refinements usually decrease the number of edge cuts. The process consists of three phases, i.e. the coarsening phase, the partitioning phase and the refining phase. Details can be found in [99]. The results of application of the MGP method can be found in figure 60, which shows 8 partitions on the Subdoo 4 elements airfoil.

6.4. Pre-ordering and smoothing technique

Although the RCB (RAB) and RGB schemes can guarantee a load balance, it is still difficult to obtain a 'good' partitioning result, i.e. connectivity of the sub-domain and smoothness of the cutting line. From the results obtained, some problems are revealed. One is that singular elements are sometimes found in the results of the RGB partitioning. This is obviously not a 'good' partitioning. The reason is that theoretically the RGB partitioning can only guarantee ONE sub-domain in which a root vertex is connected. Another problem is that the cutting line will normally take the saw-tooth shape (figures 50, 52 and 54) resulting in an increase in the cost of communications. These two phenomena are also reported in reference [87]. To date however the method of overcoming these shortcomings appears not to have been addressed. Here two algorithms are proposed to improve them.

One technique tested is pre-ordering. Generally the mesh generated by the advancing front technique or Delaunay triangulation is in a random order. Hence the graph of the mesh is also ordered randomly. This will result in the phenonemon of isolated element (figure 54). We found that such isolated element sometimes can be avoided by carrying out a pre-ordering step before implementing the partitioning. The algorithm of pre-ordering can be found in table IX.

The results of RGB are found to be improved by a pre-ordering technique. This can be seen in Figure 55 with pre-ordering when compared to figure 54.

1	do the partitioning using either RCB, RAB or RGB giving the initial sub-domain (without smoothing)
2	Flag each vertex with the number the same as that of the sub-domain it belongs to
3	do loop over each sub-domain (for each vertex counting the identity of its neighbouring vertex. If the identity of all its neighbours did not belong to this sub-domain then it means this vertex is separated from its sub-domain. Find out where it belongs and change its identity. If more neighbouring vertices belong to same neighbouring sub-domain found then this vertex is better belonging to that sub-domain. Change their identity.) end the do loop
4	re-counting the number of vertices in each sub-domain (this may result in a small lack of balance.)
5	output the sub-domain depending on its flag number

Table X. Algorithm of smoothing

Another technique is called smoothing. After partitioning the domain under the rule of load balancing using a particular strategy, there will normally be produced a saw-tooth shape boundary between sub-domains. The smoothing algorithm is used to adjust some vertices and change their identity, which will result in a cutting line with relative smoothness. The algorithm of smoothing is described in table X.

Figures 50, 52 and 54 give the result with smoothing and pre-ordering. Compared with their counterparts without using smoothing and pre-ordering techniques it can be seen that the partitioning results are improved.

6.5. Domain dividing technique

All the above discussions are based on certain pre-conditions, i.e. given a domain of interest, firstly construct the unstructured mesh, then partition the mesh using a particular strategy. The advantage of using this approach is that the load balance can be retained, although pre-ordering and smoothing need be considered as discussed

1	Define the interested domain
2	Divide the domain into several sub-domain
3	for each sub-domain use Delaunay triangulation (DT) to construct the unstructured mesh. As the DT method always takes the given boundary points as its triangular node, hence there will be no over-lapping points occuring in the common line between neighbouring sub-domains
4	Construct the relationship between the sub-domains and the data structure of the communication information

Table XI. Algorithm of domain dividing technique(DDT)

above. The shortcoming is a lack of knowledge of the quality of the partitioning results, i.e. the relationship between each sub-domain and the shape of cutting lines. Hence another approach is to consider the reverse way. For the given domain, divide it first into several sub-domains. In this way we can organise the relationships of the sub-domains and make the connections as well as the shape of cutting lines as simple as possible. Secondly we can construct the unstructured mesh in each sub-domain. Unfortunately one cannot guarantee strictly the load balancing in this way. This strategy is called the domain dividing technique (DDT). The algorithm of the DDT method is shown in table XI.

Figure 56 gives an example for the NACA 0012 airfoil shape with 4 sub-domains using the DDT method.

6.6. Comparisons

In this section firstly we will give some quantitative comparisons between three algorithms i.e. RCB, RAB and RGB. Before embarking on this we will make some general observations of the algorithms: both RCB and RAB produce long and narrow sub-domains; RGB creates slightly more compact sub-domains, but sometimes they will have disconnected sub-domains. Hence in order to get a "good" partitioning, a smoothing technique is needed for the RCB and RAB algorithms, while both pre-ordering and smoothing techniques are necessary for the RGB algorithm.

To obtain a more quantitative comparison , the number of cutting edges, named

Method	O	S	elements in number of sub-domain								E_c
RCB	No	No	606	607	607	607	606	607	607	607	232
RCB	No	Yes	605	606	606	608	607	608	607	607	227
RAB	No	No	606	607	607	607	606	607	607	607	262
RAB	No	Yes	596	611	604	605	601	619	599	619	206
RGB	No	No	606	607	607	607	606	607	607	607	330
RGB	Yes	No	606	607	607	607	606	607	607	607	366
RGB	No	Yes	591	603	597	626	586	612	616	623	255
RGB	Yes	Yes	591	603	607	616	593	605	613	626	253

Table XII. Partitioning results of application RCB, RAB and RGB on airfoil problems: nelem=4854 node=2504

E_c, in each sub-domains will be presented. Under the condition of balanced load the value of E_c represents the information exchange cost. The greater the value E_c, the more CPU time will be consumed during the communications.

Table XII gives the number of elements in each sub-domain and the number of cutting edges (E_c) for the three algorithms RCB, RAB and RGB with or without pre-ordering and smoothing. The unstructured mesh around the NACA 0012 airfoil shape considered here is generated by the AFT method. It results in 4854 elements and 2504 nodes.

Table XIII illustrates the results on a fine mesh which includes 21152 elements.

It should be noted in the above two tables that 'O' represents pre-ordering and 'S' represents smoothing.

Table XII shows that RGB will normally create more cutting edges than RCB and RAB. By using a smoothing technique E_c will reduce in each of the RCB, RAB and RGB methods. For the RGB algorithm in particular, by using both smoothing and pre-ordering, E_c reduces by 25% compared to that without its use. On the other hand the change in load balance is only 3%. Hence the benefit is clear. From Table XIII the same conclusions can be obtained.

Then the results of the DDT method will be discussed. The size and shape of sub-domains depend on the domain dividing strategy chosen. More compact, connected sub-domains can be achieved under careful investigation. The problem is how to

Method	O	S	elements in number of sub-domain								E_c
RCB	No	No	2644	2644	2644	2644	2644	2644	2644	2644	504
RCB	No	Yes	2643	2644	2646	2643	2643	2644	2643	2646	499
RAB	No	No	2644	2644	2644	2644	2644	2644	2644	2644	594
RAB	No	Yes	2617	2644	2644	2641	2645	2655	2629	2677	501
RGB(8)	No	No	2644	2644	2644	2644	2644	2644	2644	2644	805
RGB(8)	Yes	Yes	2598	2629	2620	2661	2603	2682	2657	2702	595
RGB(16)	No	No	1322	1322	1322	1322	1322	1322	1322	1322	1210
			1322	1322	1322	1322	1322	1322	1322	1322	
RGB(16)	Yes	Yes	1274	1324	1297	1333	1309	1311	1308	1352	928
			1296	1307	1331	1351	1311	1346	1336	1366	

Table XIII. Partitioning results of application RCB, RAB and RGB on airfoil problems: nelem = 21152 node = 10666

Method	elements in sub-domain				E_c
DDT	1040	1178	1184	1090	200

Table XIV. Partitioning results by DDT method

achieve load balance between sub-domains.

Table XIV illustrates the 4-subdomain partitioning results using the DDT method. Also we are interested in comparing the number of elements in each sub-domain and the number of cutting edges. Here we use Delaunay triangulation to generate the unstructured mesh in each sub-domain.

Table XIV shows that the number of elements in each sub-domain is different (min=1040 max=1184). By using the DDT method one can define the position of the boundary points (edges) resulting in the same number and position on the boundary. However it is not possible to know how many elements will be generated within the defined domain. Thus how to retain a reasonable load balance when using DDT needs still to be investigated.

Method	Partition						
	2	4	8	16	32	64	128
RSB	178	293	481	789	1269	1879	2744
MGP	139	259	428	800	1265	1827	2715

Table XV. The number of edge cuts E_c on Subdoo 4 elements airfoil

Method	Partition						
	2	4	8	16	32	64	128
RSB	2.421	5.763	8.48	10.981	13.815	16.468	20.362
MGP	1.55	1.57	1.61	1.8	1.9	2.74	2.8

Table XVI. The CPU time (seconds) for partitioning on Subdoo 4 elements airfoil

At last the comparison of the application of the RSB and MGP methods on large problems will be illustrated. It should mentioned here that the RSB code (version 2.2) has been kindly supplied by Dr. Simon of SG (he was formerly with NASA) and the MGP code (version 2.0) developed by Drs. Karypis and Kumar has been downloaded from the public domain source mentioned above. The example is the Subdoo 4-elements airfoil with an unstructured grid generated by the AFT method. The grid includes 29849 elements and 15183 nodes. The quality (the number of edge cuts) and the efficiency (CPU time) for partitioning this problem with RSB and MGP are given in the table XV and XVI.

The results in table XV demonstrate the same quality achieved by the RSB and MGP codes on large problems. From table XVI it can be seen that more efficiency is delivered by the MGP method. The rate of speed-up increases with the increase of the number of partitions. With 2 partitions the ratio of CPU time used is 1.56 and with 128 partition the ratio increases to 7.27.

6.7. Conclusions

From the above research the following conclusions are made:

1) RCB, RAB and RGB are three simple and easily used methods for partitioning

an unstructured mesh;

2) With a pre-ordering and smoothing technique the quality of partitioning by RCB, RAB and RGB can be considerably improved;

3) The DDT is a potential method of partitioning. But the achievement of good load balancing still needs further investigation;

4) The RSB and MGP are the two partitioning methods that produce the best quality. Compared to the other three methods they offer far better partitioning results, however the RSB consumes more CPU time because of the calculation of the *Fielder* vector. The MGP method offers a faster speed to achieve the results. Both methods supply nearly the same quality of partition.

Following partitioning, the next step is to implement the parallel procedure with the Euler/NS flow solver on an unstructured mesh. The major challenge remaining is how to construct an efficient message communication model between sub-domains. A data structure need be constructed to fulfill this requirement.

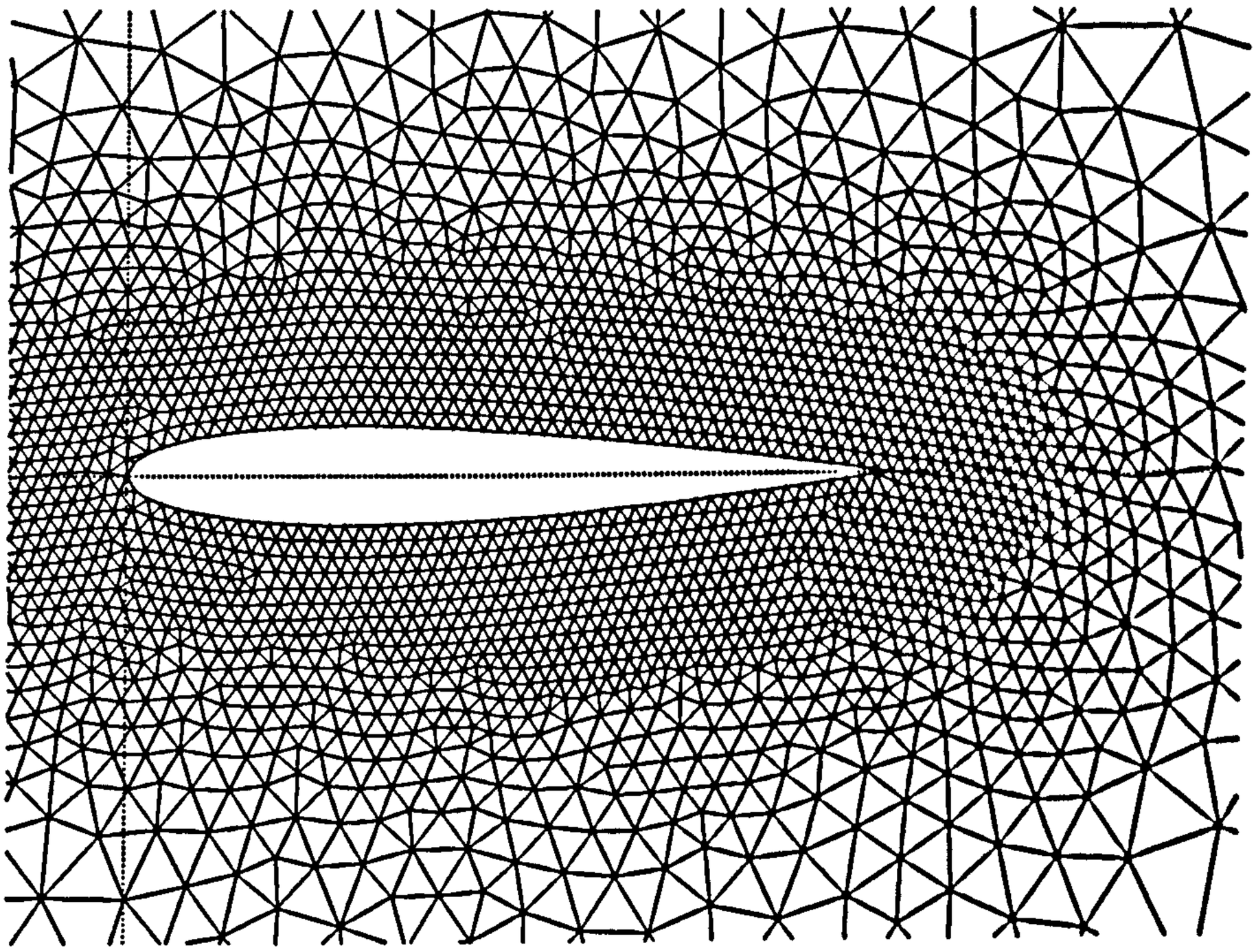


Fig. 46. Unstructured grids around NACA 0012 airfoil generated by AFT method

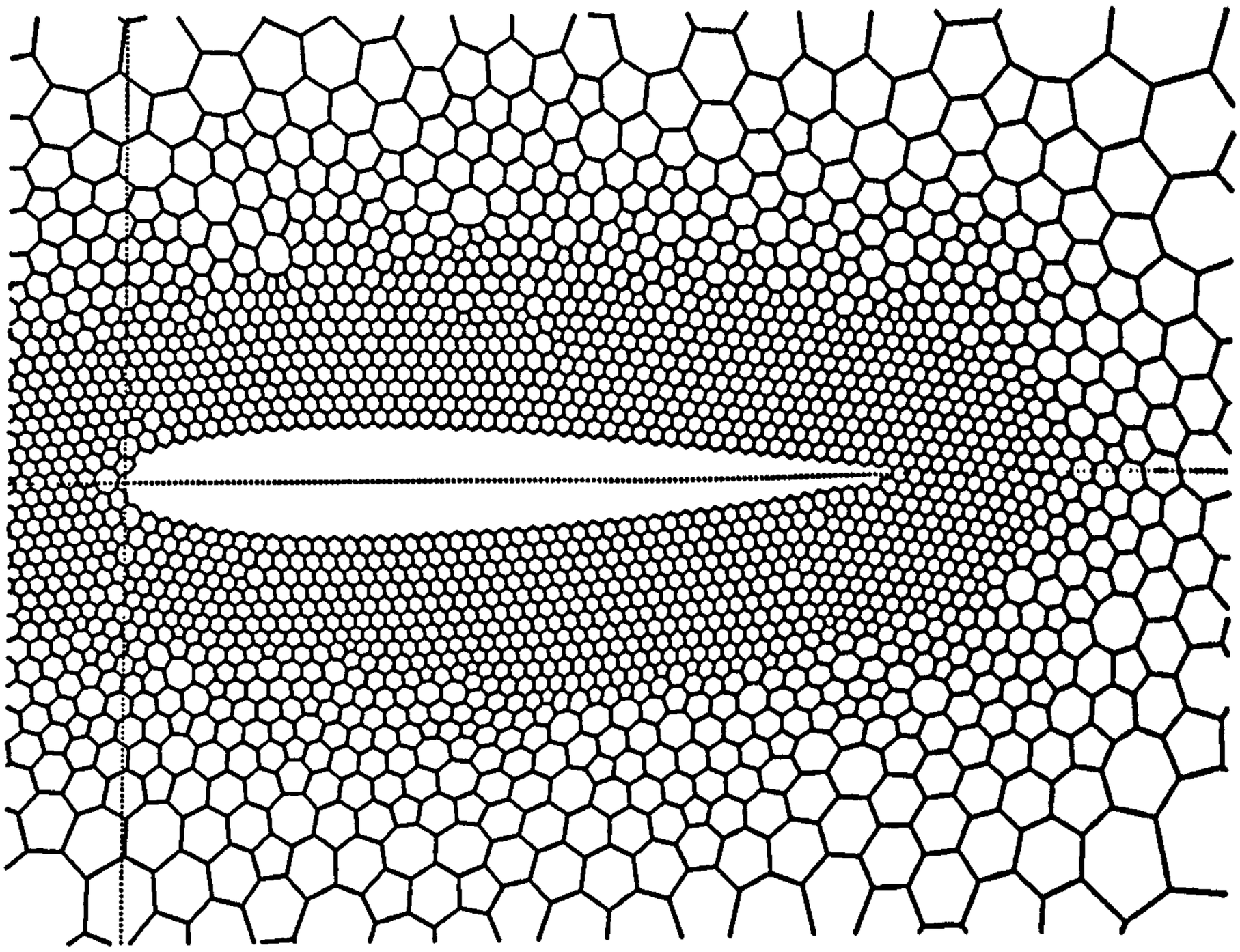


Fig. 47. Dual graph around NACA 0012 airfoil

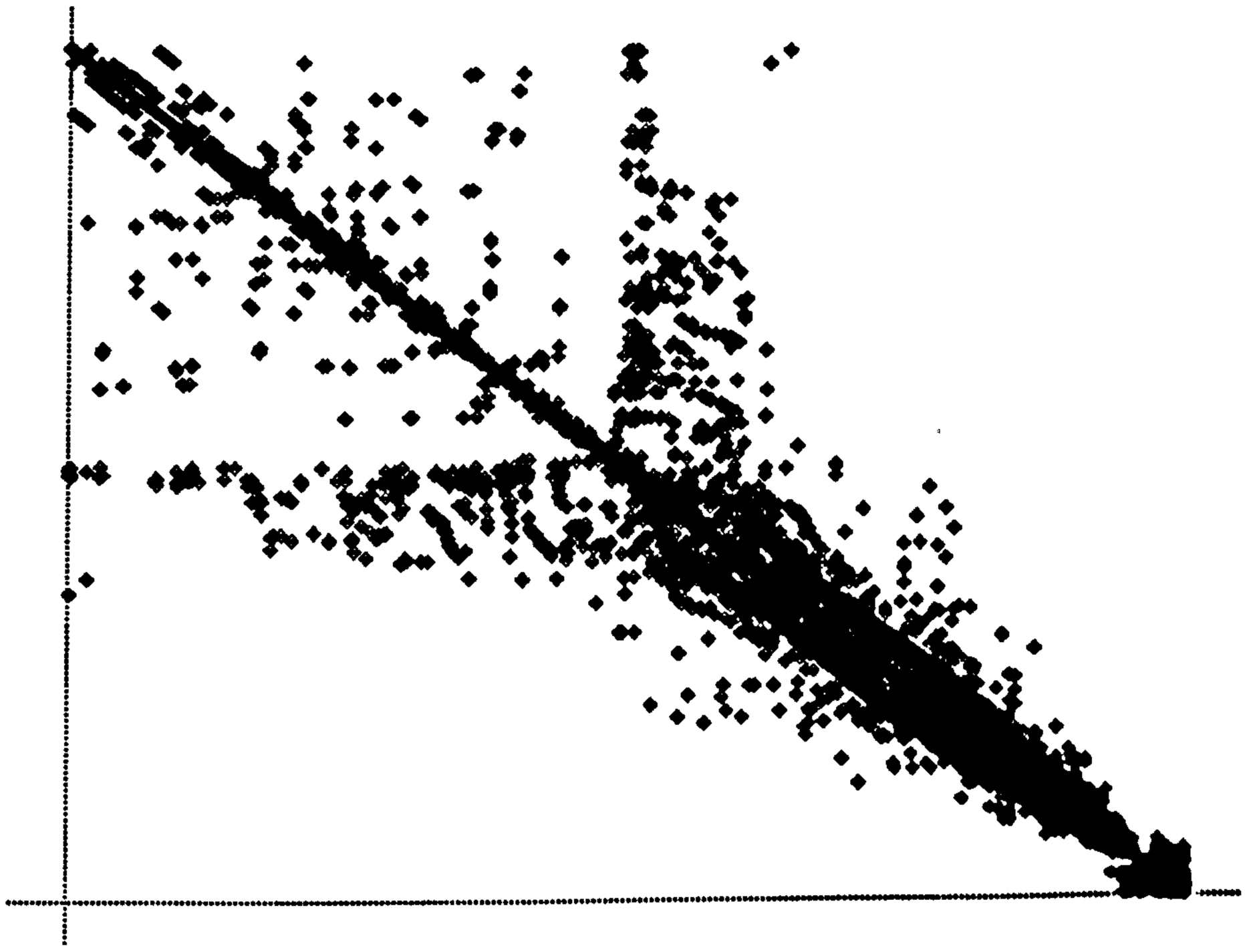


Fig. 48. Nonzero entries of Laplacian matrix from natural ordering

Fig. 49. Nonzero entries of Laplacian matrix after reverse Cuthill-Mckee ordering

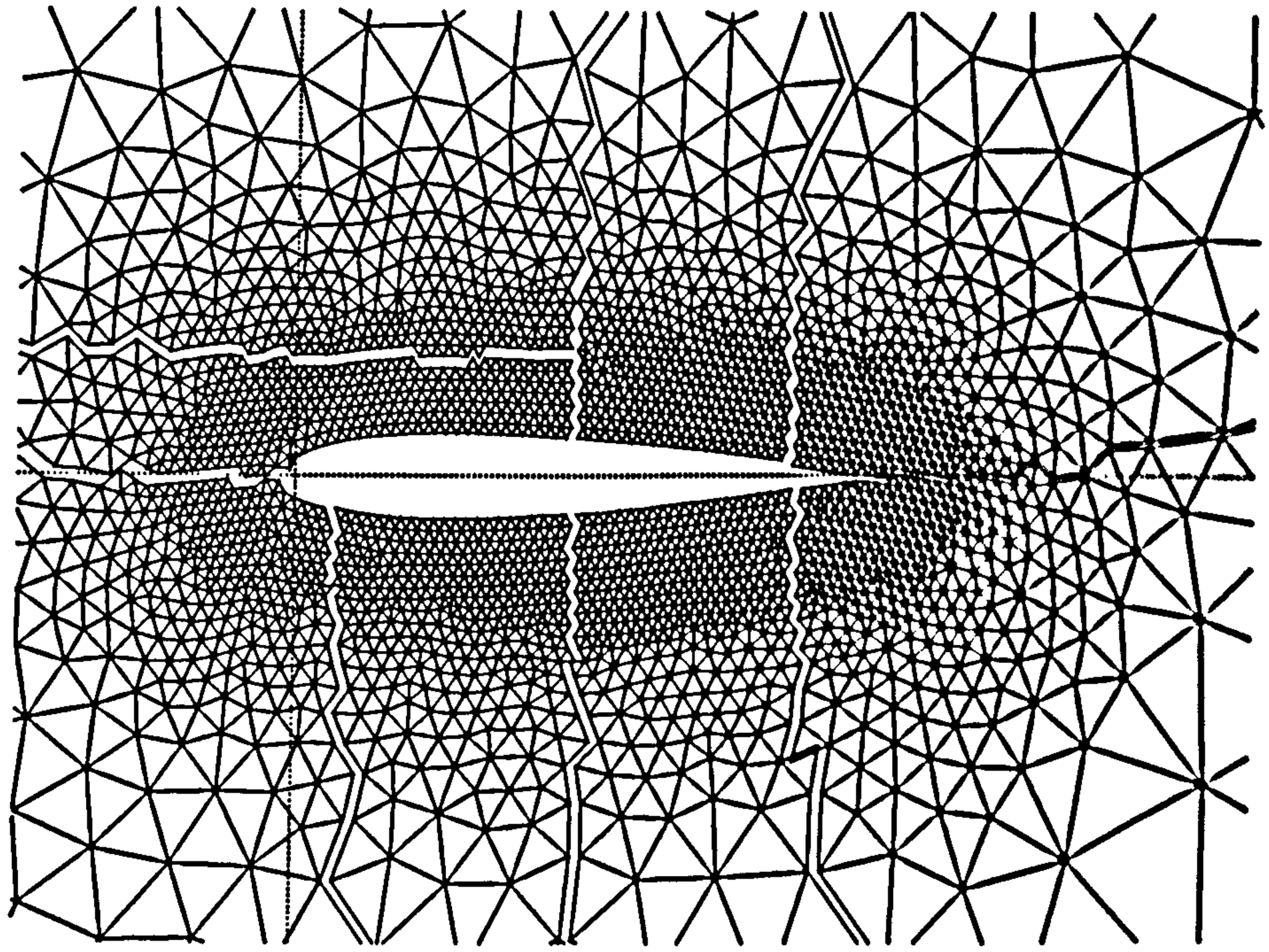


Fig. 50. RCB 8 sub-domain partitioning without smoothing

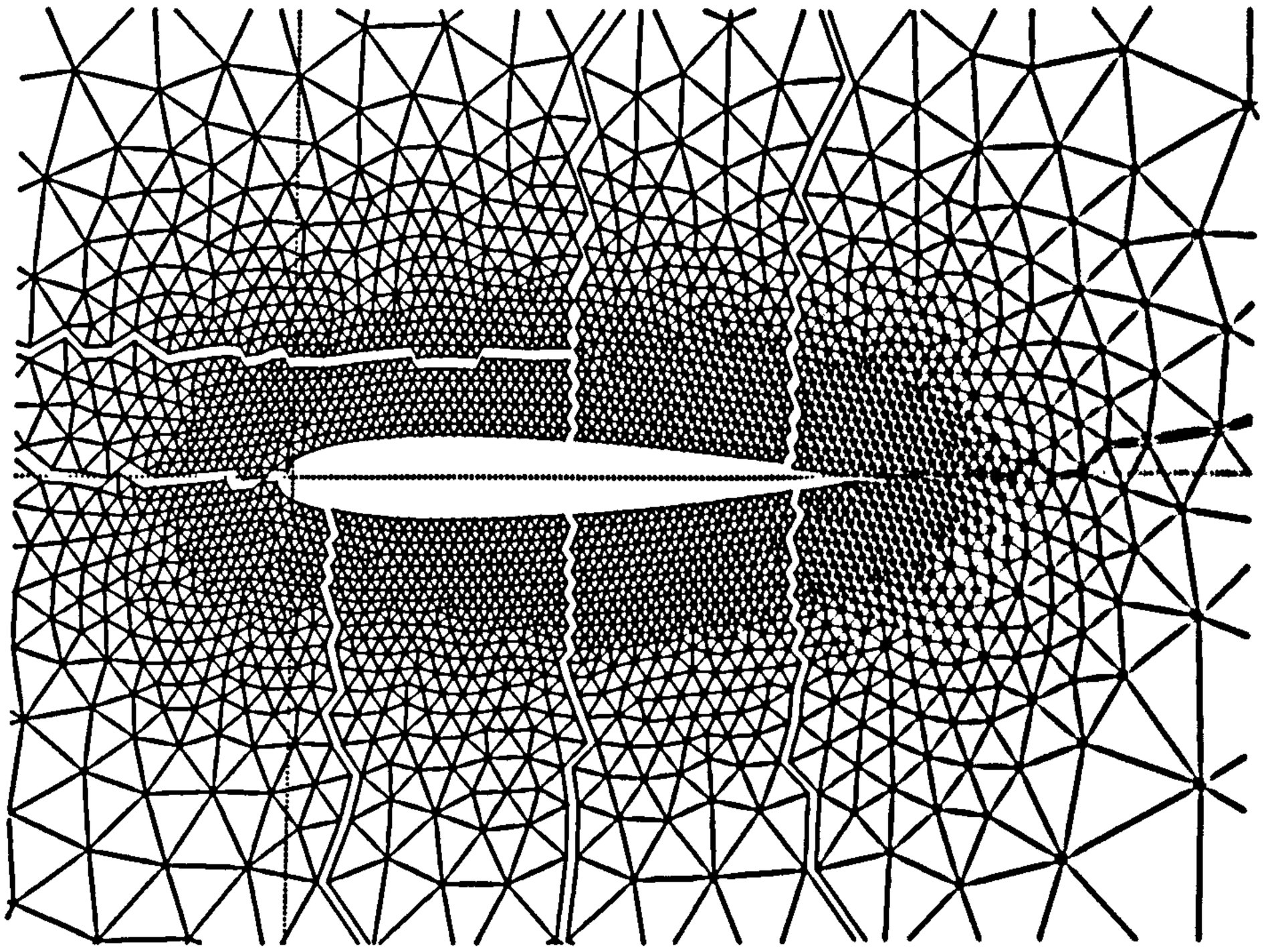


Fig. 51. RCB 8 sub-domain partitioning with smoothing

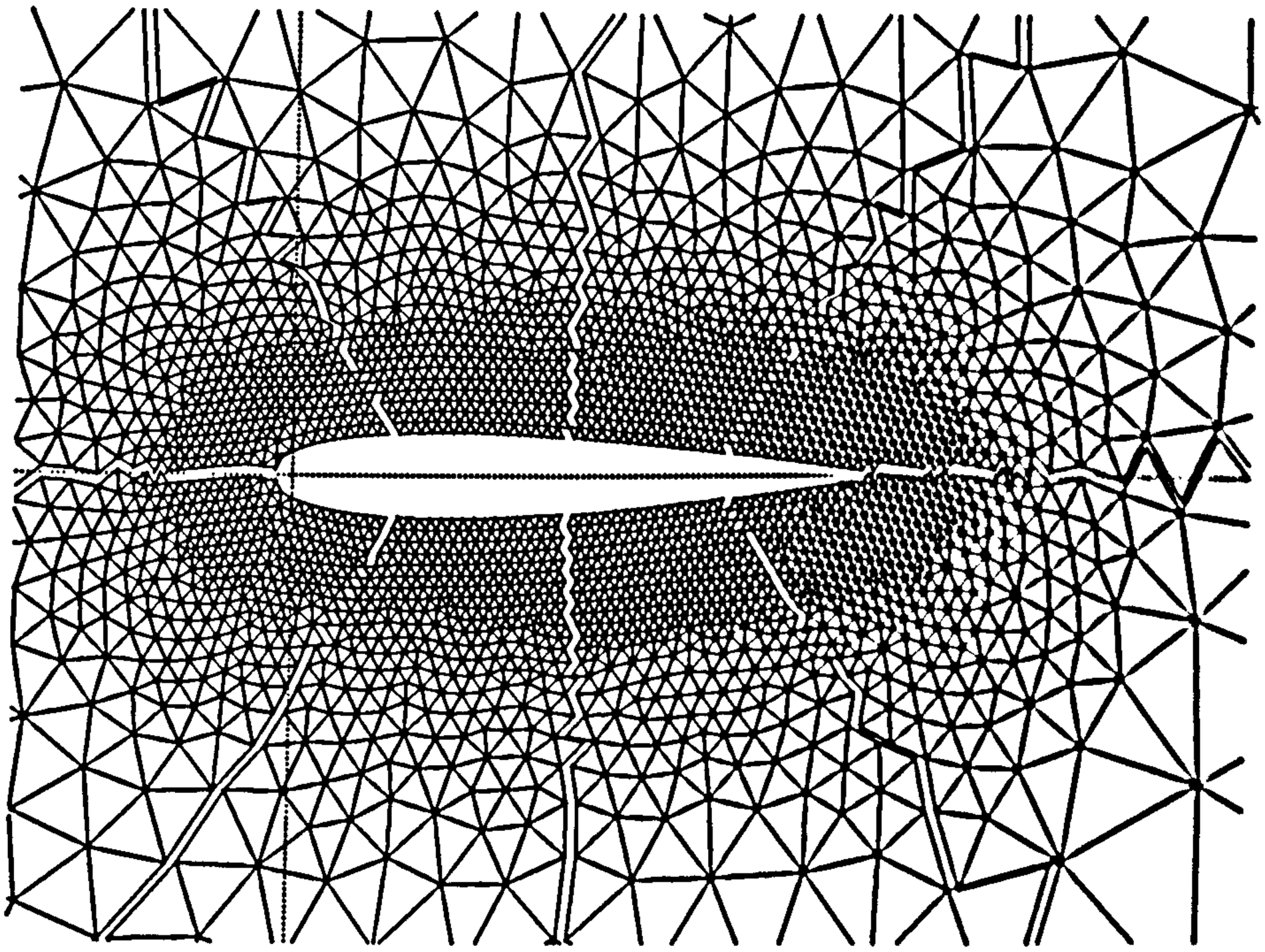


Fig. 52. RAB 8 sub-domain partitioning without smoothing

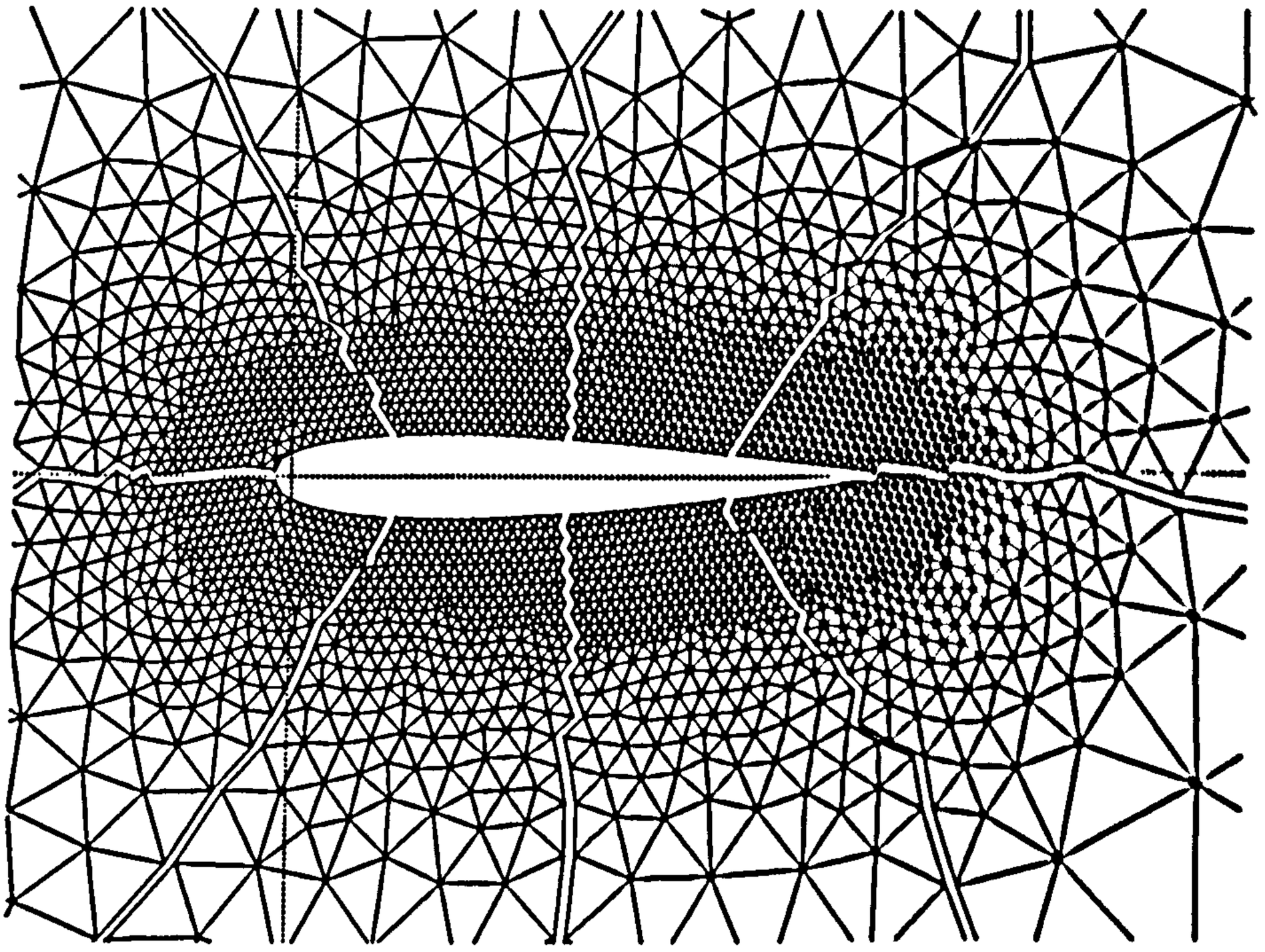


Fig. 53. RAB 8 sub-domain partitioning with smoothing

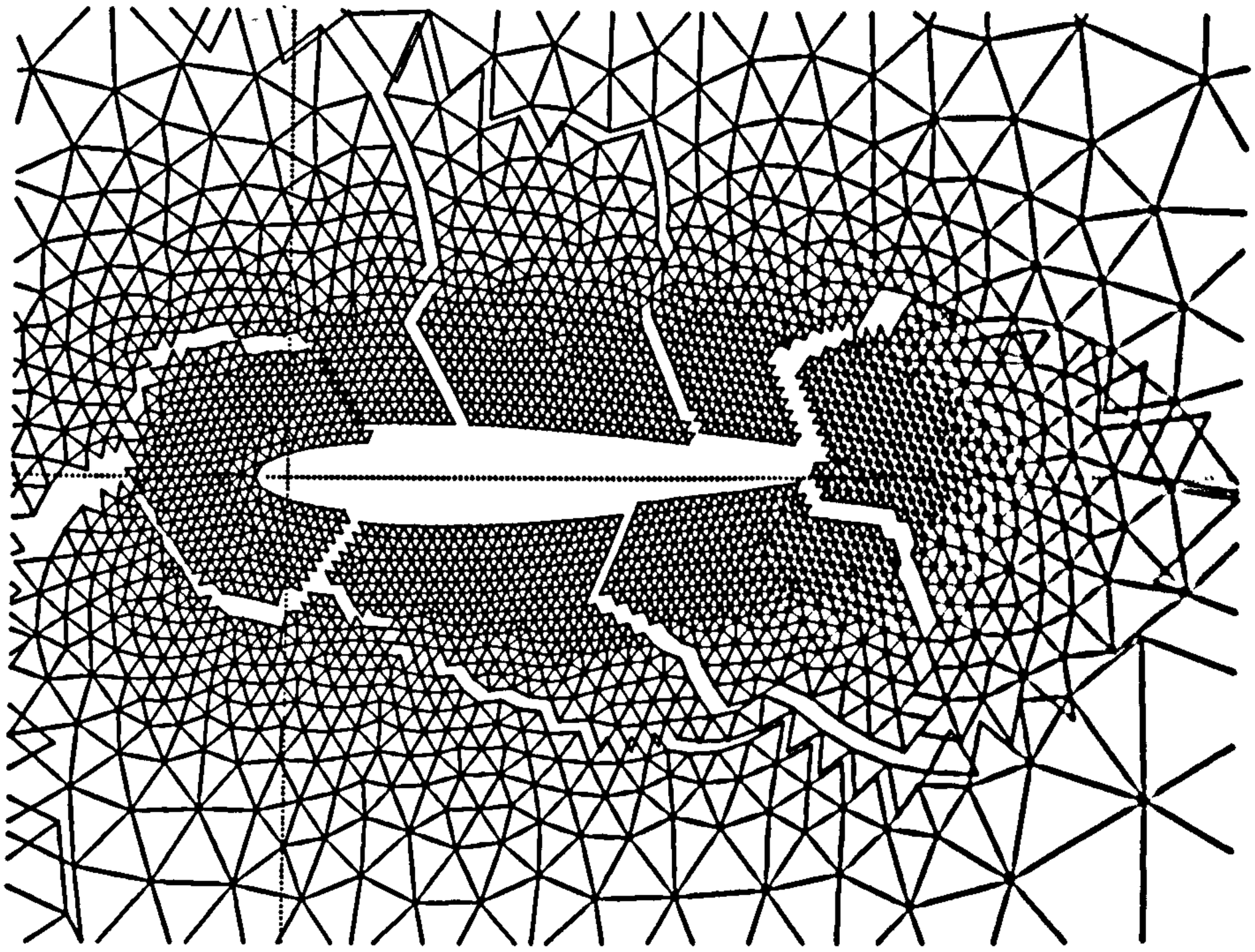


Fig. 54. RGB 8 sub-domain partitioning without smoothing and pre-ordering

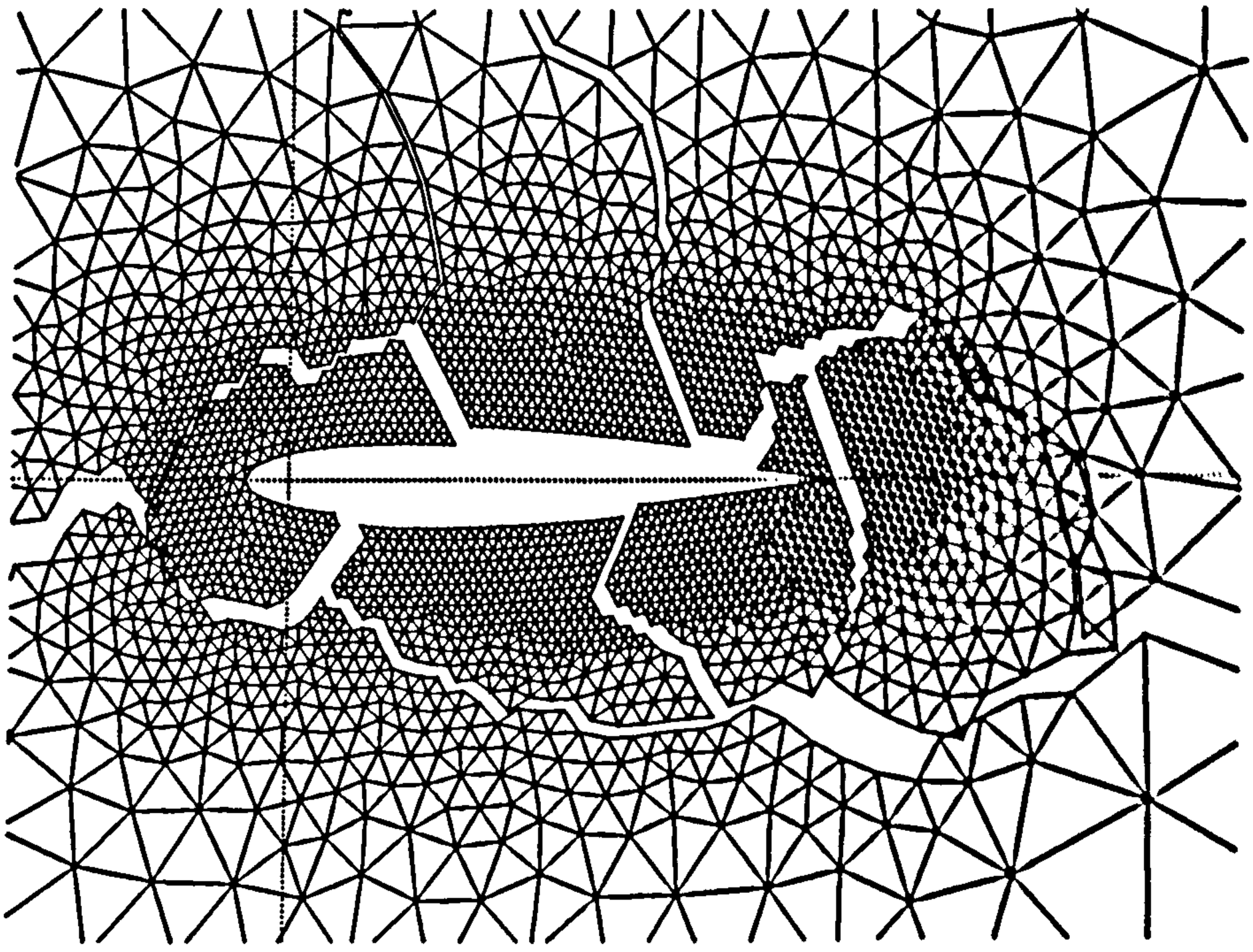


Fig. 55. RGB 8 sub-domain partitioning with smoothing and pre-ordering

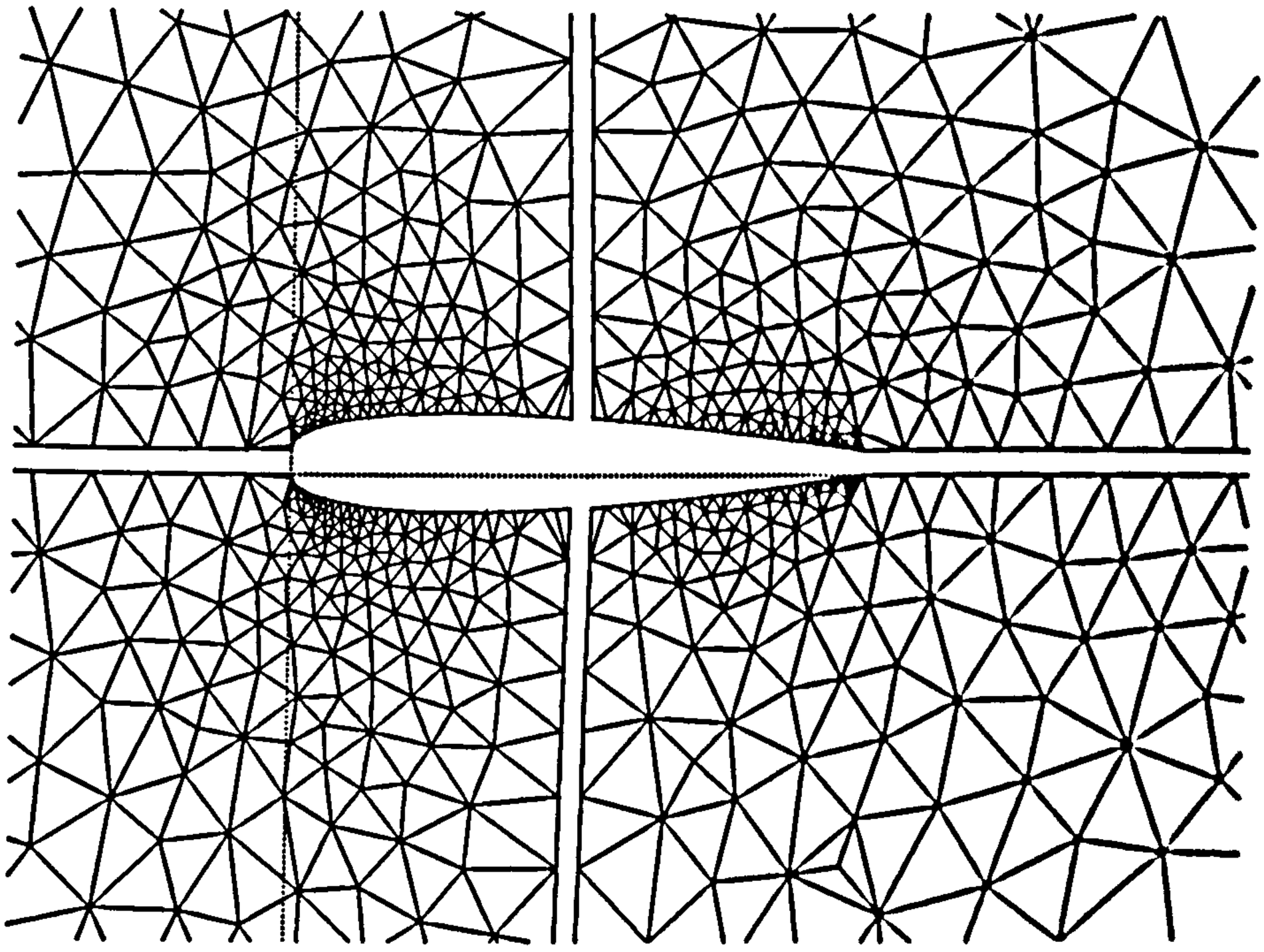


Fig. 56. DDT 4 partitioning with unstructured grids in sub-domain generated by Delauney triangulation method

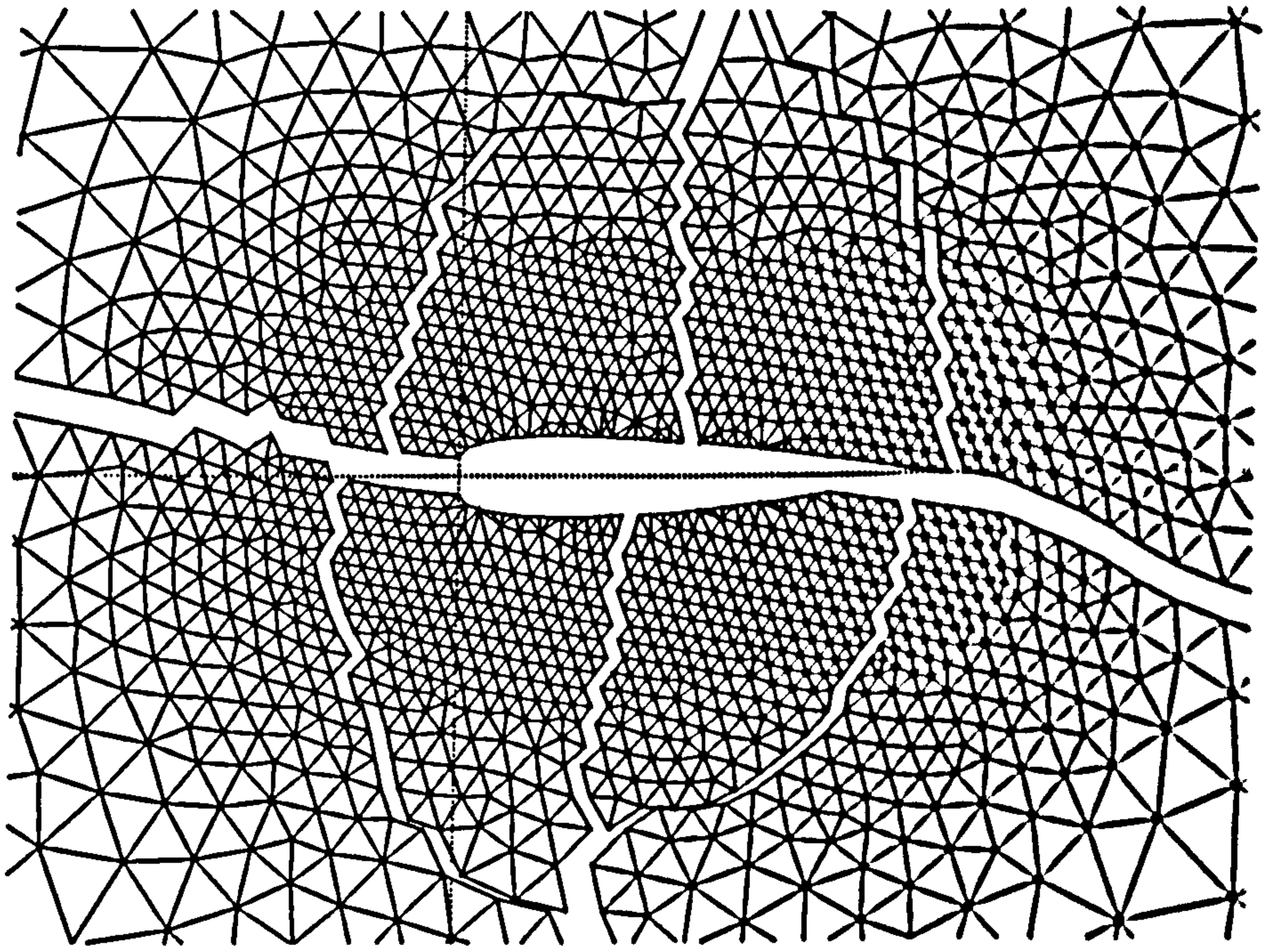


Fig. 57. RSB 8 sub-domain partitioning on NACA 0012 airfoil

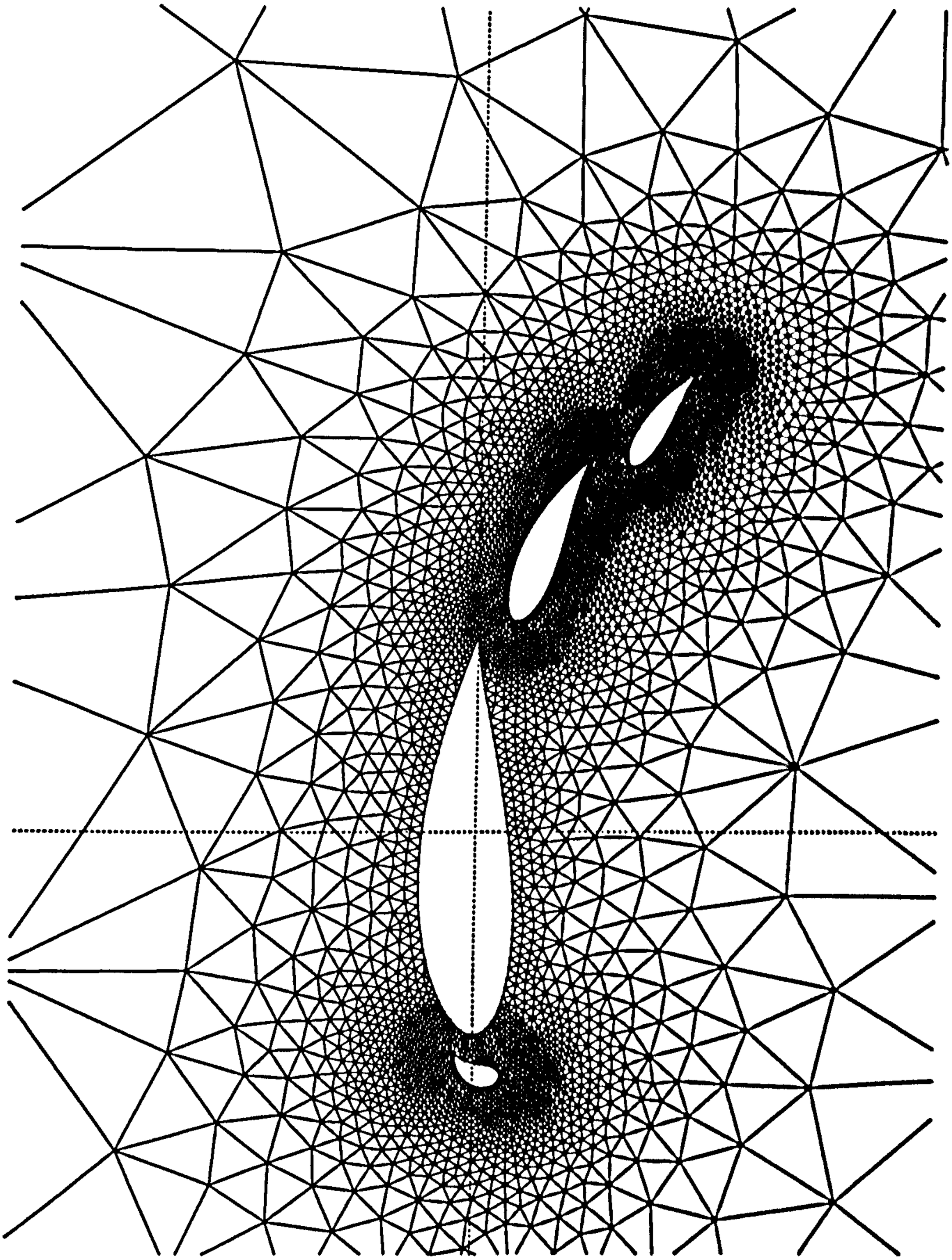


Fig. 58. Unstructured grids around subdoo 4-elements airfoil generated by AFT method

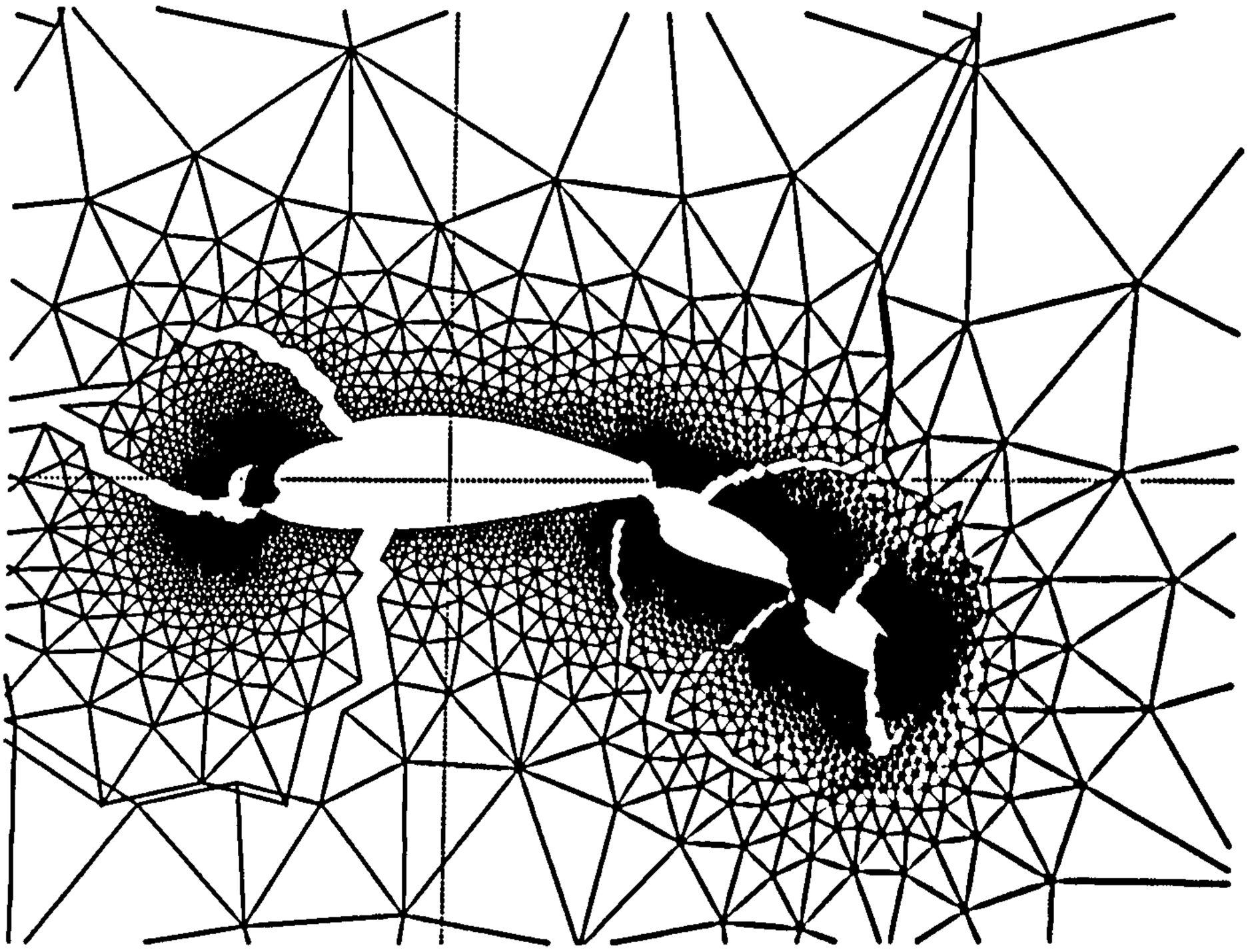


Fig. 59. RSB 8 sub-domain partitioning on subdoo 4-elements airfoil

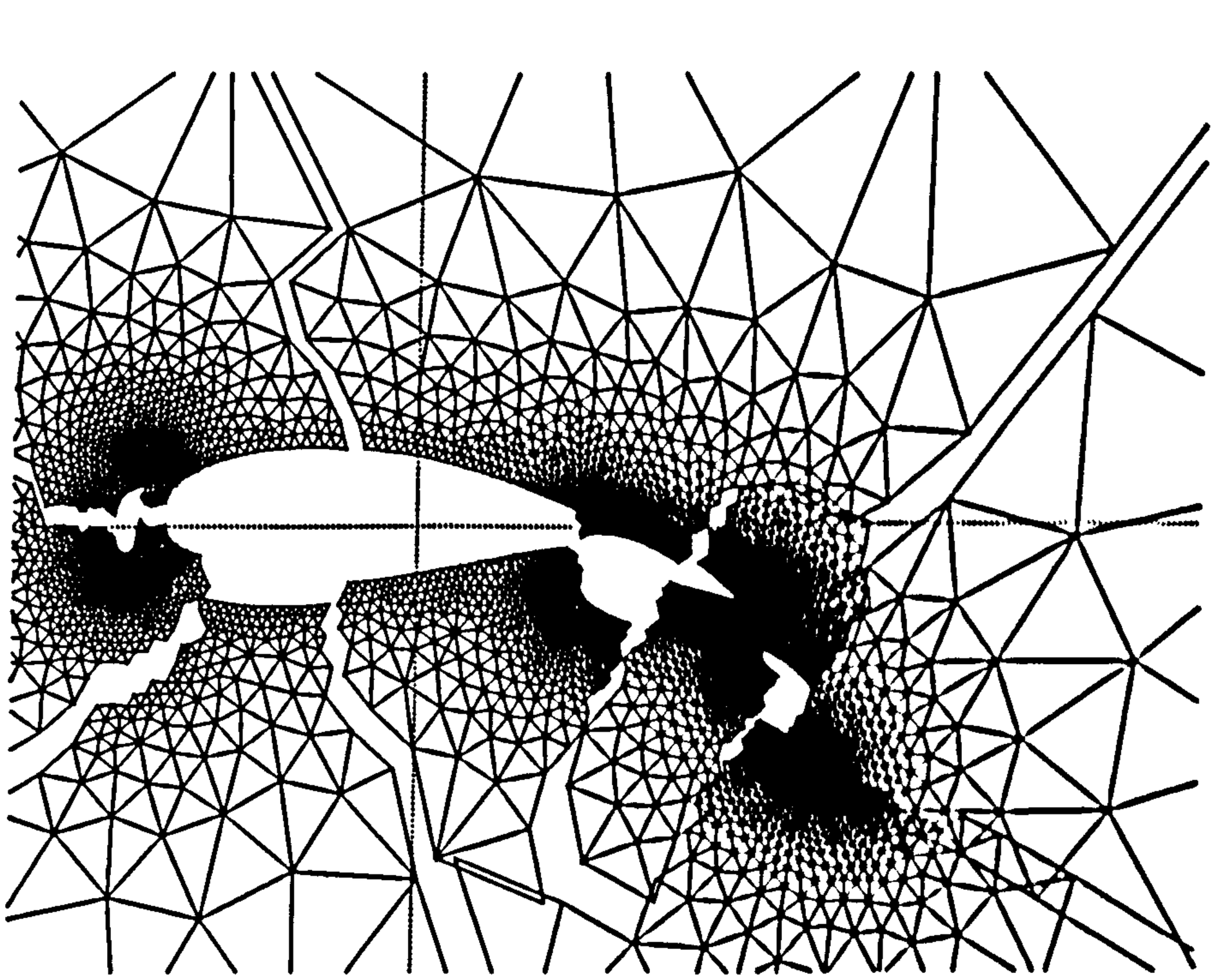


Fig. 60. MGP 8 sub-domain partitioning on subdoo 4-elements airfoil

CHAPTER 7

PARALLEL COMPUTING ON UNSTRUCTURED GRID

7.1. Introduction

Computational fluid dynamics (CFD) as its name implies is inevitably linked with computing problems, such as processing power, memory technology, networking and accessibility. To complete the simulation of fluid flow within limited wall-clock time is always the goal for every CFD researcher. In the last fifteen years CFD has obtained considerable benefit from the revolution happening in the computing science and the computer industry. The simulation of complex three-dimensional flows, which seemed impossible on sequential computers a few years ago, now becomes possible with the development of vector supercomputers. The appearance of distributed-memory parallel computers offers the next cost-effective leap forward in terms of computing power and size of memory.

Much research has been carried out in the field of unstructured grid computations on parallel platforms. Apart from the pre-processing procedures, i.e. the interface with the geometry producing package and parallel grid generation, and the post-processing procedures, i.e. parallel flow visualization and interface with other application softwares, the issues connected with the parallelism of the flow solver are grid partitioning, message communication, data structure and parallel algorithms. The partitioning of unstructured grids for parallel computing has been investigated by many researchers. The methods, as described in the previous chapter, can be broadly classified into geometry-based and graph-based algorithms. It appears that the graph-based algorithms, in particular, the recursive spectral bisection (RSB) technique of Pothen, Simon and Liou [66] and the multilevel graph partitioning (MGP) of Karypis and Kumar [99], produce improved partitioning results. After partitioning the next problem is message communication. Well-designed lower-storaged data structures need to be constructed. Message exchange between neighbouring sub-domains need be done at speed during the iteration. Finally the CFD code need be paralleled.

Unstructured grid flow solvers have been implemented on various parallel machines [74], [76], [78] and [102]. These studies have shown that good performance may be obtained by paying careful attention to the issues mentioned above. In particular flow solvers with explicit and point implicit schemes possess almost complete parallelism in their nature. They require only simple update procedures that in-

volve local dependencies. On a parallel computer, such schemes typically require the communications only between the first nearest elements (in case of inviscid first order schemes considered), the first and second nearest elements (in case of viscous first order schemes considered) and all three types of nearest elements (in case of high order scheme considered). The definition of the nearest element is given in figure 62. Mavriplis, Das, Saltz and Vermeland [102] reported the impressive performance using explicit schemes with the multigrid method. Similar researches have been done by Venkatakrishnan, Simon and Barth [76] and Hammond and Barth [103] using an explicit scheme with the finite volume method on an unstructured grid. On the other hand the implicit scheme requires the solution of coupled equations which involves global dependencies. Thus the design of an implicit scheme on distributed-memory parallel computers is a little more difficult. Researches of Johan et al [74] and Venkatakrishnan [78] have shown that the implicit schemes can be designed carefully to produce good performance when solving unstructured grid problems on parallel computers.

Following this, in section 7.2 we discuss briefly the partitioning issue of unstructured grids. Then in section 7.3 the message communication patterns together with the data structure are described. In section 7.4 the issues involved in parallelizing cell-centred finite volume schemes for solving the Navier-Stokes equations on triangular unstructured grids on a workstation cluster — a parallel computer system based on a workstation network under the parallel environment supported by parallel virtual machine (PVM) software is explained. Section 7.5 gives some application examples and section 7.6 makes some conclusions to this chapter.

7.2. Partitioning problems

An efficient partitioning of unstructured grids for a distributed memory machine is one that ensures an equal distribution of computational workload among the processors (normally termed *load balancing*) and minimizes the amount of time spent in inter-processor communications. Thus the total execution or wall clock time including the sum of the time required for computation and communication will be minimized.

In most CFD solvers with cell-centred schemes on unstructured grids, the computational time is typically a function of the number of elements and sometimes the shape of the domain as well. The computational work involved in the computation of residuals is directly proportional to the number of edges which is linearly related to the number of elements in that domain. In the case that the load were not equally distributed, some processors will have to sit idle and waste time waiting for other processors to catch up.

The actual cost of communication between processors can often be accurately modelled by the linear relationship as

$$\text{Communication Cost} = \alpha + \beta m \tag{7.1}$$

where α is the time required to start-up (also known as latency) a message, β is the rate of data-transfer between two neighbouring processor and m is the length. For n messages, the total cost should be

$$\text{Total Communication Cost} = \sum_{n}(\alpha + \beta m) \tag{7.2}$$

Obviously this cost can be reduced in two ways. One is to reduce the number of messages to be communicated. The other is to reduce the individual message lengths. In practice it is difficult to partition the unstructured grids while simultaneously minimizing the number and lengths of messages. On most modern parallel computers, the latency is small enough that minimizing the number of neighbours is not necessary. Hence most of the partitioning methods follow the definition of minimizing the length of messages under the condition of load balancing.

The partitioning methods, e.g. recursive type algorithms (RCB, RGB and RSB) and multilevel graph partitioning (MGP), will create subdomains which have balanced computational loads in each processor. It should be noted that our particular application is on parallel CFD computing using the cell-centred finite volume schemes with unknown variables at the centre of the element. Hence to partition the domain into sub-domains means to separate the elements. Also note that whatever the partitioning algorithms are used the resulting sub-domains will be flagged with integer numbers. Those elements with same flag number mean they belong to the same sub-domain. Those edges shared by elements with different flag numbers imply they are the interfaces of the sub-domains.

The discussion and application of RCB, RGB, RSB and MGP on partitioning the two-dimensional airfoil cases can be found in the previous chapter. Here only some comments are given. Normally the RCB gives extreme efficiency as the sorting is done in $logN$ operations, where N is the total number of vertices. But such techniques sometimes lead to disconnected subdomains which imply higher cost for message transfer. The RGB technique by level sets (e.g. Cuthill-Mckee algorithm) produces partitions with long boundaries because it uses a breadth-first search to define the level sets. The RSB technique produces uniform, mostly connected sub-domains with short boundaries. Theoretical results by *Fielder* (summarized in [101]) show that one of the two sub-domains formed by RSB partitioning is always connected. As the spectral

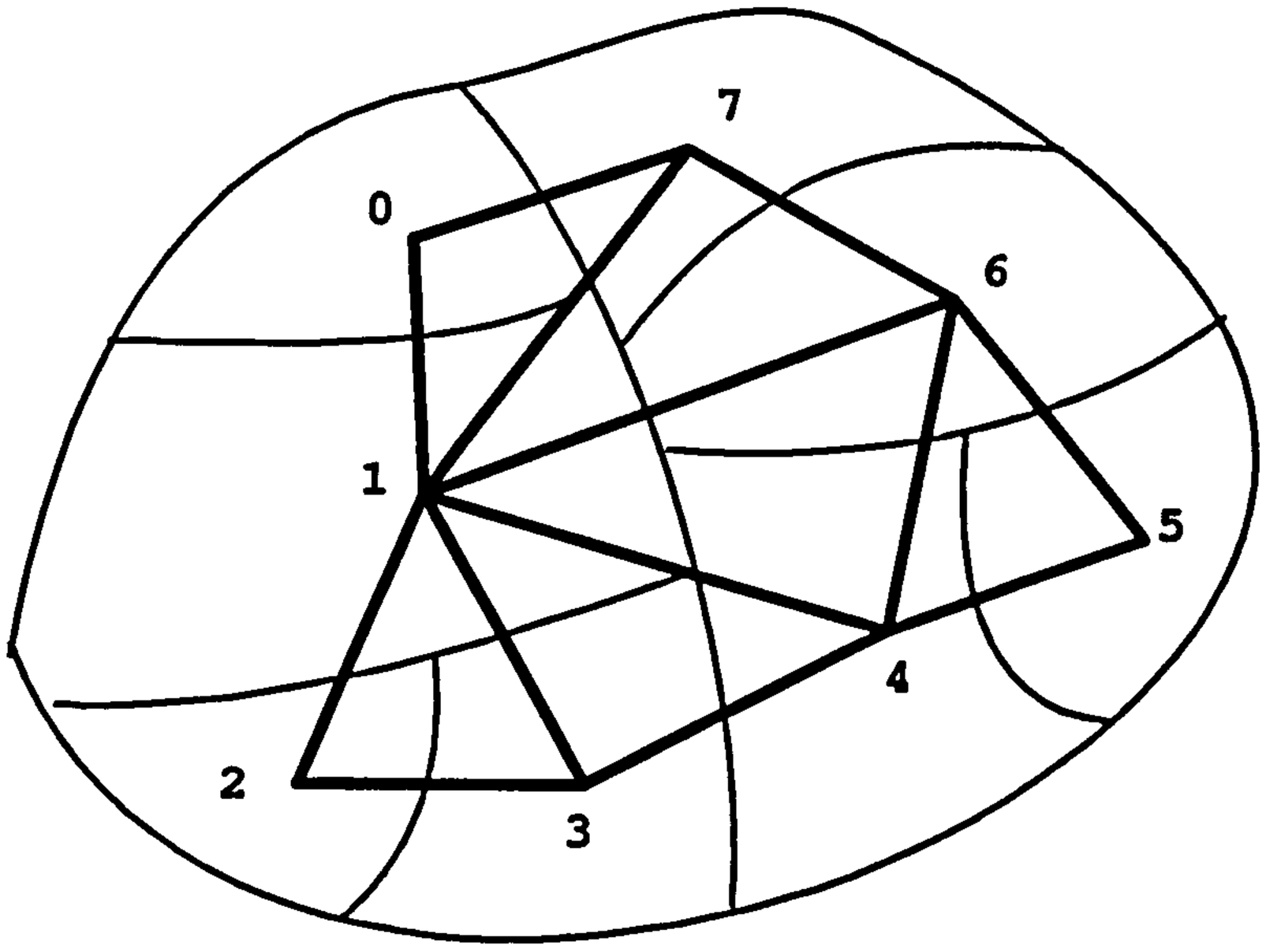


Fig. 61. 8-way partitioning of domain and its associated partition communication graph

partitioning results in shorter length messages it will reduce the communication costs. However the RSB method is expensive due to the use of the computation of the *Fielder* vector. Improvements have been made by Barnard and Simon [68] of multilevel spectral bisection (MSB). A different partitioning strategy based on multilevel graph partitioning (MGP) method has been investigated by Karypis and Kumar [99]. The developed package Metis V2.0 achieves the same quality as that obtained by RSB and MSB methods but the execution time is reduced by nearly one order.

7.3. Message communication

A global element-node connectivity data structure is used to define the unstructured grid. After partitioning, each element is coloured with a flag number. Those elements with the same flag number belong to same the sub-domain and will be posted to one processor. For simplicity we keep the global data structure and index unchanged, thus the local data sets can be derived with the global indices. The relation of sub-domains can be illustrated by the communication graph of figure 61.

Table XVII shows the adjacent processor listing in the communication graph of figure 61 and Table XVIII presents the message communication relations among the sub-domains (processors) in figure 61. It means that, for example, processor 1 has six adjacent processors (0, 2, 3, 4, 6 and 7). The message communication will be done between adjacent processors.

Processor	Adjacent Processor
0	1 7
1	0 2 3 4 6 7
2	3 1
3	2 4 1
4	5 3 6 1
5	4 6
6	7 5 4 1
7	6 0 1

Table XVII. Adjacent processor list in communication graph

Processor	Permuted Processor					
0	1	7	-	-	-	-
1	0	2	3	4	6	7
2	3	1	-	-	-	-
3	2	4	1	-	-	-
4	5	3	6	1	-	-
5	4	6	-	-	-	-
6	7	5	4	-	1	-
7	6	0	-	-	-	1

Table XVIII. Message communication relations in communication graph

During implementation, each local data set must contain two kind of communication information, i.e. (1) those messages required to be sent to other processors sharing common interface boundaries; and (2) those messages needed to be received from other processors sharing common interface boundaries, thus the computation problem within each processor becomes well-defined and the iteration can proceed. Those sending and receiving messages in each processor for communication consist of the following:

(i)NADJPROC: Number of ADJacent PROCessors (processors handling the adjacent partitions);

(ii)NBVS: Number of Boundary Vertices in common interface with processor NADJPROC that need to Send message;

(iii)NINTBVS(*,1): Local indices for the vertices sending message on present processor, length NBVS(NADJPROC);

(iv)NINTBVS(*,2): Local indices on adjacent processor receiving information, length NBVS(NADJPROC);

(v)NBVR: Number of Boundary Vertices in common interface with processor NADJPROC that need to receive message;

(vi)NINTBVR(*,1): Local indices for the vertices on present processor receiving message, length NBVR(NADJPROC);

(vii)NINTBVR(*,2): Local indices on adjacent processor sending information, length NBVR(NADJPROC).

The data structures will be illustrated by three examples, which represent three communication patterns, i.e. inviscid first order, viscous first order and high order patterns. At first we will give the definition of three types of nearest neighbour elements. The first-nearest neighbour elements are those sharing the common edge with elements belonging to different sub-domains (denoted by s1 and r1 in figure 62). The second-nearest neighbour elements are those sharing the common node points with elements belonging to different sub-domains (denoted by s2 and r2 in figure 62). The third-nearest neighbour elements are those holding the node opposite to the common edge with elements belonging to different sub-domains (denoted by s3 and r3 in figure 62). It is thus clear that for the flow solver with a cell-centred finite volume scheme on an unstructured grid, only the first-nearest elements need to be sent and received in case of inviscid first order scheme; both first and second nearest neighbour elements need to be exchanged between adjacent processors for viscous flow with a first order scheme and all three types of nearest neighbour elements are required to communicate between adjacent processors in case of high order schemes

Pattern 1	NADJPROC = 2,3	NBVS = 16,16	NBVR = 16,16
Pattern 2	NADJPROC = 2,3	NBVS = 32,32	NBVR = 36,36
Pattern 3	NADJPROC = 2,3	NBVS = 52,52	NBVR = 68,68

Table XIX. Three patterns of data structure for processor 1

used.

Figure 62 shows a three-way partition with the interface boundaries indicated by thick lines. Each of the elements shown is stored by two or three processors depending on which case is considered. The entries of the data structures for processor 1 are shown in table XIX.

It can be seen for communication pattern 1 (inviscid first order case) the sending and receiving data appears in pairs as the element sharing the common edge always has one-to-one correspondent relations. But for pattern 2 (viscous first order case) and pattern 3 (high order case) the numbers of sending and receiving are normally different.

It should be noted that the partitioning procedure is separated from the flow solver as a pre-processing step. The resulting flag file will then be read by in the solver code. As we keep the global address unchanged so no local address is used. The flow solver is constructed in an edge-based data structure so that each edge also needs to be flagged. This is done before the do-loop over the edges begins. Finally the data structure required for communication at the interface of adjacent processors is set up before the do-loop started.

7.4. Parallelism of the flow solver

The parallel implementation for the developed Navier-Stokes flow solver described in chapters 2, 3, 4 and 5 is completed under the environment of the parallel virtual machine (PVM) package installed on the workstation cluster of the Department of Aerospace Engineering at Glasgow University. In the following we will first briefly describe the PVM software. Then the implementation procedure will be described.

PVM (parallel virtual machine) is designed as a software package which permits a heterogeneous collection of serial, parallel and vector computers which are linked to a network to appear as one large computing resource. Hence PVM collects the power of a number of computers and may be used to solve the previously unsolvable "grand challenge" problems. Overall PVM has the following main features: (1) it provides a

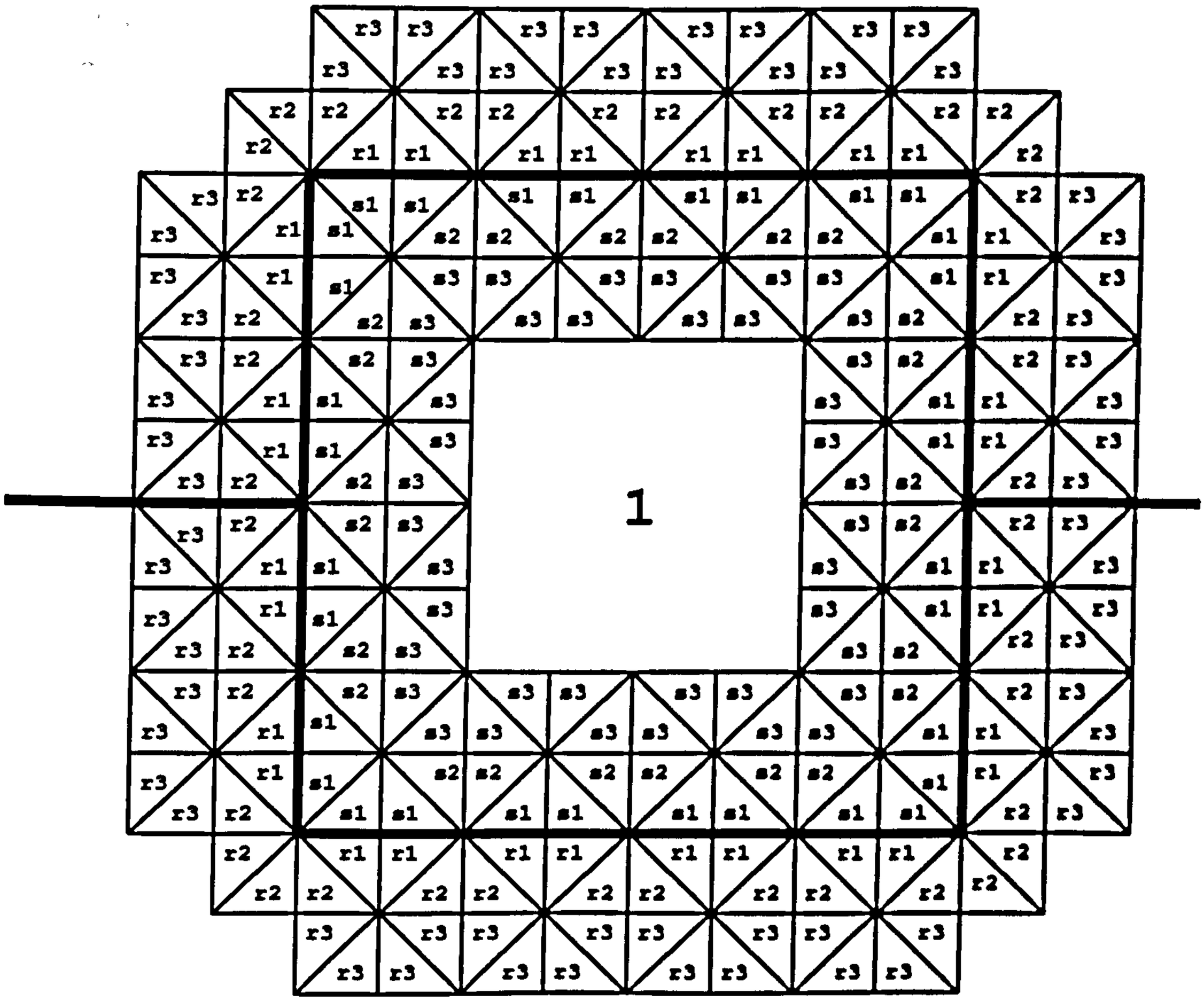


Fig. 62. Three-way partitioning and sending and receiving elements along the interface boundary for processor 1

unified framework within which parallel programs can be developed efficiently using existing hardware; (2) it transparently handles all message routines, data conversions and task schedulings across a network of incompatible computer architectures; (3) the programming interface is straightforward allowing simple program structures to be implemented in an intuitive manner; and (4) the user can easily write his own application as a collection of co-operating tasks which access PVM resources through a library of standard interface routines. For parallelising CFD codes, the most useful routines in PVM involve sending and receiving messages between adjacent processors. In PVM sending a message includes a three step process: (1) initialize a buffer (a set of messages) using the pvmfinitsend; (2) pack the data into the buffer using pvmfpack; and (3) send the contents of the buffer to another process using pvmfsend or to a number of processes (multi-cast) using pvmfmcast. The receiving a message is a two step process: (1) call the blocking routine pvmfrecv or the non-blocking routine pvmfnrecv; and (2) unpack the message buffer using pvmfunpack. A detailed description of the PVM software and its routines can be found in [104].

The implementation procedure is illustrated as follows. When the code is first started it needs to define a host computer (also termed the parent computer) by calling the routine "pvmfmytid". Then an identification number is given by calling "pvmfparent". After that the parent computer will spawn a few child computers depending on the number of processor required. In parallel computing there are two ways. One is called the master-slave model. That means that master machine (or parent computer) assigns the work to each slave machine (or child computer) and collects the resulting messages from the slave machine. The message passing is only done between master and slave. There is no communication between slaves. The drawback of this method is that the master will be idle when the slave machine is working. The other one is called the master-master model. This means that every machine including the parent and child computers are all the master. They do the work within their own domain and communicate with each other. On load-balanced partitions this method can make use of the computing power providing great efficiency.

The master-master model is used in present research. The flow solver is constructed based on the unstructured grid by the cell-centred finite volume method. The efficient edge-based data structure is used. The flux computation is carried out on edges. Following the input of the flag file from the partitioning preprocessing step, the process of flagging the edge needs to be done before the loop over the edge begins. The construction of the message passing data structure is then done. As discussed in the previous section the data sets can be built in three patterns depending on the computational cases considered, i.e. inviscid first order, viscous first order and high

order cases. The message buffer of the viscous first order case is nearly twice that in the inviscid first order cases and the message in the high order case is nearly four times that required in the inviscid first order case. Before the edge loop begins, all necessary data need to be communicated between processors by calling the message sending and receiving routines. Then the computational problem on each processor is completely defined and can start the loop over the edge within the sub-domain with which it is dealing. After the calculation on all edges is over, the values at the centre of the elements will be updated within each processor and the message passing routine (sending and receiving) will be called again to update the data in the data structure pattern. The iteration, update and message passing will continue until the convergence criteria is reached. Finally the parallel process will be ended by calling "pvmfexit".

7.5. Application of transonic airfoil parallel computing problems on work-station cluster

We consider the example of a transonic flow past a NACA 0012 airfoil with a freestream Mach number $M_\infty = 0.75$ and an angle of attack of 2^o. The performance is presented for one mesh size with four different cases, i.e. inviscid flow with the first order scheme, viscous flow with the first order scheme, inviscid flow with the high order scheme and viscous flow with the high order scheme. The mesh contains 6354 triangles, 3267 vortices and 9441 edges. The mesh and its computational results for the inviscid first order case by sequential code can be found in chapter 3. Here we consider the same example using the parallel code.

The partitioning methods used are recursive spectral bisection (RSB) and multilevel graph partitioning (MGP). The quality (the number of edge cuts) and the efficiency (CPU time) for partitioning this problem with RSB and MGP are given in tables XX and XXI. The results in table XX demonstrate the same good quality achieved both by the RSB and MGP methods. The efficiency of the MGP method is higher than that of the RSB method (see table XXI). For example the MGP method is nearly four times faster than the RSB method when 8 partitioning considered.

Figure 63 gives the communication graphs for k-way (k=2, 3, 4, 5, 6 and 8) partitioning by the RSB method for the present example. Also figure 64 shows the communication graphs for k-way (k=2, 3, 4, 5, 6 and 8) partitioning by the MGP method. It is clear that different partitioning methods result in different communication graphs.

The message communication data structure is constructed in the way as described in section 7.3. Here only the results of 8-way partition data structure are illustrated

Method	Partition					
	2	3	4	5	6	8
RSB	49	100	118	153	180	231
MGP	55	99	128	148	183	222

Table XX. The number of edge cuts E_c on NACA 0012 airfoil with 6354 triangles, 3267 vortices and 9441 edges

Method	Partition					
	2	3	4	5	6	8
RSB	0.56	1.021	1.143	1.388	1.488	1.674
MGP	0.32	0.34	0.37	0.39	0.40	0.42

Table XXI. The CPU time (seconds) for partitioning on NACA 0012 airfoil with 6354 triangles, 3267 vortices and 9441 edges

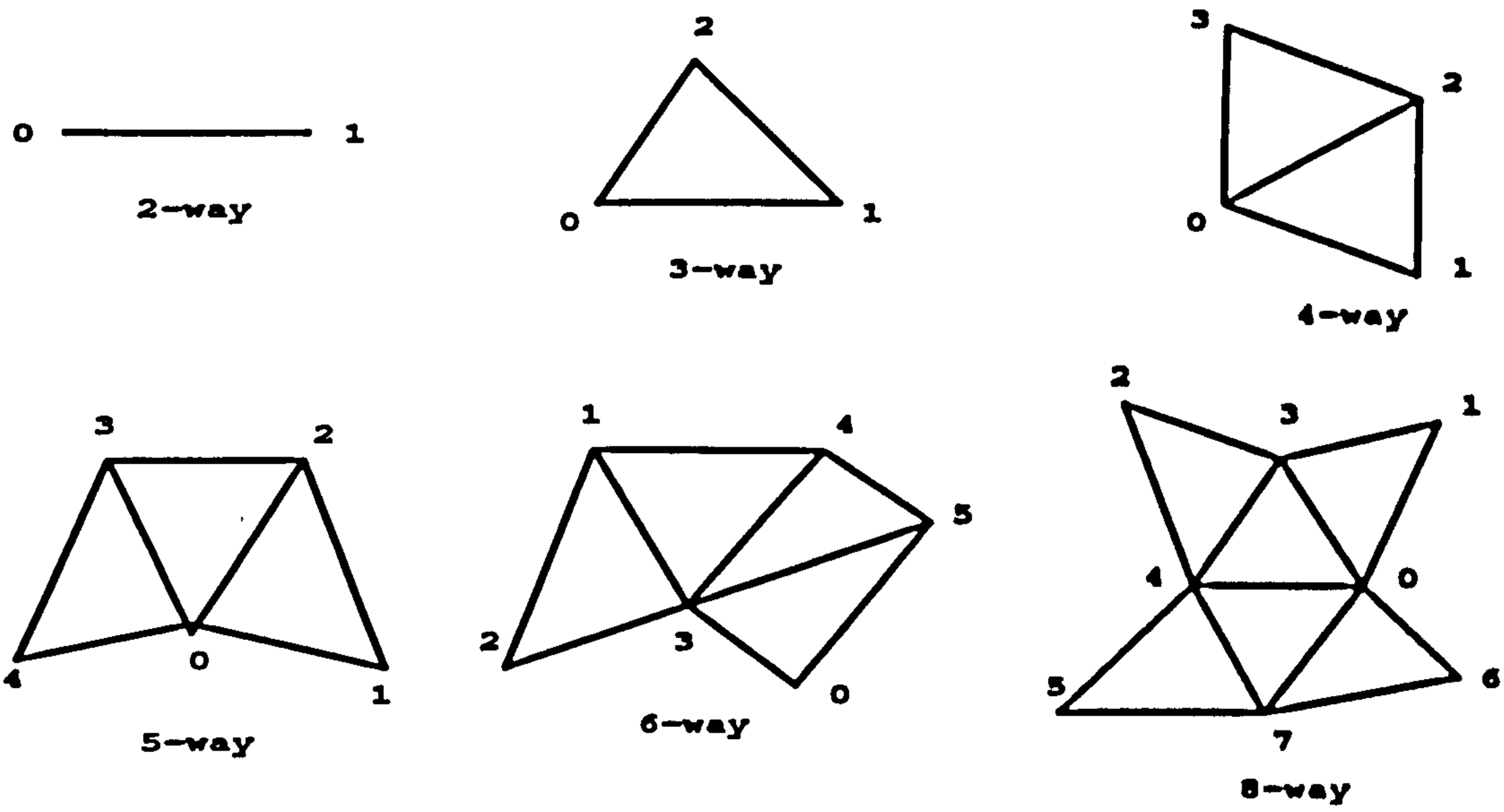


Fig. 63. k-way partitioning of domain and its associated partition communication graph by RSB method

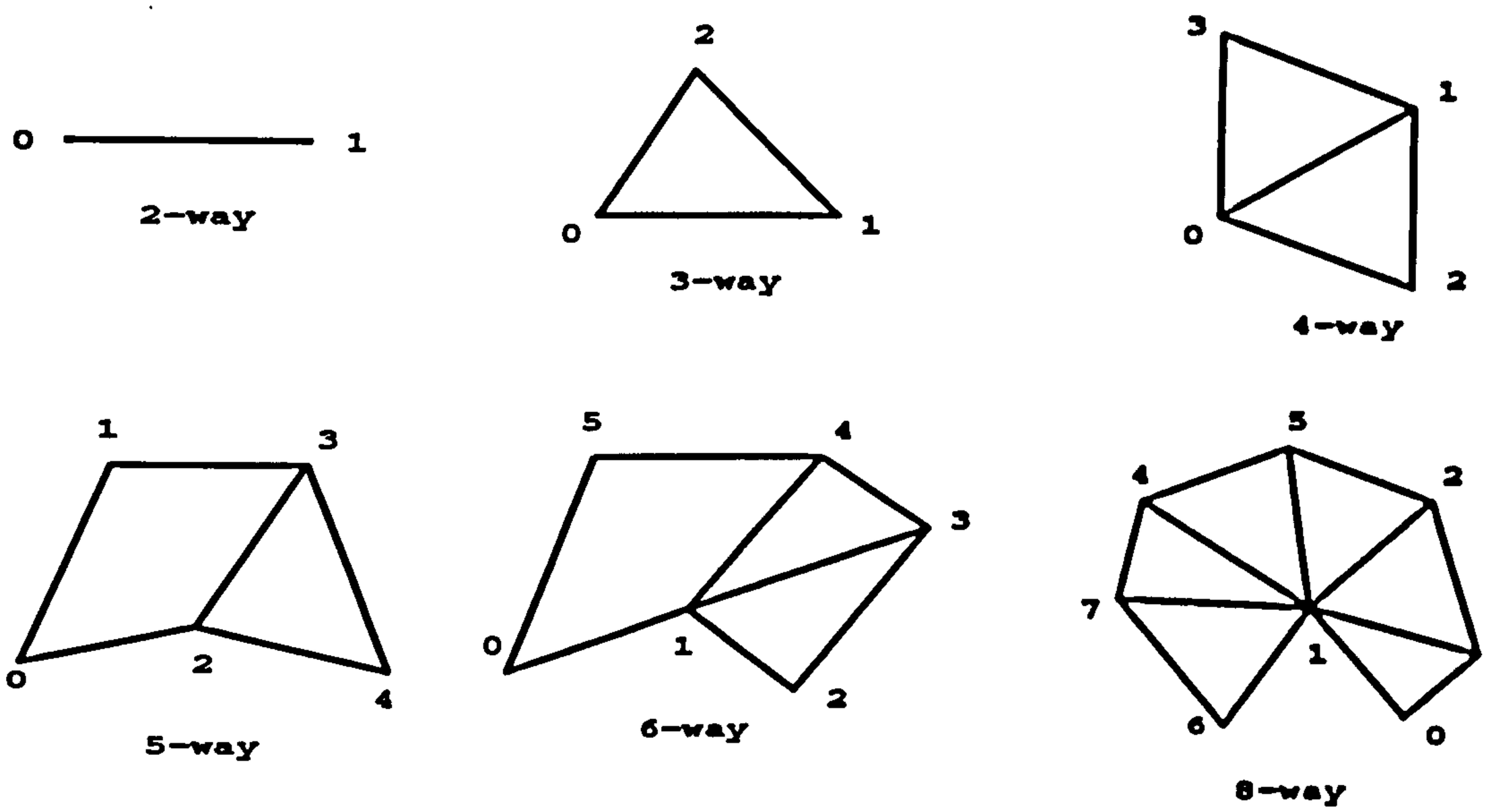


Fig. 64. k-way partitioning of domain and its associated partition communication graph by MGP method

by tables XXII, XXIII, XXIV, XXV, XXVI, XXVII, XXVIII, XXIX, XXX, and XXXI for all test cases. Table XXII represents the message communication between 8 processors for the inviscid first order case using the RSB partitionings. It means that, for example, processor 0 needs to send (or receive) 28 messages to (or from) processor 1; no message communications between processor 0 and processor 2; processor 0 needs to send (or receive) 4 messages to (or from) processor 3; processor 0 needs to send (or receive) 2 messages to (or from) processor 4; no message communications between processor 0 and processor 5; processor 0 needs to send (or receive) 21 messages to (or from) processor 6; and processor 0 needs to send (or receive) 3 messages to (or from) processor 7. As the inviscid first order case considered, the messages of sending and receiving are the same. Table XXIII and XXIV represent the message communication between 8 processors for the viscous first order case using the RSB partitionings. It can be illustrated that, for example, processor 0 needs to send 55 messages to processor 1 but receive 57 messages from processor 1; no message communications between processor 0 and processor 2; processor 0 needs to send 8 messages to processor 3 but receive 9 messages from processor 3; processor 0 needs to send 6 messages to processor 4 but receive 5 messages from processor 4; no message communications between processor 0 and processor 5; processor 0 needs to send 46 messages to processor 6 but receive 45 messages from processor 6; and processor 0

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	28	-	4	2	-	21	3
1	28	-	-	31	-	-	-	-
2	-	-	-	35	13	-	-	-
3	4	31	35	-	10	-	-	-
4	2	-	13	10	-	27	-	9
5	-	-	-	-	27	-	-	25
6	21	-	-	-	-	-	-	23
7	3	-	-	-	9	25	23	-

Table XXII. 8-way message passing (sending and receiving) for pattern 1 (inviscid first order case) by RSB partitioning

needs to send 8 messages to processor 7 but receive 6 messages from processor 7. As the viscous first order case considered, the messages of sending and receiving are different. Other tables can be explained in the similar way.

Finally the parallel computing is completed on the cluster with the support of PVM software. Table XXXII gives the results of CPU time per iteration in seconds of four different cases and with the single and multi-processors. The partitioning result of the RSB method is used in this calculation. Table XXXIII show the results with the same considerations but using the partitioning results of the MGP method. Small differences are found due to the same good quality partitioning. Here the speedup is defined as the ratio of the time with which a sequential code and its parallel equivalent finishes per iteration. Figure 65 shows that the CPU time decreases and speedup increases by parallel computing for all four test cases of RSB partitioning results. The speedup value of 6 has been achieved when using 8 processors on the cluster. Figure 66 shows the similar results when the MGP partitioning method is used. Similarly speedup of 6 is achieved.

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	55	-	8	6	-	46	8
1	57	-	-	62	-	-	-	-
2	-	-	-	69	28	-	-	-
3	9	65	72	-	22	-	-	-
4	5	-	28	22	-	55	-	19
5	-	-	-	-	58	-	-	51
6	45	-	-	-	-	-	-	47
7	6	-	-	-	20	52	47	-

Table XXIII. 8-way sending messages for pattern 2 (viscous first order case) by RSB partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	57	-	9	5	-	45	6
1	55	-	-	65	-	-	-	-
2	-	-	-	72	28	-	-	-
3	8	62	69	-	22	-	-	-
4	6	-	28	22	-	58	-	20
5	-	-	-	-	55	-	-	52
6	46	-	-	-	-	-	-	47
7	8	-	-	-	19	51	47	-

Table XXIV. 8-way receiving messages for pattern 2 (viscous first order case) by RSB partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	93	-	17	10	-	78	14
1	94	-	-	116	-	-	-	-
2	-	-	-	114	49	-	-	-
3	16	122	120	-	40	-	-	-
4	8	-	49	37	-	82	-	34
5	-	-	-	-	91	-	-	85
6	74	-	-	-	-	-	-	88
7	9	-	-	-	35	89	90	-

Table XXV. 8-way sending messages for pattern 3 (high order case) by RSB partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	94	-	16	8	-	74	9
1	93	-	-	112	-	-	-	-
2	-	-	-	120	49	-	-	-
3	17	116	114	-	37	-	-	-
4	10	-	49	40	-	91	-	35
5	-	-	-	-	82	-	-	89
6	78	-	-	-	-	-	-	90
7	14	-	-	-	34	85	88	-

Table XXVI. 8-way receiving messages for pattern 3 (high order case) by RSB partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	25	-	22	-	-	-	-
1	25	-	3	4	5	1	13	13
2	-	3	-	24	-	24	-	-
3	22	4	24	-	-	-	-	-
4	-	5	-	-	-	22	-	36
5	-	1	24	-	22	-	-	-
6	-	13	-	-	-	-	-	30
7	-	13	-	-	36	-	30	-

Table XXVII. 8-way message passing (sending and receiving) for pattern 1 (inviscid first order case) by MGP partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	50	-	46	-	-	-	-
1	52	-	10	11	13	3	27	29
2	-	7	-	51	-	53	-	-
3	47	10	51	-	-	-	-	-
4	-	11	-	-	-	47	-	74
5	-	4	52	-	44	-	-	-
6	-	29	-	-	-	-	-	61
7	-	29	-	-	73	-	64	-

Table XXVIII. 8-way sending messages for pattern 2 (viscous first order case) by MGP partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	52	-	47	-	-	-	-
1	50	-	7	10	11	4	29	29
2	-	10	-	51	-	52	-	-
3	46	11	51	-	-	-	-	-
4	-	13	-	-	-	44	-	73
5	-	3	53	-	47	-	-	-
6	-	27	-	-	-	-	-	64
7	-	29	-	-	74	-	61	-

Table XXIX. 8-way receiving messages for pattern 2 (viscous first order case) by MGP partitioning

Processors	Processors							
	0	1	2	3	4	5	6	7
0	-	82	-	76	-	-	-	-
1	91	-	18	24	22	5	50	53
2	-	11	-	86	-	81	-	-
3	82	22	91	-	-	-	-	-
4	-	21	-	-	-	79	-	127
5	-	7	89	-	78	-	-	-
6	-	54	-	-	-	-	-	102
7	-	53	-	-	128	-	108	-

Table XXX. 8-way sending messages for pattern 3 (high order case) by MGP partitioning

Processors	Proccessors							
	0	1	2	3	4	5	6	7
0	-	91	-	82	-	-	-	-
1	82	-	11	22	21	7	54	53
2	-	18	-	91	-	89	-	-
3	76	24	86	-	-	-	-	-
4	-	22	-	-	-	78	-	128
5	-	5	81	-	79	-	-	-
6	-	50	-	-	-	-	-	108
7	-	53	-	-	127	-	102	-

Table XXXI. 8-way receiving messages for pattern 3 (high order case) by MGP partitioning

Cases	Proccssors						
	1	2	3	4	5	6	8
case 1	0.899	0.478	0.340	0.260	0.217	0.184	0.149
case 2	1.269	0.710	0.473	0.378	0.305	0.262	0.222
case 3	1.711	0.978	0.610	0.479	0.408	0.337	0.269
case 4	2.077	1.194	0.800	0.603	0.495	0.422	0.329

Table XXXII. Performance of parallel computing based on RSB partitioning

Cases	Processors						
	1	2	3	4	5	6	8
case 1	0.899	0.482	0.335	0.259	0.210	0.191	0.145
case 2	1.269	0.706	0.487	0.376	0.304	0.257	0.209
case 3	1.711	0.963	0.668	0.478	0.388	0.343	0.294
case 4	2.077	1.214	0.815	0.635	0.499	0.410	0.338

Table XXXIII. Performance of parallel computing based on MGP partitioning

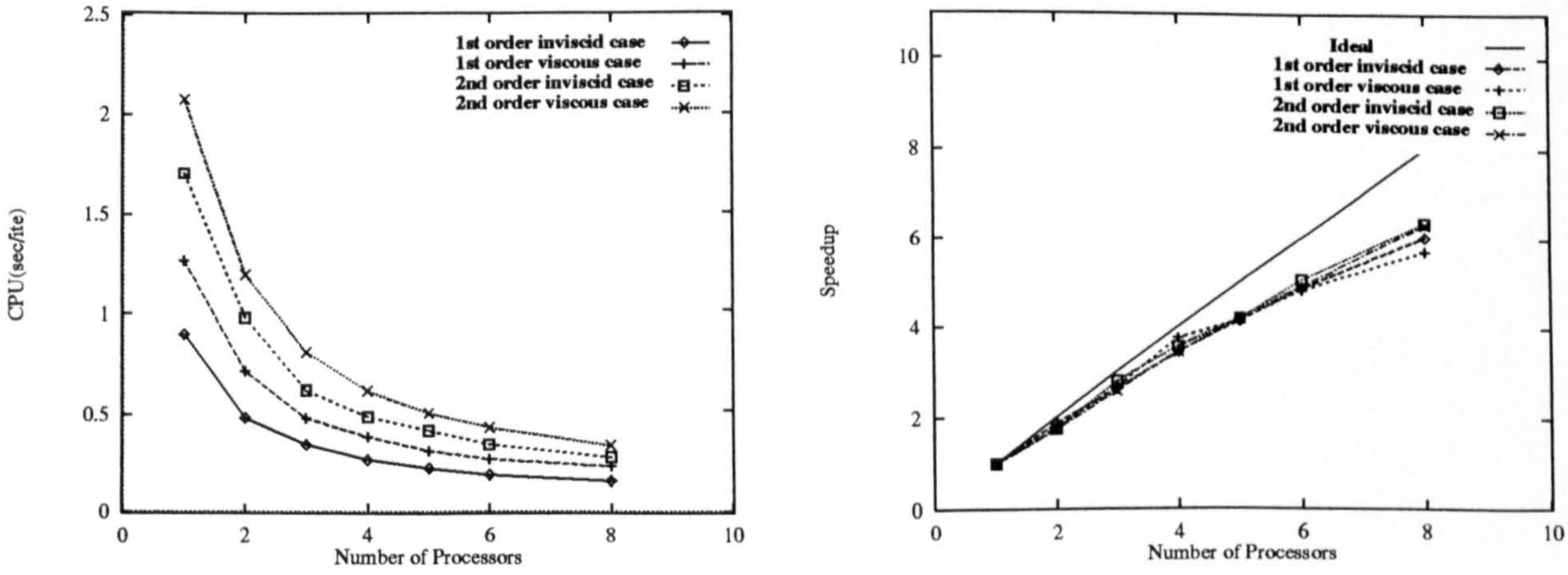
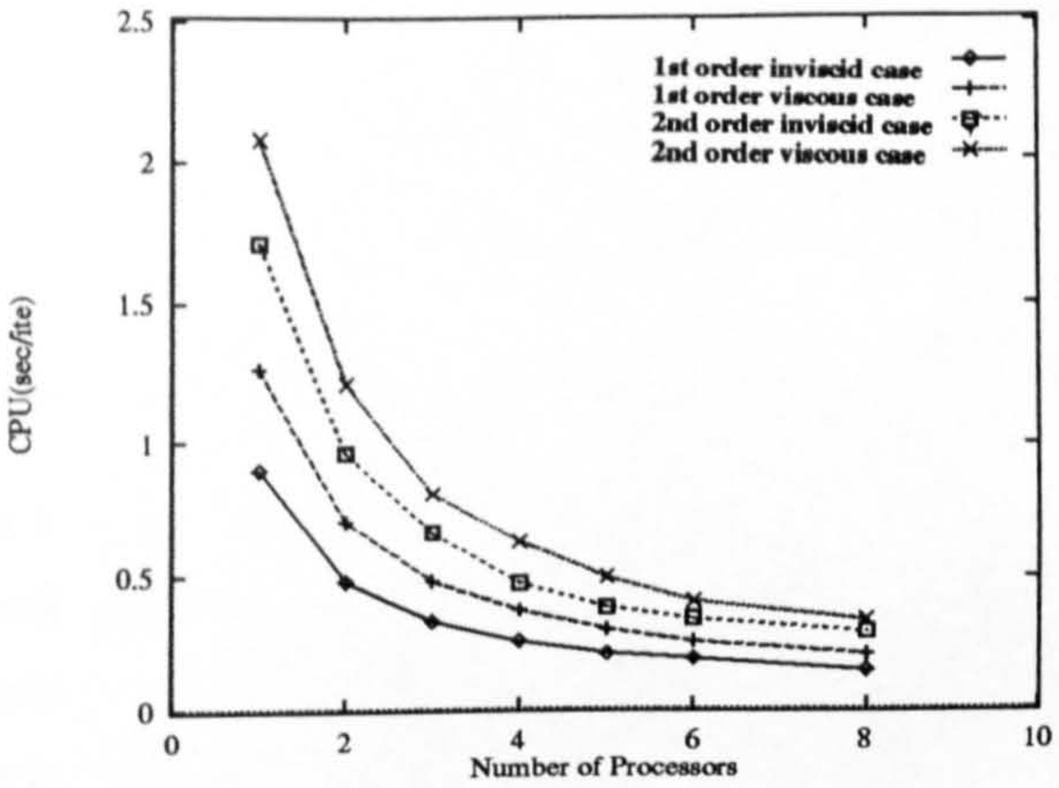


Fig. 65. CPU time and speedup of parallel computing of NACA 0012 airfoil by RSB partitioning



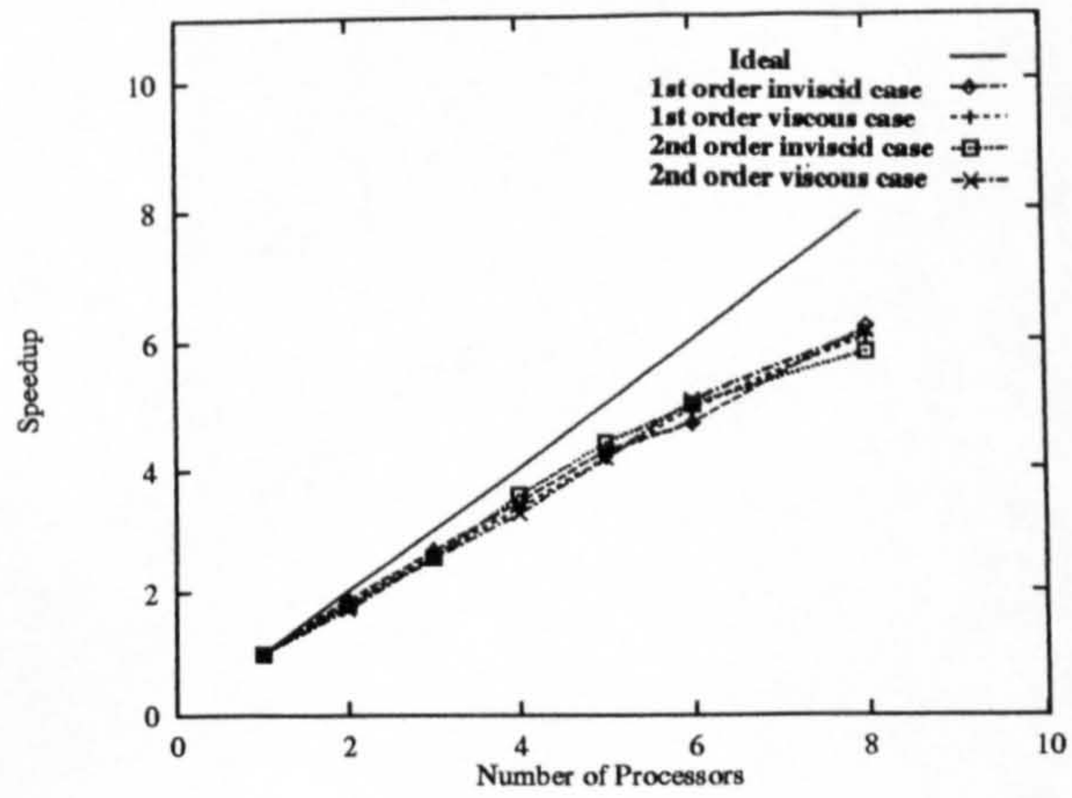


Fig. 66. CPU time and speedup of parallel computing of NACA 0012 airfoil by MGP partitioning

7.6. Conclusions

Implementation of Navier-Stokes code on a parallel cluster is discussed. A summary of partitioning issues on unstructured grids is given. Construction of an efficient data structure is illustrated in detail which has been implemented in a flow solver as a pre-processer. Demonstration has been done with the NS flow solver applied on a transonic flow over NACA 0012 airfoil with inviscid first order, viscous first order, inviscid high order and viscous high order cases considered. Reasonable efficiency and speedups of 6 on 8 processors is achieved.

CHAPTER 8

CONCLUDING REMARKS AND FURTHER RESEARCH

8.1. Concluding remarks

In the present research, two main topics have been discussed. One is the development and evaluation of a discretisation scheme for the solution of the two-dimensional compressible Euler equations on unstructured grids and Navier-Stokes equations on hybrid structured/unstructured grids. Such a scheme is based on the extension of first order upwinding to second order accuracy by Taylor series expansion together with gradient calculations and limiters. The viscous flow problems, both laminar flow with lower Reynolds number and the turbulent flow with higher Reynolds number, have been successfully simulated by the use of the proposed hybrid structured/unstructured grid. The other topic is the implementation of the developed Euler/Navier-Stokes flow solver with powerful state-of-art parallel computing techniques on unstructured grids. Several efficient domain partitioning methods on unstructured grids have been discussed and applied. Reasonable speedup performance has been achieved by the test example executed on a cluster: a parallel computing system based on workstation networks under the parallel virtual machine (PVM) environment. Some conclusions from the research are listed as follows:

(1) A general description of the two-dimensional compressible Navier-Stokes equations relating to computational fluid dynamics was carried out in Chapter 2, which sets up the basis of the overall researches throughout this thesis;

(2) Chapter 3 gives the numerical solution of the two-dimensional inviscid Euler flow equations for various configurations of the aerospace industry interest. The mesh used is the unstructured grid generated by the advancing front technique. A cell-centred finite volume method has been used to discretize the equations. Both explicit and point implicit iteration algorithms are derived. The flux calculation using Roe's and Osher's approximate Riemann solver are studied. It was shown that both the schemes of Roe and Osher lead to an accurate representation of discontinuities (e.g. shock waves). It was also shown that the point implicit scheme performs better in convergence than did the explicit scheme. Validation exercises have been carried out for the following cases: subsonic and transonic flow over airfoils; supersonic flow past a compression corner; and hypersonic flow past a cylinder and a blunt body. Numerical results of supersonic and hypersonic flows are improved by adding to these codes a high order reconstruction method based on the MUSCL approach. An adaptive

remeshing procedure is also applied in test cases in order to get improved solutions;

(3) The general method of high order reconstruction on unstructured grids is discussed in Chapter 4. The Taylor series expansion is adopted to implement this. The calculation of the gradient value at a reference point is carried out using either the Green-Gauss integral or the least-square methods. Five recently proposed limiter construction methods have been used and performance has been compared using the test example of transonic flow over a RAE 2822 airfoil. It has been shown that similar pressure distribution results are obtained by all limiters except within regions in the vicinity of shock waves where the limiter is active. The difficulty in convergence behaviour is demonstrated using a mid-mod type limiter such as the Barth-Jesperson limiter. It seems that only the Venkatakrishnan limiter works well in the improvement of residual convergence. Other limiters do not work as well in the present work as was shown in their original publications. Also the convergence history given by the least-square method seems better than that given by the Green-Gauss method in the test;

(4) The formulation of viscous terms in the Navier-Stokes equations and implementation of a turbulence model on hybrid structured/unstructured grids were presented in detail in Chapter 5. Different from the discretization of inviscid terms, the central-difference scheme is used in viscous term discretization. The unstructured grid is generally not suitable for the viscous problem, because of the feature that it cannot easily create highly stretched grids which is necessary for viscous flow computation. Here a hybrid grid generation method named the "skin" method is proposed. The resulting grid with a structured grid in the near-wall surface and an unstructured grid in other regions has been successfully used in the laminar flow calculation. For the high Reynolds number case, flow is likely to become turbulent. A Baldwin-Lomax algebraic turbulence model is implemented in the NS flow solver. Good performance has been shown by the test case of subsonic flow over a NACA 0012 airfoil and transonic flow over a RAE 2822 airfoil (case 9). Good agreement with experiment has been achieved;

(5) Discussion on domain partitioning on unstructured grids has been given in Chapter 6. The definition of the partitioning problem is illustrated. Then several currently used partitioning methods, i.e. recursive coordinate bisection (RCB), recursive graph bisection (RGB) (together with reverse Cuthill-Mckee (RCM) ordering), recursive spectral bisection (RSB) and multilevel graph partitioning (MGP), have been described and performance of their application on airfoil problems has been illustrated. Some pre-ordering and smoothing algorithms have been proposed and developed in order to improve the partitioning quality by RCB and RGB. The

domain dividing technique (DDT) is also mentioned. Generally, the RSB and MGP partitioning methods give better results, and also the MGP method has been found to be much cheaper to run than the RSB method. All methods have been applied on either single or multi-element airfoils, resulting in load balanced partitioning. The quality of partitioning, judged by the number of edge cuts, is similar in all cases;

(6) The development of the parallel code for the numerical methods described in Chapters 2, 3, 4 and 5 and the domain decomposition method of Chapter 6 has been completed in Chapter 7. Through using the RSB and MGP methods, a load balanced partitioning result with good quality (number of edge cuts minimized) is achieved. A method of construction of the message passing data structure is proposed. The definition of three types of so-called nearest elements is given and also illustrated in the examples. The NS flow solver is parallelized by coupling with the standard subroutines of PVM software and has been successfully executed on a cluster: a parallel computer system developed on a network of workstations using the PVM environment. Performance of parallel computing on airfoil problems has been demonstrated on up to 8 processors. Message passing data structures (sending and receiving) and communication graphs have been illustrated by tables and graphs. Promising speedup results have been achieved.

8.2. Suggestion for further research

The ultimate goal of CFD code is to complete computations with more accuracy and less cpu time. Hence research work could be carried out to further these two aspirations.

8.2.1. The improvement of computational accuracy

Higher Order Discretized Schemes: The temporal accuracy of the discretization of unknowns can be improved by using second order schemes such as developed by Batina [44] and Hwang and Liu [105]. Also the spatial accuracy of the discretization of inviscid terms can be replaced by higher than second order schemes. For example in the finite volume community, Barth [51] [87] derived a higher order scheme that involved the reconstruction of variables satisfying the property of conservation of mean, k-exactness. The k-exactness refers to the property of being able to reconstruct exactly polynomials of degree $\leq$ k. The key idea is to extend the scheme to enable the coefficients in the polynomial to be determined. Chakravarthy et al [106] also proposed a similar approach to achieve higher order accuracy. Harten and Chakravarthy [23] have proposed a framework for applying ENO schemes on

unstructured grids.

Multi-Dimensional Upwinding Scheme: Assuming a one-dimensional Riemann property to resolve the discontinuities across cell interfaces in a multi-dimensional problem is certainly not the most accurate approach. Thus multi-dimensional Riemann solvers based on replacing the propagation of the disturbances by a limited number of waves should be considered. Reference to this techque can be found in [107].

Turbulence Modelling: Although the Baldwin-Lomax algebraic turbulence model has been successfully applied in the present research with the help of the hybrid grid, field turbulence models, such as one- and two- equations are still a topic for further development. The $k - \epsilon$ model has been successfully applied on unstructured grids in [65]. Two fairly new one-equation models, i.e. the Baldwin-Barth model [63] and the Spalart-Allmaras model [64], have become popular, particularly for unstructured grid applications [45].

Adaption: The adaption has been used in this research based on the adaptive remeshing procedure proposed by Peraire et al [50]. Other adaptive techniques, especially adaptive mesh refinement (AMR), should be considered. A survey on AMR technique was given by Powell et al [49].

8.2.2. The improvement of convergence speed

With unstructured grids the convergence to steady state is usually unacceptably slow compared to its structured grid counterpart. Therefore acceleration techniques are required.

Multigrid Method: The multigrid method has been demonstrated as an efficient way to obtain the steady-state solutions of compressible Navier-Stokes equations on unstructured grids. The multigrid approach concerns convergence acceleration by time stepping on successively coarser meshes. The idea behind this methodology is the error associated with the different bands of frequencies being damped on different sizes of mesh. For structured grids a coarse grid can be easily derived from a fine grid. In the case of unstructured grids, special approaches must be considered [24].

Implicit Scheme: Implicit schemes for compressible Navier-Stokes equations have been developed to accelerate the convergence. Throughout this thesis a point implicit

iterative procedure is used. It is also possible to further this by using more sophisticated fully implicit schemes such as the Generalized Minimal RESidual (GMRES). Venkatakrishnan and Mavriplis [46] have tested a family of implicit schemes on unstructured grids by GMRES with ILU preconditioning. Another two implicit method that have been investigated are based on the use of "snakes" [47] and "linelets" [48].

Parallel Computing: Alongside the developments of advanced algorithms of CFD, the recent appearance of parallel computing also supplies an avenue to achieve acceleration of computation within reasonable wall clock time. Some of the issues that need be addressed are partitioning of the grid, message patterns, data structures and design of parallel algorithms. In Chapters 6 and 7 some initial researches in parallel computing on unstructured grids have been given. Futher research is needed in this most active area.

8.2.3. Extend to 3D cases

More realistic applications require a 3D flow solver. Technically, the proposed scheme in this thesis can be extended to the 3D case without much qualitative modification. The first thing be considered is 3D unstructured grid generation. Some well-designed commercial packages with an interface to CAD platforms, suitable to deal with real 3D problems and producing reasonable unstructured grids with less efforts for users are available. The Geomesh P-cube and Tgrid software of the Rampant package (a product of Fluent) were tested by the author to generate satisfactorily 3D unstructurd grids over an M6 wing with 70,000 tetrahedral elements. The generation procedure is completed within several minutes by a SG Indy workstation. Figure 67 illustrates the result. However a hybrid grid generation method needs to be extended to the 3D case in order to calculate viscous flows.

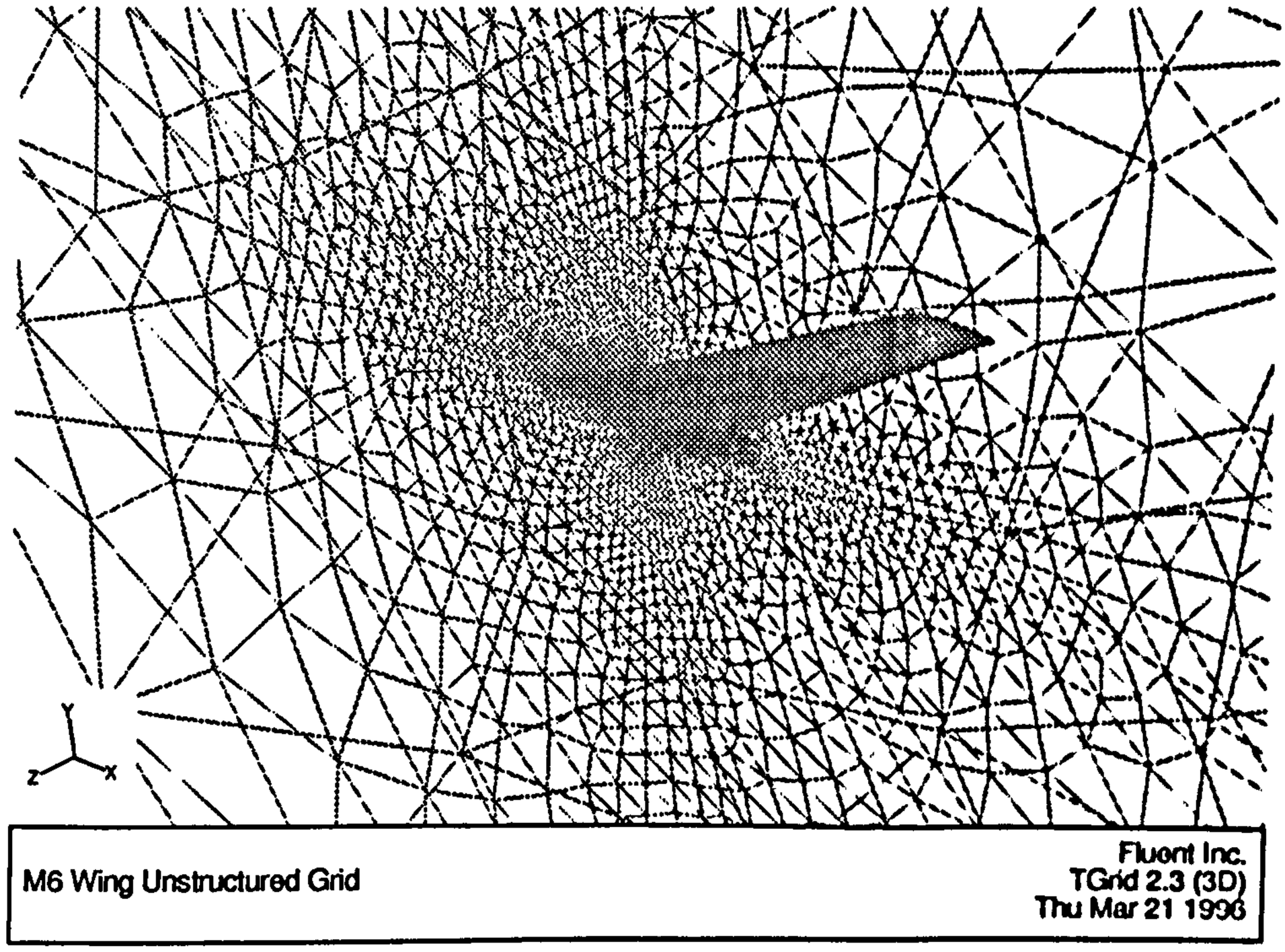


Fig. 67. 3D unstructured grids generated by Geomesh P-cube and Tgrid

REFERENCES

[1] Lax,P.D., 'Hyperbolic Difference Equations: A Review of the Courant-Friedrichs-Lewy Paper in the Light of Recent Developments', *IBM Journal, pp.235-238*, (1967).

[2] Crank,J. and Nicolson,P., 'A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of Heat Conduction Type', *Proceedings of the Cambridge Philosophical Society, Vol.43, pp.50-67*, (1947).

[3] Du Fort,E.C. and Frankel,S.P., 'Stability Conditions in the Numerical Treatment of Parabolic Differential Equations', *Mathematics Tables and Other Aids to Computation, Vol.7, pp.135-152*, (1953).

[4] Peaceman,D.W. and Rachford,H.H., 'The Numerical Solution of Parabolic and Elliptic Differential Equations', *Journal of SIAM, Vol.3, pp.28-41*, (1955).

[5] Lax,P.D. and Wendroff,B., 'Systems of Conservation Laws', *Communications on Pure and Applied Mathematics, Vol.13, pp.217-237*, (1960).

[6] Richtmyer,R.D. and Morton,K.W., 'Difference Method for Initial Value Problems', *2nd Eds. Interscience Publishers, John Wiley and Sons*, (1967).

[7] MacCormack,R.W., 'The Effect of Viscosity in Hypervelocity Impact Cratering', *AIAA paper 69-354*, (1969).

[8] Beaming,R.M. and Warming,R.F., 'An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form', *Journal of Computational Physics, Vol.22, pp.87-110*, (1976).

[9] von Neumann,J. and Richtmyer,R.D., 'A Method for the Numerical Calculation of Hydrodynamics Shocks', *Journal of Applied Physics, Vol.21, pp.232-257*, (1950).

[10] Courant,R., Isaacson,E., and Rees,M., 'On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences', *Communications on Pure and Applied Mathematics, Vol.5, pp.243-255*, (1952).

[11] Godunov,S.K., 'Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics', *Matematicheskii Sbornik, Vol.47, pp.272-306*, (1959).

[12] Steger,J.L., and Warming,R.F., 'Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods' *Journal of Computational Physics Vol.40, pp.263-293*, (1981).

[13] van Leer,B., 'Flux-Vector Splitting for the Euler Equations' *Lecture Notes in Physics, Vol.170, Springer-Verlag, Berlin, pp.507-512*, (1982).

[14] Roe,P.L., 'Approximate Riemann Solvers, Parameter Vectors and Difference Schemes' *Journal of Computational Physics ,Vol.43, pp.357-372*, (1981).

[15] Roe,P.L., 'Characteristic-Based Schemes for the Euler Equations' *Annual Review of Fluid Mechanics, Vol.18, pp.337-365*, (1986).

[16] Osher,S., 'Numerical Solution of Singular Perturbation Problems and Hyperbolic Systems of Conservation Laws' *North-Holland Mathematical Studies, Vol.97, North-Holland,Amsterdam, p.179*, (1981).

[17] Osher,S., 'Riemann Solvers, the Entropy Condition and Difference Approximations' *SIAM Journal on Numerical Analysis, Vol.21, pp.217-235*, (1984).

[18] Boris,J.P., and Book,D.L., 'Flux Corrected Transport, 1 SHASTA, a Flux Transport Algorithm that works', *Journal of Computational Physics Vol.11, PP.38-69*, (1973).

[19] Harten,A., 'On a Class of High-Resolution Total-Variation-Stable Finite Difference Schemes' *SIAM Journal on Numerical Analysis, Vol.21, pp.1-23*, (1984).

[20] Shu,C., 'TVB Uniformly High-Order Schemes for Conservation Laws' *Mathematics of Computation, Vol.49, pp.105-121*, (1987).

[21] van Leer,B., 'Towards the Ultimate Conservative Difference Scheme V. A second order sequal to Godunov's method' *Journal Computational Physics, 32(1979), pp.101-136.*

[22] Harten,A., Engquist,B., Osher,S., and Chakravarthy,S.R., 'Uniformly High Order Accurate Essentially Non-Oscillatory Schemes,III' *Journal of Computational Physics, Vol.71, pp.231-303*, (1987).

[23] Harten,A. and Chakravarthy,S.R., 'Multi-Dimensional ENO Schemes for General Geometries', *ICASE, NASA Langley Research Center, ICASE report 91-76*, (Sept.,1991).

[24] Venkatakrishnan,V., 'Perspective of Unstructured Grid Flow Solvers', *AIAA Journal Vol.34, No.3*, (1996).

[25] Jameson,A. and Mavriplis,D.J., 'Finite Volume Solution of the Two-Dimensional Euler Equation on Regular Triangular Meshes', *AIAA Journal, Vol.24, pp.611-618*, (1986).

[26] Jameson,A., Baker,T.J., and Weatherill,N.P., 'Calculation of Inviscid Transonic Flow over a Complete Aircraft', *AIAA paper 86-0103*, (1986).

[27] Desideri,J.A., and Dervieux,A., 'Compressible Flow Solvers Using Unstructured Grids', *VKI Lecture Series, 1988-05, von Karman Institute for Fluid Dynamics, Belgium, pp.1-115*, (1988).

[28] Lohner,R., Morgan,K., Peraire,J., and Vahdati,M., 'Finite Element Flux-Corrected Transport(FCT) for the Euler and Navier-Stokes Equations' *Computational Methods in Applied Mechnics and Engineering, Vol.7, pp.1093-1109*, (1987).

[29] Batina,J.T., 'Implicit Flux-Split Euler Schemes for Unsteady Aerodynamic Analysis Involving Unstructured Dynamic Meshes' *AIAA Journal, Vol.29, No.11, pp.1836-1843*, (1991).

[30] Knight D.D., 'A Fully Implicit Navier-Stokes Algorithm Using an Unstructured Grid and Flux-Difference Splitting', *AIAA paper 93-0875*, (Jan.1993).

[31] Frink, N.T., 'Upwind Scheme for Solving the Euler Equations on Unstructured Tetrahedral Meshes' *AIAA Journal, Vol.30, No.1, pp.70-77*, (1992).

[32] Venkatakrishnan,V., and Barth,T.J., 'Application of Direct Solvers to Unstructured Meshes for the Euler and Navier-Stokes Equations Using Upwind Schemes' *AIAA paper 89-0364*, (1989).

[33] Barth,T.J., and Jesperson,D.C., 'The Design and Application of Upwinding Schemes on Unstructured Meshes' *AIAA paper 89-0366*, (1989).

[34] Venkatakrishnan,V., 'Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters' *Journal Computational Physics 118,(1995) pp.120-130.*

[35] Bishop,D.G., and Noack,R.W., 'An Implicit flow Solver with Upwinding Differencing for Three Dimensional Hybrid Grids' *AIAA-95-1707-Cp*, (1995).

[36] van Rosendale,J., 'Floating Shock Fitting via Lagrangian Adaptive Meshes' *AIAA -95-1721-Cp*, (1995).

[37] Aftosmis,M., Gaitonde,D., and Tavares,T.S., 'Behavior of Linear Reconstruction Techniques on Unstructured Meshes' *AIAA Journal, Vol.33, No.11, pp.2038-2049*, (1995).

[38] Frink,N.T., Parikh,P., and Pirzadeh,S., 'Aerodynamic Analysis of Complex Configurations Using Unstructured Grids' *AIAA paper 91-3292*, (1991).

[39] Barth,T.J., 'A Three-Dimensional Upwind Euler Solver for Unstructured Meshes' *AIAA Paper 91-1548CP*, (July,1991).

[40] Mavriplis,D.J., 'Three-Dimensional Multigrid Reynolds-Averaged Navier-Stokes Solver for Unstructured Meshes ', *AIAA Journal, Vol.33, No.3, pp.445-453*, (1995).

[41] Potsdam,M., Intemann,G., Frink,N.T., Campbell,R., Smith,L., and Pirzadeh,S., 'Wing Pylon Fillet Design Using Unstructured Mesh Euler Solvers' *AIAA Paper 93-3500*, (Aug.,1993).

[42] Thareja,R.R., Stewart,J.R., Hassan,O., Morgan,K., and Peraire,J., 'A Point-Implicit Unstructured Grid Solver for the Euler and Navier-Stokes Equations' *International Journal for Numerical Methods in Fluids, Vol.9, pp.405-425*, (1989).

[43] Hassan,O., Morgan,K., and Peraire,J., 'An Adaptive Implicit/Explicit Finite Element Scheme for Compressible High Speed Flows' *AIAA paper 89-0363*, (1989).

[44] Batina,J.T., 'Implicit Upwind Solution Algorithms for Three-Dimensional Unstructured Meshes' *AIAA Journal, Vol.31, No.5, pp.801-805*, (1993).

[45] Anderson,W.K.,and Bonhaus,D.L., 'An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids' *Computers and Fluids, Vol.23, No.1, pp.1-21*, (1994).

[46] Venkatakrishnan,V., Mavriplis,D.J., 'Implicit Solvers for Unstructured Meshes', *Journal of Computational Physics, Vol.105, pp.83-91*, (1993).

[47] Hassan,O., Morgan,K., and Peraire,J., 'An Implicit Finite Element Method for High Speed Flows', *AIAA paper 90-0402*, (1990).

[48] Lohner,R., 'Simple Elements and Linelets for Incompressible Flows', *in CIMNE Finite Elements in the 90's, edited by E.Onate, J.Periaux, A. Samuelsson. Barcelona: Spring-Verlag*, (1991).

[49] Powell,K.G., Roe, P.L. and Quirk,J.J., 'Adaptive-Mesh Algorithms for Computational Fluid Dynamics' *Algorithmic Trends in Computational Fluid Dynamics, edited by M.Y.Hussaini, A.Kumar, and M.D.Salas, Spring-Verlag, New York, pp.301-337*, (1992).

[50] Peraire,J., Vahdati,M., Morgan,K., and Zienkiewicz,O.,C., 'Adaptive Remeshing for Compressible Flow Computations' *Journal of Computational Physics, Vol.72, pp.449-466*, (1987).

[51] Barth,T.J., 'Recent Developments in High Order K-Exact Reconstruction on Unstructured Meshes' *AIAA Paper 93-0668*, (1993).

[52] Lo,S.H., 'Volume Discretization into Tetrahedra,Part I: Verification and Orientation of Boundary Surfaces' *Computers and Structures, Vol.39, No.5, pp.493-500*, (1991).

[53] Thompson, J.F., and Weatherill, N.P., 'Aspects of Numerical Grid Generation: Current Science and Art', *AIAA paper 93-3539*, (1993).

[54] Mavriplis,D.J., 'Algebraic Turbulence Modelling for Unstructured and Adaptive Meshes' *AIAA paper 90-1653*, (1990).

[55] Mavriplis,D.J., 'Euler and Navier-Stokes Computations for Airfoil Geometries Using Unstructured Meshes' *Canadian Aeronautics and Space Journal, Vol.36, No.2, pp.62-71*, (1990).

[56] Vilsmerier,R., and Hanel,D.,, 'Adaptive Solutions for Compressible Flows on Unstructured Strongly Anisotropic Grids' *Computational Fluids Dynamics, Proceedings of the 1st European Conference. Eds. J.Periaux, C.Hirsch and W.Kordulla. Brussels:Elsevier*, (1992).

[57] Hassan,O., Probert, Morgan,K., and Peraire,J., 'Line Relaxation Methods for the 2D and 3D Compressible Flows', *AIAA paper 93-3366*, (1993).

[58] Pirzadeh,S., 'Unstructured Viscous Grid Generation by Radvanary Front Method' *11th AIAA Applied Aerodynamics Meeting, Monterey, CA*, (1993).

[59] Weatherill,N.P., Hassan,O., Marcum,D.L., and Marchant,M.J., 'Grid Generation by the Delauney Triangulation' *Lecture Series 1994-02, VKI Jan.24-28*, (1994).

[60] Holmes,D.,G., and Connell,S.,D., 'Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids' *AIAA paper 89-1932-CP*, (1989).

[61] Baldwin,B.S., and Lomax,H., 'Thin Layer Approximation and Algebriac Model for Separated Turbulent Flows' *AIAA paper 78-257*, (1978).

[62] Kallinderis,Y., 'Algebraic Turbulence Modelling for Adaptive Unstructured Meshes' *AIAA Journal, Vol.30, No.3, pp.631-639*, (1992).

[63] Baldwin,B.S., and Barth,T.J., 'A One-Equation Turbulence Transport Model for High Reynolds Wall-Bounded Flows' *AIAA paper 91-0610*, (1991).

[64] Spalart,S.R., and Allmaras,S.A., 'A One-Equation Turbulence Model for Aerodynamic Flows' *AIAA paper 92-0439*, (1992).

[65] Mavriplis,D.J., and Martinelli,L., 'Multigrid Solution of Compressible Turbulent Flow on Unstructured Meshes Using a Two-Equation Model' *International Journal for Numerical Methods in Fluids, Vol.18, pp.887-914*, (1994).

[66] Pothen,A., Simon,H.D., and Liou,K.P., 'Partitioning Sparse Matrices with Eigenvectors of Graphs', *SIAM Journal of Matrix Analysis and Application, Vol.11, No.3, pp.430-452*, (1990).

[67] Simon,H.D., 'Partitioning of Unstructured Problems for Parallel Processing', *Computing Systems in Engineering, Vol.2, pp.135-148*, (1991).

[68] Barnard,S.T., and Simon,H.D., 'A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems', *In Proceedings of the sixth SIAM conference on Parallel Processing for Scientific Computing, pp.711-718*, (1993).

[69] Miller,G.L., Teng,S.H., and Vavasis,S.A., 'A Unified Geometric Approach to-Graph Seperators', *In Proceedings of the 31th Annual Symposium on Foundations of Computer Science, pp.538-547*, (1991).

[70] Miller,G.L., Teng,S.H., Thurston,W., and Vavasis,S.A., 'Automatic Mesh Partitioning', *In A.George, John R. Gilbert, and J.W.-H. Liu editors "Sparse Matrix Computations:Graph Theory Issues and Algorithms", Spring-Verlag, New York*, (1993).

[71] Hendrickson,B., and Leland,R., 'A Multilevel Algorithm for Partitioning Graphs', *Technical Report, SAND 93-1301, Sandia National Laboratories, USA*, (1993).

[72] Karypis,G., and Kumar,V., 'Multilevel Graph Partition and Sparse Matrix Ordering', *In International Conference on Parallel Processing*, (1995).

[73] Farhat,C., Fezoui,L., and Lanteri,S., 'Two-Dimensional Viscous Flow Computation on the Connection Machine: Unstructured Meshes, Upwind Schemes and Parallel Computation', *Computational Method in Applied Mechanics and Emgineering, Vol.102, pp.61-88*, (1993).

[74] Johan,Z., Hughes,T.J.R., Mathur,K.K., and Johnson,S.L., 'A Data Parallel Finite Element Method for Computational Fluid Dynamics on the Connection Machine System', *Computational Method in Applied Mechnics Engineering, Vol.99, pp.113-124*, (1992).

[75] Mavriplis,D.J., Das,R., Saltz,J., and Vermeland,R.E., 'Implementation of a Parallel Unstructured Euler Solver on Shared and Distributed Memory Machines', *Journal of Supercomputing, Vol.8, No.4, pp.329-344*, (1994).

[76] Venkatakrishnan,V., Simon, H.D., and Barth, T.J., 'A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids', *Journal of Supercomputing, Vol.6, pp.117-137*, (1992).

[77] Ramamurthi,R., Sandberg,W., and Lohner,R., 'Evaluation of a Scable 3D Finite Element Incompressible Flow Solver', *AIAA paper 94-0756*, (1994).

[78] Venkatakrishnan,V., 'Parallel Implicit Unstructured Grid Euler Solvers', *AIAA Journal, Vol.32, pp.1985-1991*, (1994).

[79] Mavriplis,D.J., 'Mesh Generation and Adaptivity for Complex Geometries', *Handbook of Computational Fluid Mechanics, Academic, San Diego, CA*, (1995).

[80] Anderson,W.K., 'A Grid Generation and Flow Solution Method for the Euler Equation of Unstructured Grids' *Journal of Computational Physics, Vol.110, pp.23-28*, (1994).

[81] Harten,A., 'High Resolution Schemes for Hypersonic Conservation Law' *DOE/ER/03077-175, Courant Mathematics and Computing Lab. NYU*, 1982

[82] Dubuc,L., 'Two-Dimensional Navier-Stokes Solver Using an Upwind Scheme on Unstructured Grids' *Final Report, Sept.1992 Dept. of Acorspace Engineering, University of Glasgow*, 1992.

[83] Hirsch,C., 'Numerical Computation of Internal and External Flows' *Vol.2, John Weiley & Sons*, (1988).

[84] Van Leer,B., Van Albada,G.D., and Roberts,W.W., 'A Comparative Study of Computational Methods in Cosmic Gas Dynamics' *Astronomy and Astrophysics, 108*, (1982).

[85] Morgan,K., Peraire,J., Peiro,J., 'Unstructured Grid Methods for Compressible Flows', *AGARD R-787(5)*, (1992).

[86] Hirsch,C., and Van Ransbeeck,P., 'Multidimensional Upwinding and Artificial Dissipation for the Euler/Navier-Stokes Equations', *AIAA-95-1702-Cp*, (1995).

[87] Barth,T.J., 'Aspects of Unstructured Grids and Finite Volume Solvers for the Euler and Navier-Stokes Equations' *AGARD R-787(6)*, (1992).

[88] Golub,G., and Load,C.V., 'Matrix Computations' *Johns Hopkins Univ. Press. Baltimore.MD*, (1991).

[89] Yao,Y.F., 'Hybrid Mesh Generation Techniques and Euler Flow Solver Validation' *Aero. Report 9501, Department of Aerospace Engineering, University of Glasgow, Jan.*, (1995).

[90] Thompson,J.F., Lijewski,L.E., Cipolla,J., and Gatlin,B., 'Program EAGLE User's Manual Vol.I - Introduction and Grid Applications' *AFATL-TR-88-117*, (1988).

[91] Soltani,S., 'An Upwind Scheme for the Equations of Compressible Flow on Unstructured Grids' *Thesis submitted for the degree of Doctor of philosophy of the University of London, July*, (1991)

[92] Bristeau,M.O., Glowinski,R., Periaux,J. and Viviand,H., 'Numerical Simulation of Compressible Navier-Stokes Flows' *Notes on Numerical Fluid Mechanics Volume 18*, (1985).

[93] Marcum,D.L., and Weatherill,N.P., 'Finite Element Calculation of Inviscid and Viscous Flow Fields about Launch Vehicle Configurations' *VIII International Conference on Finite Elements in Fluids, Barcelona,Spain, September 20-24*, (1993).

[94] Mavriplis,D.J., 'Algebraic Turbulence Modelling for Unstructured and Adaptive Meshes' *AIAA Journal, Vol.29, No.12, pp.2086-2093*, (1991).

[95] Pan,D, and Cheng, J.C., 'Upwind Finite-Volume Navier-Stokes Computations on Unstructured Triangular Meshes' *AIAA Journal, Vol.31, No.9, pp.1618-1625*, (1993).

[96] Cebeci,T., 'Calculation of Compressible Turbulence Boundary Layers with Heat and Mass Transfer' *AIAA paper 70-741*, (1970).

[97] Thibert,J.J., Grandjacques,M., and Ohmman,L.H., 'NACA 0012 Airfoil: Experimental Data Base for Computer Program Assessment' *AGARD Advisory Rept.138*, (May,1979).

[98] Cook,P.H., McDonald,M.A., and Firmin,M.C.P., 'Aerofoil RAE 2822 Pressure Distributions and Boundary Layer and Wake Measurements' *AGARD Advisory Rept.138*, (May,1979).

[99] Karypis,G., and Kumar,V., 'A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs', *Technical Report:95-035, Department of Computer Science, University of Minnesota, Minneapolis,USA*, (1995).

[100] George,A., and Liu,J.W.-H., 'Computer Solution of Large Sparse Positive Definite Systems' *Prentice Hall, Englewood Cliffs*, (1981).

[101] Parlett,N., 'The symmetric Eigenvalue Problem' *Prentice Hall, Englewood Cliffs*, (1980).

[102] Mavriplis,D.J., Das,R., Saltz,J., and Vermeland,R.E., 'Implementation of a Parallel Unstructured Euler Solver on Shared and Distributed Memory Machines', *ICASE Report No.92-68*, (1992).

[103] Hammond,S., and Barth, T.J., 'On a Massively Parallel Euler Solver for Unstructured Grids', *in Horst D.Simon editor, Parallel CFD:Implementations and Results Using Parallel Computers, pp.55-70 MIT Press, Cambridge, Mass.*, (1992).

[104] Geist,A., Beguelin, A., Dongarra, J., Jiang,W., Manchek, R., and Sunderam, V., 'PVM 3 User's Guide and Reference Manual', *Oak Ridge National Laboratory ORNL/TM-12187, Oak Ridge, Tennessee, USA*, (May 1994).

[105] Hwang,C.J., and Liu,J.L., 'Locally Implicit Total-Variation-Diminishing Schemes on Unstructured Triangular Meshes', *AIAA Journal, Vol.29, No.10, pp.1619-1626*, (1991).

[106] Chakravarthy,S., Szema,K.Y., Ramakrishnan,S.,Burman,R.,and Schultz,R., 'Inviscid CFD for Store Separation Using Unified Boundary Conditions', *AIAA paper 93-3404*, (Aug.,1993).

[107] Deconinck,H., Struijs,R., Paillere,H., Bourgois,G., and Roe, P.L., 'Multidimensional Upwind Methods for Unstructured Grids' *AGARD R-787(4)*, (1992).

[108] Drikakis,D., 'Parallel multiblock characteristic-based method for three-dimensional incompressible flows' *Advances in Engineering Software, July 1996, Vol.26, No.2, pp.111-119*, (1996).