https://theses.gla.ac.uk/

Theses Digitisation:

https://www.gla.ac.uk/myglasgow/research/enlighten/theses/digitisation/

This is a digitised version of the original print thesis.

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses
https://theses.gla.ac.uk/
research-enlighten@glasgow.ac.uk

Generalized Differential-Integral Quadrature and Application to the Simulation of Incompressible Viscous Flows Including Parallel Computation

by

SHU Chang, B.Sc., M.Sc.

A Thesis Submitted to the Faculty of Engineering, the University of Glasgow for the degree of Doctor of Philosophy

April 1991

© C. Shu, 1991

ProQuest Number: 10987051

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 10987051

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

To

Wang Hang

CONTENTS

ACKNOWLEDGEMENTS

The author would like to express his deepest appreciation to his supervisor, Professor Bryan E. Richards for his support and advice throughout this study and especially for his help in the writing of this thesis.

The author also wishes to acknowledge Professor Henry Y. Wong for his help in many aspects, Dr. Qin Ning, Dr. Jiang Dachun, Dr. Wang Zhijian, Mr. Xu Xiao, for their valuable discussions, during this research. Thanks are also due to Miss Margaret Simpson, the secretary of the department, for her kind help, to the staff of Glasgow University Computing Centre, especially to Dr. Derek Higgins, Dr. Paul Rosenberg, for their fruitful service.

The author is indebted to Professor Cao Qipeng of Nanjing Aeronautical Institute, for his many suggestions and encouragement during this period.

Finally, the author would like to thank his parents and especially his wife, Hang, for their support and encouragement throughout the course of this study.

The author gratefully acknowledges the support of the University Scholarship of Glasgow and an ORS award from U.K. Government.

SUMMARY

This research covers three topics: the development of numerical techniques for the solution of partial differential and integral equations; simulations of incompressible viscous flows using these techniques; and their extension to parallel computation of the incompressible N-S equations.

The differential quadrature (DQ) technique, presented by Bellman et al (1972, 1986), was extended to a general case in this research, based on the analysis of a high order polynomial approximation in the overall domain and the analysis of a linear vector space. Generalized differential quadrature (GDQ) has overcome the difficulty of DQ in solving a set of algebraic equations to obtain the weighting coefficients for the discretized derivatives. In the GDQ method, the weighting coefficients for the first derivative can be easily calculated by an algebraic formulation without any restriction on choice of grid points. The weighting coefficients for the second or higher order derivative can then be determined using a recurrence relationship.

The relationship between GDQ and other numerical techniques has been investigated. It was found that the GDQ method can be considered as the highest order finite difference scheme for a domain with a given mesh. Application of both GDQ and Chebyshev pseudospectral methods provides the same weighting coefficients for the first derivative when the grid points are chosen as the roots of the *N*th order Chebyshev polynomial for both methods.

Some basic features of the GDQ method such as error, stability analyses, and the influence of the distribution of grid points and the types of boundary conditions on the eigenvalues of the GDQ spatial discretization matrix have been studied. It was found that improvements in stability were achieved when the grid was stretched near the boundary. This was not the case when the grid was stretched near the mid-point even for very small minimum step size. This behaviour differs from conventional low order finite difference schemes in which the local optimum time step size can be evaluated through the CFL condition. In the GDQ method the time step size is evaluated by the eigenvalues. A uniform grid gives a smaller value of the maximum eigenvalue compared with the stretched grid, but it may cause stability problems. Hence a uniform grid allows the use of a larger time step size, and hence needs less computation time if the solution is stable. For convection-diffusion problems such as the vorticity transport equation in incompressible flows, a uniform grid may cause an instability problem when the diffusive term becomes relatively small compared with the convective term. For this case, the grid stretched near the boundary is recommended to use for improving the stability. The Dirichlet boundary conditions make the solution more stable than Neumann boundary conditions, but give a larger value of the maximum eigenvalue, resulting in more computation time.

A generalized integral quadrature (GIQ) method was also developed, based on the same concept as GDQ. In the GIQ method, not only the integral of a function over the whole domain but also the integral of a function over a part of the whole domain can be approximated by the combination of all the functional values in the whole domain. The weighting coefficients for GIQ can be obtained by inverting the matrix which is derived from the GDQ formulation. The estimation of the errors incurred by GIQ was also studied.

For the application of GDQ to CFD problems, the **2D**, laminar, incompressible N-S equations have been chosen to validate the approach. Solutions include: driven cavity flow; flow past a circular cylinder; flow past a backward facing step; and natural convection in a square cavity. For direct comparison with other numerical techniques, driven cavity flow was also simulated using the second order time split MacCormack finite difference scheme for the vorticity equation and the SIP (strongly implicit procedure) preconditioning technique for the stream function equation. Numerical experiments showed that GDQ results using appreciably fewer grid points are more accurate, and need much less computation time than conventional low order finite difference results using a large number of grid points. Furthermore, to relieve the time step size as the number of grid points increases, and towards the treatment of complex problems, the multi-domain GDQ technique was developed and presented. This was then applied to solve specific complicated flow cases. Examples include the flow past a backward facing step with 3 subdomains, and the flow past a square step with 5 subdomains. A GDQ-GIQ approach for the solution of the boundary layer equations has also been developed in this research. The **1D**, **2D**, **3D** steady and **2D** unsteady boundary layers have been simulated using this approach, each illustrated by means of a specific test problem. Numerical results are demonstrated to be very accurate compared with exact or other numerical results, with the use of only a few grid points.

To test parallel computation on the transputer-based Meiko Computing Surface, the above mentioned multi-domain GDQ solutions of the flows past a backward facing step and a square step were simulated using 3 and 5 slave processors. The program on each processor was written in FORTRAN and run from an Occam harness to control the placement of, and the communication between, processors. For the driven cavity flow problem, two formulations of the N-S equations (vorticity-stream function and vorticity-velocity) and three methods for dealing with the interfaces between subdomains (i.e. patched by enforcing continuity to the function and its normal derivative; patched using a Lagrange interpolation scheme; and overlapped) were studied comparatively. Additionally, an idea for the development of a general code which can be run on any array of transputers without modification to the program was discussed, and was successfully applied to the driven cavity flow problem.

CHAPTER ONE

INTRODUCTION

1.1 The Current Numerical Techniques in CFD

It is well known that Newtonian fluid flow can be governed by the Navier-Stokes (N-S) equations. If the appropriate initial and boundary conditions are given, then the problem is well-posed. The ideal way to find solutions of the N-S equations is through an analytical method. Unfortunately, it is impossible to find analytical solutions of the N-S equations for a general case since the N-S equations are highly nonlinear. Conclusions from linear analysis, such as the principle of linear superposition, are not valid. In fact, only for flows past idealized geometries, such as Poiseuille flow, can analytical solutions be found with restrictive simplifying assumptions. To investigate fluid dynamic systems for practical cases, other ways need to be found. One way is the use of computational fluid dynamics (CFD) to solve the N-S equations or their simplified forms such as the Euler equations, the potential equations, the boundary layer equations and the triple-deck equations on modern digital computers. Compared with the analytical approach, CFD requires relatively few restrictive assumptions and gives a complete description of the flow field for all variables. Quite complex configurations can be treated, and the methods are relatively easy to apply. Relative to the experiments, CFD has few Mach number and scale limitations and is cost effective. During the last few decades, as computers have developed, the cost of numerical simulations of the flows had been greatly reduced by orders of magnitude while at the same time the cost of experimental measurements still remain at the same level. Numerical simulation has an additional advantage over experiment in that diagnostic 'probing' of the computer simulation does not disturb the flow and obscure the phenomena under investigation.

1.1.1 Time Discretization

The time-dependent N-S equations or their simplified forms are commonly used for unsteady or even steady state resolutions. Then a time marching scheme is employed. For the discretization of the time derivative, there are two methods, namely explicit and implicit schemes.

The explicit approach discretizes the time derivative by forward difference schemes, in which the solutions at one time step can be determined explicitly from those at previous time steps. The major advantage of this approach is that it is easy to apply and needs less operational counts per time step. Among the methods used in this category, the MacCormack two step explicit scheme (1969) and the Jameson explicit multi-stage Runge-Kutta scheme (Jameson et al 1981, Swanson et al 1985) were widely employed for solutions of the N-S equations or the Euler equations. The shortcoming of the explicit approach is that the stability condition is often so restrictive because of the CFL condition that the time interval becomes very small and a large number of time steps need be taken before a steady state is reached.

To relax the time interval barrier in the time marching process, one can turn to the implicit approach which discretizes the time derivative by backward or central difference schemes. As compared with explicit schemes, implicit schemes greatly improve convergence to steady state by taking much larger time intervals. The penalty, however is, that more operational counts per time step and more coding work are required. The Beam-Warming (1978) and the MacCormack (1982) implicit schemes are two versions, which are extensively used for solutions of the N-S equations.

It is often difficult to judge whether either the implicit or the explicit approaches are more efficient for a particular application. Some researchers prefer explicit schemes while others favour implicit schemes.

1.1.2 Spatial Discretization

There are several ways to discretize the spatial derivatives in CFD which will now be reviewed.

1.1.2.1 Finite Difference Methods

Finite difference methods are employed in all areas of CFD to solve both inviscid and viscous flow equations. For solutions of the Euler or N-S equations, central and upwind finite difference schemes are usually used. The MacCormack explicit (1969) and implicit (1982) schemes, the Beam-Warming implicit scheme (1978) and the Jameson explicit multi-stage Runge-Kutta scheme (1981) are typical central difference schemes. These schemes, in general, work well for problems with a smooth behaviour. But if the flow field has a discontinuity such as a shock wave, these schemes always produce high frequency oscillation in the vicinity of that discontinuity. In order to remove the oscillation, one can add some form of artificial dissipation (Rizzi and Eriksson, 1983) in the discretized equations solved. The choice of the parameter in the artificial dissipation term, which has a great influence on the results and the convergence to the steady state, depends on personal experience, and sometimes, is not straightforward. More recently, a variety of upwind differencing schemes which produce dissipation automatically have been presented, based on assumptions related to the method of characteristics and wave propagation. These schemes difference the convective term (derivative or the flux) along the direction of wave propagation. The monotone, total variation diminishing (TVD), flux splitting and flux difference schemes fall in this category. The work of Steger and Warming (1981), Van Leer (1982), Chakravarthy and Osher (1985), Roe (1981), Yee (1987) and Harten (1983) has shown that this approach is very efficient in capturing the shock wave. As compared to the central difference scheme, Pulliam (1986) pointed out that the upwind scheme is equivalent to a central difference scheme plus a certain form of dissipation. To capture the shock wave with high resolutions, some workers (Morton et al 1989) preferred to use the shock fitting technique in which the Rankine-Hugoniot relations are applied.

For simulating the flow around a complex geometry, a coordinate transformation is usually required. This can be done by the technique of grid generation. For details, see, the work of Thompson (1978), Thompson et al (1982), Steger et al (1980) and Eiseman (1982).

1.1.2.2 Finite Element Methods

Finite element methods are mostly used in solid mechanics and structural analysis. They have however made a practical contribution to CFD since the 1970s. For details, see, for example, the work of Chung (1978), Hughes et al (1979), Morton (1982), Gresho et al (1984), Morgan et al (1987), Hassan et al (1989). The mathematical foundation of the finite element approach is based on either the variational (or extremal) principle or the principle of weighted residuals. The finite element methods used in CFD are mostly based on the principle of weighted residuals. Amongst them, the Galerkin method is extensively used in which the weighting functions are taken as the same as the basis functions (interpolation functions). For solutions of hyperbolic problems, the Taylor-Galerkin method (Donea et al 1988) and the Petrov-Galerkin method (Hughes et al 1982) are widely used.

The finite element approach has been developed by mathematicians into a very elegant, rigorous, formal framework, with precise mathematical conditions for existence and convergence criteria and exactly derived error bounds. It can be used for structured and unstructured grid topology, and treats the boundary conditions more accurately than the finite difference approach. The shortcoming of this approach is the requirement of more coding work and more computational operation.

1.1.2.3 Finite Volume Methods

The finite volume method was apparently introduced into CFD independently by McDonald (1971) and MacCormack and Paullay (1972) for the solution of the two-dimensional, time-dependent Euler equations, and extended by Rizzi and Inouye (1973) to three-dimensional flows. This approach integrates the conservation form of the governing equations (in differential form) on a finite cell, where the primary variable is normally defined at the cell centre. All the spatial integrals, in finite volume methods, are approximated by the product of the spatial quantity and the average value of the integral. Thus, for a general problem where a coordinate transformation is needed, the finite volume methods involve the treatment of geometric terms such as volumes, areas, and normal velocity components rather than Jacobians, matrices, and contravariant velocity components used in the finite difference methods.

1.1.2.4 Spectral Methods

Spectral methods may be viewed as an extreme development of the class of discretization schemes known as the method of weighted residuals. The key elements of this approach are the basis functions and the weighting functions. The choice of the basis functions is one of the features which distinguishes spectral methods from finite element methods. The basis functions for spectral methods are infinitely differentiable global functions. In the case of finite element methods, the domain is subdivided into small elements, and a basis function is specified in each element. The basis functions are thus local in character, and well suited for handling complex geometries. The spectral methods may be considered as an extension of the finite element methods, and can be viewed as a whole space approximation technique. According to the choice of the weighting functions, there are three most commonly used spectral schemes, namely, the Galerkin scheme where the weighting functions are taken as the same as the basis functions; the collocation scheme where the weighting functions are the delta functions centered at collocation points; and the tau scheme which is similar to the Galerkin scheme in the way that the differential equation is enforced, but the weighting functions not needing to satisfy the boundary conditions. The spectral methods have been successful in incompressible flow simulation. For details, see, the books of Canuto et al (1987), Gottlieb and Orszag (1977), and the work of Ku et al (1985), Orszag (1980), Hussaini and Zang (1987), Streett (1987) and Kim and Moin (1985). For treating more general geometries, the so-called spectral element methods have been recently developed which can be considered as a cross between spectral methods and finite element methods. The successful applications of this scheme to fluid flow simulation were contributed by Patera (1984), Korczak and Patera (1986), Phillips and Karageorghis (1988), etc.

1.1.2.5 Boundary Integral Methods

The boundary integral methods are also called the boundary element methods. This approach is based on linear system analysis. For a given linear system, its fundamental solutions can be found, and in terms of the superposition theorem, the general solution of the system can be expressed as the linear combination of the fundamental solutions. Since all the fundamental solutions satisfy the system equation exactly, so does the general solution. Thus, for a particular problem, the determination of the coefficients in the linear combination form is only required. This can be carried out by replacing the general solution with boundary conditions. Thus, the boundary integral methods generally involve solution of unknowns on the boundary. The major advantage of this scheme is that the dimension of the problem can be reduced by one, and therefore, the computational complexity is reduced. Because this approach is based on linear analysis, it is difficult to apply to nonlinear problems. If it was thus used, some approximations need be introduced. Successful application of boundary integral methods to incompressible viscous flows has been achieved by Wu et al (1978, 1984).

1.1.3 Numerical Methods of the Resultant Equation System

After spatial discretization, the resultant semi-discrete equation system can be solved by available numerical techniques for the solution of ordinary differential equations. Amongst them, the multi-stage Runge-Kutta, Crank-Nicolson, Euler explicit, MacCormack explicit and predictor-corrector schemes, are widely used.

If the time derivative is discretized implicitly, the resultant equation system is a set of algebraic equations, which can be solved by two methods, namely, the direct method and the iterative method. The most frequently used direct methods are the Gaussian elimination method and the LU decomposition method. If the order of the algebraic equation system is very large, the direct method is less efficient because of storage problems and round-off errors. For this case, iterative methods are recommended. The basic iterative methods are the point Jacobi method and the point Gauss-Seidel method. Usually, the convergence rate of these methods are very slow. To increase the convergence rate, over-relaxation methods can be introduced. In this category, there are the Jacobi over-relaxation method, successive over-relaxation method (SOR) and successive line over-relaxation method (SLOR). Some preconditioning techniques such as the Richardson method, the strongly implicit procedure (SIP) and conjugate gradient method, can also improve the convergence rate.

To accelerate the convergence rate for a general problem, the multi-grid technique is a promising approach. The conventional iterative methods show slow convergence rate because of the poor damping of low frequency errors. In fact, high frequency errors are damped after a few iterations, but low frequency errors are not. The multi-grid approach can be considered as a smoother of errors in such a way that after one or more iteration sweeps through the mesh, the error behaviour is sufficiently smooth for it to be adequately represented on a coarser grid. The application of the multi-grid technique to CFD has demonstrated great success. For details, see the work of Ni (1982), Johnson (1983), Stuben and Trottenberg (1982), etc.

1.2 Challenges in CFD

Although some steady three-dimensional problems and some unsteady two-dimensional problems have been successfully simulated on modern supercomputers, we are still far from the achievement of full simulation of unsteady three-dimensional fluid flow problems because of lack of computation speed and storage memory. The current attempts at unsteady three-dimensional problems take tens or hundreds of hours on large mainframe computers available and even then have generally only limited spatial resolution. To tackle useful practical flow cases in CFD, fast numerical algorithms, and supercomputers with fast floating point operation speed and large virtual storage memory, need to be developed. To develop fast numerical algorithms, novel, flexible, efficient discretization schemes, which use few mesh points to achieve the same accuracy as conventional discretization schemes using a large number of mesh points, and the development of fast methods for the solution of the resulting algebraic equation system, are required. Since a fast numerical algorithm can greatly reduce both computer operation time and storage memory, practical unsteady three-dimensional flows can be realised on available supercomputers.

If supercomputers are developed with sufficient speed and memory, real flows may be realised using current numerical algorithms. But unfortunately, the development of single computers is limited because these computers rely on pipelines to obtain their speed, and the speed, at which information is propagated, is limited by the speed of light. Parallel supercomputers are being developed to overcome these difficulties. The architecture of the parallel supercomputer is quite different from the single supercomputer. Thus, it is necessary to explore flexible, robust parallel algorithms for use on these parallel supercomputer facilities. This is a new challenge in CFD at present.

1.3 The Scope of This Research

Towards meeting the above challenges, exploration of numerical algorithms and parallel computation was included in this research. It involves three topics: the development of new numerical techniques; the application of these to the simulation of incompressible viscous flows; and the parallel computation of incompressible Navier-Stokes flows.

In the numerical algorithm development, it is assumed that a function in the domain is sufficiently smooth that it can be approximated by a high order polynomial. The technique of generalized differential quadrature (GDQ), was firstly developed, based on the concept of the high order polynomial approximation in the overall domain, and the analysis of a linear vector space. The GDQ approach approximates the derivative of a function with respect to a coordinate direction as a weighted linear sum of all the functional values in that direction. The key to GDQ is how to determine the weighting coefficients for any order of derivatives. The details for determining the weighting coefficients of the **1D** case and extension to the multi-dimensional case have been given. The relationships between GDQ, finite difference schemes, and the Chebyshev pseudospectral method were also demonstrated. The basic properties of GDQ, such as error estimation, consistency, stability, influence of the distribution of grid points and the types of boundary conditions on the eigenvalues of the GDQ spatial discretization matrix, were also analysed. Based on the same concept as GDQ, a generalized integral quadrature (GIQ) technique was also presented. Using a linear sum of all the functional values in the whole domain to approximate an integral of a function over a part of the whole domain was studied. The study of the two-dimensional and three-dimensional integrals was also included.

For the numerical simulation, the fluid flow is assumed to be laminar, incompressible and Newtonian. Some standard examples of incompressible Navier-Stokes flows such as the driven cavity flow, the flow past a circular cylinder, the flow past a backward facing step and the natural convection in a square cavity, have been chosen to validate the numerical technique developed. Furthermore, the multi-domain GDQ technique with application to the flow past a backward facing step, the flow past a square step was also investigated. For the GIQ application, the **1D**, **2D**, **3D** steady and **2D** unsteady boundary layer equations were simulated.

In the part of the thesis involving parallel simulation, the **2D** incompressible Navier-Stokes equations were chosen for study. The multi-domain technique was used and the GDQ method was applied in each subdomain. All the cases are run on the Meiko Computing Surface (a transputer-based distributed memory multi-processor). The program on each processor was written in FORTRAN and run from an Occam harness to control the placement of and the communication between processors. The flow past a backward facing step using 3 slave processors and the flow past a square step using 5 slave processors were first investigated. For the driven cavity problem, the comparative study was made. Two formulations of the N-S equations (vorticity-stream function; vorticity-velocity) and three types of interface treatment (different number of grid points overlapped; patched with enforcing continuity condition of the function and its normal derivative; and patched with using Lagrange interpolation scheme), were studied. Finally, the development of a general code which can run on any array of transputers without any modification to the program was discussed.

1.4 Layout of This Thesis

The outline of this thesis is as follows: Chapter One consists of an introduction to the study. The description of the governing equations for the fluid flow is given in Chapter Two, and the details of the generalized differential and integral quadrature (GDQ and GIQ) techniques are given in Chapter Three. Chapter Four presents some analysis of the basic properties of the GDQ and GIQ, and application to standard model problems. The application of GDQ to the incompressible Navier-Stokes equations is given in Chapter Five, and the application of GDQ and GIQ to the boundary layer equations is given in Chapter Six. The parallel simulation of the incompressible Navier-Stokes equations is presented in Chapter Seven. Finally, some conclusions and prospects for future work are given in Chapter Eight.

CHAPTER TWO

GOVERNING EQUATIONS

2.1 Introduction

The basic equations of fluid dynamics are based on the universal laws of conservation, that is, the conservation of mass, momentum and energy. The equation derived from applying the law of conservation of mass to a fluid flow is usually called the continuity equation. The law of conservation of momentum is based on Newton's second law, which yields a vector equation known as the momentum equation. The law of conservation of energy is equivalent to the first law of thermodynamics and the resultant fluid dynamic equation is called the energy equation. In addition, the equation of state, which relates the thermodynamic variables of pressure (p), density (ρ), temperature (T), is needed in order to close the system of equations. For a general case if the velocity $\mathbf{V} = (u,v,w)$, any two thermodynamic variables and an equation of state are known, the complete description of a fluid is available. For some special cases, the complete governing equations can be simplified. In this chapter, we describe only the governing equation for incompressible viscous flows.

2.2 Incompressible Navier-Stokes Equations

2.2.1 Differential Form

The incompressible flow has a feature that

$$\frac{D\rho}{Dt} = 0 \tag{2.1}$$

where $\frac{D}{Dt}$ is a differential operator

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}$$

Substituting condition (2.1) into the compressible Navier-Stokes equations leads to

$$\nabla\cdot\mathbf{V} = 0 \tag{2.2}$$

$$\rho\frac{D\mathbf{V}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{V} + \rho\mathbf{f}_e \tag{2.3}$$

$$\rho c_v \frac{DT}{Dt} = k\,\nabla^2 T + \Phi + q_h \tag{2.4}$$

where ∇^2 is the Laplacian operator, μ is the dynamic viscosity of the fluid, $\mathbf{f}_e$ is the external volume force, q_h is the external volume heat source, c_v is the coefficient of specific heat under constant volume, k is the coefficient of thermal conductivity and

$$\Phi = \mu\Big[2(\frac{\partial u}{\partial x})^2 + 2(\frac{\partial v}{\partial y})^2 + 2(\frac{\partial w}{\partial z})^2 + (\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})^2 + (\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z})^2$$
$$+ (\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x})^2 - \frac{2}{3}(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z})^2\Big] \tag{2.5}$$

The rate of dissipation of mechanical energy is usually small in incompressible flows, thus Φ can be negligible in the equation (2.4). And only the case in which there is no external volume heat source, i.e. $q_h = 0$, is studied. So the energy equation can be reduced to

$$\rho c_v \frac{DT}{Dt} = k\,\nabla^2 T \tag{2.6}$$

In most cases, the temperature appears only in the energy equation so that we can uncouple this equation from the continuity and momentum equations. For many applications, the temperature changes are either insignificant or unimportant and it is not necessary to solve the energy equation. But in the natural convection case, the energy equation should be directly coupled with the momentum equations, because the buoyancy force, caused by the temperature distribution, is the dominant motive force of this system.

In natural convection, the external volume force is the gravitational force, that is

$$\rho \mathbf{f}_e = \rho \mathbf{g} \tag{2.7}$$

where $\mathbf{g}$ is the vector of acceleration due to gravity. It is common to use the Boussinesq approximation for the study of natural convection. The variation of density ρ is, according to the Boussinesq approximation, included only in the calculation of the buoyancy terms (i.e. $\rho\mathbf{g}$), and proportional to the temperature. The buoyancy force is written as

$$\rho \mathbf{f}_e = \rho_0 \, [1 - \beta(T - T_0)]\mathbf{g} \tag{2.8}$$

where ρ_0, T_0, are the reference density and temperature and β is the thermal expansion coefficient. Using (2.8) and the Boussinesq approximation, (2.3) is now simplified to

$$\frac{D\mathbf{V}}{Dt} = - \frac{1}{\rho_0} \nabla p + \nu \nabla^2 \mathbf{V} + \mathbf{g}\left[1 - \beta(T - T_0)\right] \tag{2.9}$$

where $\nu = \mu/\rho_0$ is the kinematic viscosity. In the Cartesian system, if the coordinate axis y is chosen to be the opposite direction of the gravity force, then

$$\mathbf{g} = (0, -g, 0) \tag{2.10}$$

For the natural convection in a square cavity, the scalar form of (2.9), in the two-dimensional case, is

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = - \frac{1}{\rho_0} \frac{\partial p}{\partial x} + \nu \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] \tag{2.11}$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = - \frac{1}{\rho_0} \frac{\partial p}{\partial y} + \nu \left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right] - g\left[1 - \beta(T - T_0)\right] \tag{2.12}$$

2.2.2 Non-Dimensional Form

For the case that only the continuity and momentum equations are solved, the flow variables and coordinates can be non-dimensionalized by

$$\begin{cases} x^* = \dfrac{x}{L} \quad y^* = \dfrac{y}{L} \quad z^* = \dfrac{z}{L} \quad t^* = \dfrac{t}{L/V_\infty} \\ u^* = \dfrac{u}{V_\infty} \quad v^* = \dfrac{v}{V_\infty} \quad w^* = \dfrac{w}{V_\infty} \quad p^* = \dfrac{p}{\rho_\infty V_\infty^2} \end{cases} \tag{2.13}$$

where the nondimensional variables are denoted by an asterisk, freestream conditions are denoted by ∞ and L is the reference length used in the Reynolds number

$$Re = \frac{\rho_\infty V_\infty L}{\mu}$$

Substituting the above relations into (2.2), (2.3) with $\mathbf{f}_e = 0$ yields the following nondimensional equations

continuity

$$\frac{\partial u^*}{\partial x^*} + \frac{\partial v^*}{\partial y^*} + \frac{\partial w^*}{\partial z^*} = 0 \tag{2.14}$$

x - momentum

$$\frac{\partial u^*}{\partial t^*} + u^* \frac{\partial u^*}{\partial x^*} + v^* \frac{\partial u^*}{\partial y^*} + w^* \frac{\partial u^*}{\partial z^*} = - \frac{\partial p^*}{\partial x^*} + \frac{1}{Re}\left[\frac{\partial^2 u^*}{\partial x^{*2}} + \frac{\partial^2 u^*}{\partial y^{*2}} + \frac{\partial^2 u^*}{\partial z^{*2}}\right] \tag{2.15}$$

y - momentum

$$\frac{\partial v^*}{\partial t^*} + u^* \frac{\partial v^*}{\partial x^*} + v^* \frac{\partial v^*}{\partial y^*} + w^* \frac{\partial v^*}{\partial z^*} = - \frac{\partial p^*}{\partial y^*} + \frac{1}{Re}\left[\frac{\partial^2 v^*}{\partial x^{*2}} + \frac{\partial^2 v^*}{\partial y^{*2}} + \frac{\partial^2 v^*}{\partial z^{*2}}\right] \tag{2.16}$$

z - momentum

$$\frac{\partial w^*}{\partial t^*} + u^* \frac{\partial w^*}{\partial x^*} + v^* \frac{\partial w^*}{\partial y^*} + w^* \frac{\partial w^*}{\partial z^*} = - \frac{\partial p^*}{\partial z^*} + \frac{1}{Re}\left[\frac{\partial^2 w^*}{\partial x^{*2}} + \frac{\partial^2 w^*}{\partial y^{*2}} + \frac{\partial^2 w^*}{\partial z^{*2}}\right] \qquad (2.17)$$

For two-dimensional problems, most researchers (de Vahl Davis, 1983; Ghia et al, 1982; Ku et al, 1985) favour the use of vorticity ω and stream function ψ as dependent variables

$$\omega = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \qquad (2.18)$$

$$u = \frac{\partial \psi}{\partial y}, \qquad v = - \frac{\partial \psi}{\partial x} \qquad (2.19)$$

Thus for natural convection problems, equations (2.11) and (2.12) can be combined to give

$$\frac{\partial \omega}{\partial t} + u \frac{\partial \omega}{\partial x} + v \frac{\partial \omega}{\partial y} = \nu\left[\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right] - g\beta \frac{\partial T}{\partial x} \qquad (2.20)$$

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \omega \qquad (2.21)$$

In (2.13), setting $V_\infty = k/(\rho_0 c_p L)$, $\rho_\infty = \rho_0$, and defining

$$T^* = \frac{T - T_0}{\Delta T}, \quad \omega^* = \frac{\omega}{V_\infty / L}, \quad \psi^* = \frac{\psi}{V_\infty L}$$

(2,20), (2.21), (2,6) can be nondimensionalized as

$$\frac{\partial \omega^*}{\partial t^*} + u^* \frac{\partial \omega^*}{\partial x^*} + v^* \frac{\partial \omega^*}{\partial y^*} = Pr\left[\frac{\partial^2 \omega^*}{\partial x^{*2}} + \frac{\partial^2 \omega^*}{\partial y^{*2}}\right] - Ra\, Pr \frac{\partial T^*}{\partial x^*} \qquad (2.22)$$

$$\frac{\partial T^*}{\partial t^*} + u^* \frac{\partial T^*}{\partial x^*} + v^* \frac{\partial T^*}{\partial y^*} = \frac{\partial^2 T^*}{\partial x^{*2}} + \frac{\partial^2 T^*}{\partial y^{*2}} \qquad (2.23)$$

$$\frac{\partial^2 \psi^*}{\partial x^{*2}} + \frac{\partial^2 \psi^*}{\partial y^{*2}} = \omega^* \qquad (2.24)$$

where Pr is Prandtl number, Ra is Rayleigh number

$$Pr = \frac{\mu c_p}{k}$$

$$Ra = \frac{c_p \rho_0 g \beta L^3 \Delta T}{k \nu}$$

For simplicity, the asterisks will be dropped, hereafter.

2.3 Boundary Layer Equations

Although many numerical solutions of the Navier-Stokes equations have been obtained on modern computers, the practical simulation of general Navier-Stokes flows is still elusive. This is true, especially for unsteady, three-dimensional flows at high Reynolds numbers since in this case, the step size near the solid boundary should be very small in order to capture the shear layer, and as a result, considerable computer time is required. In contrast, the use of the boundary layer concept or the principle of viscous-inviscid interaction can greatly reduce computer time. The concept of the boundary layer was firstly presented by Prandtl, deriving from his experimental observation in 1904. Following this concept, the whole flow field can be split into two regions: the viscous shear layer near the wall, which is governed by the boundary layer equations, and the remaining inviscid region which is governed by the Euler or potential equations. The governing equations in two regions can be solved separately or coupled in the case of viscous-inviscid interaction.

The boundary layer equations can be obtained from the Navier-Stokes equations, by using order of magnitude analysis with two constraints. These are: (1) the viscous layer must be thin relative to the characteristic streamwise dimension of the object immersed in the flow; (2) the largest viscous term must be of the same approximate magnitude as any inertia term. For the three dimensional, incompressible flow past a flat plate, let x and z denote the coordinates in the wall surface and y denote the coordinate which is perpendicular to the wall. The three dimensional equation, for this case, can be written as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.25}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} \tag{2.26}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\frac{\partial^2 w}{\partial y^2} \tag{2.27}$$

with

$$-\frac{1}{\rho}\frac{\partial p}{\partial x} = \frac{\partial Ue}{\partial t} + Ue\frac{\partial Ue}{\partial x} + We\frac{\partial Ue}{\partial z} \tag{2.28}$$

$$-\frac{1}{\rho}\frac{\partial p}{\partial z} = \frac{\partial We}{\partial t} + Ue\frac{\partial We}{\partial x} + We\frac{\partial We}{\partial z} \tag{2.29}$$

where Ue, We are the components of the outer flow velocity in the x and z direction.

2.4 Governing Equations in the General Coordinate System

2.4.1 2D Incompressible NS Equations in the General Coordinate System

For the two dimensional case, supposing the physical coordinates x, y are transformed into the general coordinates ξ, η by

$$\begin{cases} \xi = \xi(x, y) \\ \eta = \eta(x, y) \end{cases} \tag{2.30}$$

we then obtain

$$J = x_\xi y_\eta - x_\eta y_\xi \tag{2.31}$$

$$\begin{aligned} \xi_x &= J^{-1} y_\eta , \quad \eta_x = - J^{-1} y_\xi , \\ \xi_y &= - J^{-1} x_\eta , \quad \eta_y = J^{-1} x_\xi . \end{aligned} \tag{2.32}$$

Using the chain rule of partial differentiation and the following divergence formulation

$$\nabla\cdot\mathbf{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} = \frac{1}{J}\left[\frac{\partial(y_\eta F_1 - x_\eta F_2)}{\partial\xi} + \frac{\partial(x_\xi F_2 - y_\xi F_1)}{\partial\eta}\right], \tag{2.33}$$

the form of the two-dimensional, incompressible Navier-Stokes equations, in the general coordinate system, is written as

continuity

$$\frac{1}{J}\left[y_\eta \frac{\partial u}{\partial\xi} - y_\xi \frac{\partial u}{\partial\eta} + x_\xi \frac{\partial v}{\partial\eta} - x_\eta \frac{\partial v}{\partial\xi}\right] \tag{2.34}$$

x-momentum

$$\frac{\partial u}{\partial t} + \frac{1}{J}\left[\frac{\partial(Uu)}{\partial\xi} + \frac{\partial(Vu)}{\partial\eta}\right] + \frac{1}{J}\left[y_\eta \frac{\partial p}{\partial\xi} - y_\xi \frac{\partial p}{\partial\eta}\right]$$

$$= \frac{1}{J}\left\{\frac{\partial}{\partial\xi}\left[A \frac{\partial u}{\partial\xi} + B \frac{\partial u}{\partial\eta}\right] + \frac{\partial}{\partial\eta}\left[C \frac{\partial u}{\partial\eta} + D \frac{\partial u}{\partial\xi}\right]\right\} \tag{2.35}$$

y-momentum

$$\frac{\partial v}{\partial t} + \frac{1}{J}\left[\frac{\partial(Uv)}{\partial\xi} + \frac{\partial(Vv)}{\partial\eta}\right] + \frac{1}{J}\left[- x_\eta \frac{\partial p}{\partial\xi} + x_\xi \frac{\partial p}{\partial\eta}\right]$$

$$= \frac{1}{J}\left\{\frac{\partial}{\partial\xi}\left[A \frac{\partial v}{\partial\xi} + B \frac{\partial v}{\partial\eta}\right] + \frac{\partial}{\partial\eta}\left[C \frac{\partial v}{\partial\eta} + D \frac{\partial v}{\partial\xi}\right]\right\} \tag{2.36}$$

where

$$U = uy_\eta - vx_\eta, \quad V = vx_\xi - uy_\xi$$

$$A = \frac{\alpha}{Re\cdot J}, \quad B = \frac{-\beta}{Re\cdot J}, \quad C = \frac{\gamma}{Re\cdot J}, \quad D = \frac{-\beta}{Re\cdot J}$$

$$\alpha = x_\eta^2 + y_\eta^2, \quad \beta = x_\eta x_\xi + y_\eta y_\xi, \quad \gamma = x_\xi^2 + y_\xi^2$$

If the vorticity-stream function formulation or the vorticity-velocity formulation is used, their expressions, in the general coordinate system, are

vorticity equation

$$\frac{\partial\omega}{\partial t} + \frac{1}{J}\left[\frac{\partial(U\omega)}{\partial\xi} + \frac{\partial(V\omega)}{\partial\eta}\right]$$

$$= \frac{1}{J}\left\{\frac{\partial}{\partial\xi}\left[A \frac{\partial\omega}{\partial\xi} + B \frac{\partial\omega}{\partial\eta}\right] + \frac{\partial}{\partial\eta}\left[C \frac{\partial\omega}{\partial\eta} + D \frac{\partial\omega}{\partial\xi}\right]\right\} \tag{2.37}$$

stream function equation

$$\frac{\partial}{\partial \xi}\left[A_1 \frac{\partial \psi}{\partial \xi} + B_1 \frac{\partial \psi}{\partial \eta}\right] + \frac{\partial}{\partial \eta}\left[C_1 \frac{\partial \psi}{\partial \eta} + D_1 \frac{\partial \psi}{\partial \xi}\right] = J\omega \tag{2.38}$$

u-equation

$$\frac{\partial}{\partial \xi}\left[A_1 \frac{\partial u}{\partial \xi} + B_1 \frac{\partial u}{\partial \eta}\right] + \frac{\partial}{\partial \eta}\left[C_1 \frac{\partial u}{\partial \eta} + D_1 \frac{\partial u}{\partial \xi}\right]$$
$$= x_\xi \frac{\partial \omega}{\partial \eta} - x_\eta \frac{\partial \omega}{\partial \xi} \tag{2.39}$$

v-equation

$$\frac{\partial}{\partial \xi}\left[A_1 \frac{\partial v}{\partial \xi} + B_1 \frac{\partial v}{\partial \eta}\right] + \frac{\partial}{\partial \eta}\left[C_1 \frac{\partial v}{\partial \eta} + D_1 \frac{\partial v}{\partial \xi}\right]$$
$$= -\left[y_\eta \frac{\partial \omega}{\partial \xi} - y_\xi \frac{\partial \omega}{\partial \eta}\right] \tag{2.40}$$

where

$$A_1 = Re\cdot A, \qquad B_1 = Re\cdot B, \qquad C_1 = Re\cdot C, \qquad D_1 = Re\cdot D,$$

Similarly, the governing equations for natural convection problems with vorticity-stream function formulation, in the general coordinate system, can be written as

$$\frac{\partial \omega}{\partial t} + \frac{1}{J}\left[\frac{\partial (U\omega)}{\partial \xi} + \frac{\partial (V\omega)}{\partial \eta}\right] = -\frac{Ra\cdot Pr}{J}\left[y_\eta \frac{\partial T}{\partial \xi} - y_\xi \frac{\partial T}{\partial \eta}\right]$$
$$+ \frac{Pr}{J}\left\{\frac{\partial}{\partial \xi}\left[A_1 \frac{\partial \omega}{\partial \xi} + B_1 \frac{\partial \omega}{\partial \eta}\right] + \frac{\partial}{\partial \eta}\left[C_1 \frac{\partial \omega}{\partial \eta} + D_1 \frac{\partial \omega}{\partial \xi}\right]\right\} \tag{2.41}$$

$$\frac{\partial T}{\partial t} + \frac{1}{J}\left[\frac{\partial (UT)}{\partial \xi} + \frac{\partial (VT)}{\partial \eta}\right]$$
$$= \frac{1}{J}\left\{\frac{\partial}{\partial \xi}\left[A_1 \frac{\partial T}{\partial \xi} + B_1 \frac{\partial T}{\partial \eta}\right] + \frac{\partial}{\partial \eta}\left[C_1 \frac{\partial T}{\partial \eta} + D_1 \frac{\partial T}{\partial \xi}\right]\right\} \tag{2.42}$$

The stream function equation is the same as (2.38).

2.4.2 Incompressible Boundary Layer Equations for a General Case

If the boundary layer for the flow past a general geometry is investigated, the study is always based on the streamwise coordinate system. This system is usually not a Cartesian coordinate system, but an orthogonal coordinate system. Equations (2.25)-(2.27) can be written, in this system, as

$$\frac{\partial(uh_3)}{\partial\xi} + \frac{\partial(vh_1h_3)}{\partial\eta} + \frac{\partial(wh_1)}{\partial\zeta} = 0 \qquad (2.43)$$

$$\frac{\partial u}{\partial t} + \frac{u}{h_1}\cdot\frac{\partial u}{\partial\xi} + v\frac{\partial u}{\partial\eta} + \frac{w}{h_3}\cdot\frac{\partial u}{\partial\zeta} + u\cdot w\cdot K_1 - w^2\cdot K_2 = -\frac{1}{\rho\cdot h_1}\frac{\partial p}{\partial\xi} + \nu\frac{\partial^2 u}{\partial\eta^2} \qquad (2.44)$$

$$\frac{\partial w}{\partial t} + \frac{u}{h_1}\cdot\frac{\partial w}{\partial\xi} + v\frac{\partial w}{\partial\eta} + \frac{w}{h_3}\cdot\frac{\partial w}{\partial\zeta} + u\cdot w\cdot K_2 - u^2\cdot K_1 = -\frac{1}{\rho\cdot h_3}\frac{\partial p}{\partial\zeta} + \nu\frac{\partial^2 w}{\partial\eta^2} \qquad (2.45)$$

where

$$K_1 = \frac{1}{h_1\cdot h_3}\cdot\frac{\partial h_1}{\partial\zeta}, \qquad K_2 = \frac{1}{h_1\cdot h_3}\cdot\frac{\partial h_3}{\partial\xi}$$

$$(h_1)^2 = (x_\xi)^2 + (y_\xi)^2 + (z_\xi)^2$$

$$(h_3)^2 = (x_\zeta)^2 + (y_\zeta)^2 + (z_\zeta)^2$$

and ξ is directed in the primary flow direction, ζ is in the crossflow direction, and η is orthogonal to the body surface.

In order to remove the singularity for solving (2.25)-(2.27), the coordinate transformation is always required. Using the following transformation

$$T = Ue\cdot t/x, \qquad \xi = x, \qquad \zeta = z, \qquad \eta = (Ue/\nu\cdot x)^{1/2}\cdot y$$

$$F = u/Ue, \qquad G = w/We$$

equations (2.25)-(2.27) can be transformed to

$$\xi\cdot\frac{\partial F}{\partial\xi} + \frac{\partial V}{\partial\eta} + \frac{F}{2}\left[1 + K_3\right] + \frac{\xi\cdot We}{Ue}\cdot\frac{\partial G}{\partial\zeta} + G\cdot\left[K_5 - 0.5K_4\right] = 0 \qquad (2.46)$$

$$\frac{\partial F}{\partial T} + \xi F\cdot\frac{\partial F}{\partial\xi} + V\cdot\frac{\partial F}{\partial\eta} + \xi G\cdot\frac{We}{Ue}\cdot\frac{\partial F}{\partial\zeta} + K_3\cdot\left[F^2-1\right] + K_4\cdot\left[FG-1\right] = \frac{\partial^2 F}{\partial\eta^2} \qquad (2.47)$$

$$\frac{\partial G}{\partial T} + \xi F\cdot\frac{\partial G}{\partial\xi} + V\cdot\frac{\partial G}{\partial\eta} + \xi G\cdot\frac{We}{Ue}\cdot\frac{\partial G}{\partial\zeta} + K_5\cdot\left[G^2-1\right] + K_6\cdot\left[FG-1\right] = \frac{\partial^2 G}{\partial\eta^2} \qquad (2.48)$$

where

$$V = (x/\nu\cdot Ue)^{1/2}\cdot v + xF\cdot\frac{\partial\eta}{\partial x} + xG\cdot\frac{We}{Ue}\cdot\frac{\partial\eta}{\partial z}$$

$$K_3 = \frac{x}{Ue}\cdot\frac{\partial Ue}{\partial x}, \qquad K_4 = \frac{x}{Ue}\cdot\frac{We}{Ue}\cdot\frac{\partial Ue}{\partial z}$$

$$K_5 = \frac{x}{Ue}\cdot\frac{\partial We}{\partial z}, \qquad K_6 = \frac{x}{We}\cdot\frac{\partial We}{\partial x}$$

CHAPTER THREE

GENERALIZED DIFFERENTIAL AND INTEGRAL QUADRATURE

3.1 Introduction

The numerical techniques for the solution of a partial differential equation can be classified into two categories. One is based on the direct discretization of the derivatives and integrals. Another is based on the variational principles or the principles of weighted residuals. The conventional finite difference methods lie in the first category while the finite element and the spectral methods are in the second. Usually, low order methods such as finite differences and finite elements can provide accurate results by using a large number of grid points. However, in some practical applications the numerical solution of a governing equation is required at only a few specified points in a domain. But for acceptable accuracy, conventional finite difference and finite element methods also require the use of a large number of grid points to obtain the solution at those specified points. In seeking a more efficient method using just a few grid points to get an accurate result, Bellman et al (1972) introduced a method of differential quadrature, where a partial derivative of a function with respect to a coordinate direction is expressed as a linear weighted sum of all the functional variables at all mesh points along that direction. It is clear that this method is based on the direct discretization of the derivative, and therefore, is in the first category indicated above. Preliminary computational results (Bellman et al 1984, Civan et al 1983, 1984, Mingle 1977, Jang et al 1989) showed that differential quadrature has potential as an attractive approximation technique. The key technique to differential quadrature is how to determine the weighting coefficients for the discretization of any order partial derivative. To determine the weighting coefficients of the first order derivative, Bellman et al suggested two methods to carry this out. One method solves a set of algebraic equations which is obtained by satisfying the linear constrained relation for all polynomials of degree less than or equal to N-1, where N is the total number of grid points in a domain. This equation system has a unique solution because the matrix elements are composed of a Vandermonde matrix. Unfortunately, when N is large the inversion of this matrix becomes difficult. This is probably one of the reasons that applications of this scheme so far only use the number of grid points less than or equal to 13. The second method is to compute the weighting coefficient a_{ij} by an algebraic formulation with coordinates of grid points chosen as the roots of an Nth order shifted Legendre polynomial. This means that if N is specified, the distributions of grid points (very close to the boundary) are the same for different physical problems. This can provide a major drawback and restrict the application of differential quadrature. In order to overcome this drawback, the generalized differential quadrature technique was developed in this work, based on the analysis of the high order polynomial approximation in the overall domain.

3.2 Differential Quadrature

For the one dimensional unsteady problem, Bellman et al (1972) assume a function u(x,t) to be sufficiently smooth to allow the following linear constrained relation to be satisfied

$$u_x(x_i,\ t) = \sum_{j=1}^{N} a_{ij}\cdot u(x_j,\ t) \qquad (3.1)$$

for i=1, 2, ···, N

where $u_x(x_i,\ t)$ indicates the first order derivative of u(x,t) with respect to x at x_i. Substituting (3.1) into a time-dependent partial differential equation yields a set of ordinary differential equations which can be integrated by the well-developed schemes such as Runge-Kutta and Adams-Moulton.

The key technique to this procedure is to determine the weighting coefficients a_{ij}. Bellman et al suggested two ways to carry this out. The first way is to let (3.1) be exact for all polynomials of degree less than or equal to N-1 test functions $g(x)=x^k$, k=0, 1, ···, N-1, which leads to a set of linear algebraic equations

$$\sum_{j=1}^{N} a_{ij}\cdot x_j^{\,k} = k\cdot x_i^{\,k-1} \qquad (3.2)$$

for k = 0, 1, ···, N-1

i = 1, 2, ···, N

This equation system has a unique solution since its matrix is of Vandermonde form. Unfortunately, when N is large, this matrix is ill-conditioned and its inversion is difficult.

Another method is similar to the first one with an exception that the different test function

$$g(x)= \frac{L_N(x)}{(x - x_k)\cdot L'_N(x_k)} \qquad (3.3)$$

is chosen, where $L_N(x)$ is the *N*th order Legendre polynomial and $L'_N(x)$

the first order derivative of $L_N(x)$. By choosing x_k to be the roots of the shifted Legendre polynomial, Bellman et al obtained a simple algebraic formulation for a_{ij}

$$a_{ij} = \frac{L'_N(x_i)}{(x_i - x_j)\cdot L'_N(x_j)}, \qquad i \neq j \tag{3.4a}$$

$$a_{ii} = \frac{1-2x_i}{2x_i\cdot(x_i - 1)} \tag{3.4b}$$

3.3 Generalized Differential Quadrature

In order to overcome the drawback described above for differential quadrature and to obtain a similar simple formulation for a_{ij}, a method of generalized differential quadrature has been introduced in this work, based on the analysis of the polynomial vector space.

3.3.1 High Order Polynomial Approximation in the Overall Domain

Since any finite range can be transformed into the range of [0, 1] by a simple transformation, we will consider only the range [0, 1] hereafter. It is well known that a continuous function f(x) in the interval [0, 1] can be approximated by an infinite polynomial accurately in accordance with the Weierstrass polynomial approximation theorem. In practice, a truncated finite polynomial may be used. Some methods, an example being the spectral method, have successfully applied the concept of high order polynomial approximation to the solution of the partial differential equation. Following this approach, it is supposed that any smooth function in the interval [0, 1] can be approximated by the (N-1)th order polynomial.

It is easy to show that the polynomial of degree less than or equal to N-1 constitutes an N-dimensional vector space V_N with respect to the operation of addition and multiplication. From the concept of linear independence, the bases of a vector space can be considered as a linearly independent subset which spans the entire space. Here if $r_k(x)$, k=1, 2, $\cdots$, N, which are in the space V_N, are the base polynomials, any polynomial in V_N can be expressed as a linear combination of $r_k(x)$, k=1, 2, $\cdots$, N, i.e

$$\bar{f}(x) = P_N f = \sum_{k=1}^{N} c_k \cdot r_k(x) \tag{3.5}$$

where P_N is a projection operator of smooth function onto V_N, c_k is a coefficient, and $\bar{f}(x)$, $r_k(x)$ are in space V_N. The spectral method uses a high order polynomial similar to (3.5) to approximate the function f(x) in the overall domain. But the procedures for the solution of the partial differential equation are quite different. The spectral method, which is based on the principle of the weighted residuals, involves the determination of the coefficients of the base polynomials, namely, c_k, while generalized differential quadrature (to be described), which uses this formulation only to determine the weighting coefficients for discretization of any order (less than N) partial derivative, involves the determination of the functional values at grid points.

3.3.2 Determination of Weighting Coefficients for First Derivative

Equation (3.1) is a linear constrained relationship. If the base polynomials $r_k(x)$, k=1, 2, $\cdots$, N, satisfy (3.1), so does polynomial $\bar{f}(x)$. And if the base polynomial $r_k(x)$ is chosen to be x^{k-1}, the same equation system as (3.2), given by Bellman's first method, can be obtained. For generality, here the base polynomial $r_k(x)$ is chosen to be the Lagrange interpolation polynomial

$$r_k(x) = \frac{M(x)}{(x - x_k)\cdot M^{(1)}(x_k)} \tag{3.6}$$

where $M(x) = (x-x_1)\cdot(x-x_2)\cdots(x-x_N)$

$$M^{(1)}(x_k) = \prod_{j=1,\, j\neq k}^{N}(x_k - x_j)$$

For simplicity, we set

$$M(x) = N(x,\ x_j)\cdot(x - x_j),\ j=1,\ \cdots,\ N$$

with $N(x_i,\ x_j) = M^{(1)}(x_i)\cdot\delta_{ij}$, where δ_{ij} is the Kronecker operator.

Thus we have

$$M^{(k)}(x) = N^{(k)}(x,\ x_j)\cdot(x - x_j) + k\cdot N^{(k-1)}(x,\ x_j) \tag{3.7}$$

for k=1, 2, $\cdots$, N-1

where $M^{(k)}(x)$, $N^{(k)}(x,\ x_j)$ indicate the *k*th order derivative of M(x) and $N(x,\ x_j)$. Substituting (3.6) into (3.1) yields

$$a_{ij} = \frac{N^{(1)}(x_i,\ x_j)}{M^{(1)}(x_j)} \tag{3.8}$$

From (3.7), we get

$$N^{(1)}(x_i,\ x_j) = \frac{M^{(1)}(x_i)}{x_i - x_j}, \qquad \text{for } i\neq j$$

$$N^{(1)}(x_i,\ x_i) = \frac{M^{(2)}(x_i)}{2}$$

So, (3.8) can be written as

$$a_{ij} = \frac{M^{(1)}(x_i)}{(x_i-x_j)\cdot M^{(1)}(x_j)}, \qquad \text{for } i\neq j \tag{3.9a}$$

$$a_{ii} = \frac{M^{(2)}(x_i)}{2\cdot M^{(1)}(x_i)} \tag{3.9b}$$

for i, j = 1, 2, $\cdots$, N

Equation (3.9) is a simple formulation for computing a_{ij} without any restriction on choice of the coordinates of the grid points x_i. Actually, if x_i is given, it is easy to compute $M^{(1)}(x_i)$, thus a_{ij} for $i \neq j$. The calculation of a_{ii} is based on the computation of the second order derivative $M^{(2)}(x_i)$ which is not easy to be obtained. Next, it will be shown that a_{ii} can be calculated from a_{ij} ($i \neq j$).

According to the theory of a linear vector space, one set of base polynomials can be expressed uniquely by another set of base polynomials. Thus if one set of base polynomials satisfies a linear constrained relationship, say (3.1), so does another set of base polynomials. And since the weighting coefficients are only dependent on the coordinates of grid points if the number of grid points is given, the equation system for determination of a_{ij} derived from one set of base polynomials should be equivalent to that derived from other sets of base polynomials. Thus a_{ij} satisfies the following equation which is obtained by the base polynomial x^k when k=0

$$\sum_{j=1}^{N} a_{ij} = 0 , \tag{3.10}$$

where a_{ii} can be easily determined from a_{ij} ($j \neq i$) using (3.10). Equation (3.9) is a general form for calculating a_{ij}. It follows that if the coordinates of the grid points are chosen as the roots of a shifted Legendre polynomial, (3.9) is exactly the same as that given by Bellman's second method.

3.3.3 Determination of Weighting Coefficients for Higher Derivatives

For discretization of the second order derivative, we introduce the following linear constrained relation

$$u_{xx}(x_i,\ t) = \sum_{j=1}^{N} b_{ij}\cdot u(x_j,\ t) \tag{3.11}$$

for i=1, 2, $\cdots$, N

where $u_{xx}(x,\ t)$ is the second order derivative of $u(x,\ t)$ with respect to x, and Lagrange interpolated polynomials are chosen as the base polynomials (see 3.6). Using the same approach as for the first order derivative, the weighting coefficients b_{ij} become

$$b_{ij} = \frac{N^{(2)}(x_i,\ x_j)}{M^{(1)}(x_j)} \tag{3.12}$$

From (3.7), we obtain

$$N^{(2)}(x_i,\ x_j) = \frac{M^{(2)}(x_i)-2\cdot N^{(1)}(x_i,\ x_j)}{x_i - x_j}, \qquad i\neq j \tag{3.13a}$$

$$N^{(2)}(x_i,\ x_i) = \frac{M^{(3)}(x_i)}{3} \tag{3.13b}$$

Substituting (3.13), (3.9) into (3.12) yields

$$b_{ij} = 2\cdot a_{ij}\cdot(a_{ii} - \frac{1}{x_i - x_j}), \qquad \text{for } j\neq i \tag{3.14a}$$

$$b_{ii} = \frac{M^{(3)}(x_i)}{3\cdot M^{(1)}(x_i)} \tag{3.14b}$$

for i, j = 1, 2, $\cdots$, N

For $i\neq j$, b_{ij} can be calculated from a_{ij} without a double summation. In a similar analysis to the case of the first order derivative, the equation system for b_{ij} derived from the above Lagrange interpolated polynomials is equivalent to that derived from the base polynomials x^k, k = 0, 1, $\cdots$, N-1. Thus b_{ij} should also satisfy the following formulation derived from the base polynomial x^k when k=0

$$\sum_{j=1}^{N} b_{ij} = 0 \tag{3.15}$$

from which b_{ii} can be easily determined.

Furthermore, in the case of the discretization of the higher order derivative, the linear constrained relations are applied as follows

$$u_x^{(m-1)}(x_i, t) = \sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot u(x_j, t) \qquad (3.16)$$

$$u_x^{(m)}(x_i, t) = \sum_{j=1}^{N} w_{ij}^{(m)} \cdot u(x_j, t) \qquad (3.17)$$

for $i = 1, 2, \cdots, N$

where $u_x^{(m-1)}(x_i,t)$, $u_x^{(m)}(x_i,t)$ indicate the (*m*-1)th and *m*th order derivative of u(x,t) with respect to x at x_i, $w^{(m-1)}$, $w^{(m)}$ the weighting coefficients related to $u_x^{(m-1)}(x_i,t)$ and $u_x^{(m)}(x_i,t)$. Substituting (3.6) into (3.16), (3.17) and using (3.7), (3.9), a recurrence formulation is obtained as follows

$$w_{ij}^{(m)} = m \cdot \left(a_{ij} \cdot w_{ii}^{(m-1)} - \frac{w_{ij}^{(m-1)}}{x_i - x_j}\right), \qquad j \neq i \qquad (3.18)$$

for $m = 2, \cdots, N-1$; $\quad i, j = 1, 2, \cdots, N$

where a_{ij} is the weighting coefficients of the first order derivative described above. Again, in terms of the analysis of the N-dimensional linear vector space, the equation system for $w_{ij}^{(m)}$ derived from Lagrange interpolated polynomials should be equivalent to that derived from the base polynomials x^k, k=0, 1, $\cdots$, N-1. Thus $w_{ij}^{(m)}$ should satisfy the following equation obtained from the base polynomial x^k when k=0

$$\sum_{j=1}^{N} w_{ij}^{(m)} = 0 \qquad (3.19)$$

From this formulation, $w_{ii}^{(m)}$ can be easily calculated from $w_{ij}^{(m)}$, $i \neq j$.

3.3.4 Extention to the Multi-Dimensional Case

For the two-dimensional approximation of a function f(x,y) in the domain $x \in [0, 1]$, $y \in [0, 1]$, it is supposed that the value of $f(x,b)$, where b is a constant, $b \in [0, 1]$, can be approximated by an (N-1)th order polynomial $P_N(x)$ which constitutes an N-dimensional vector space V_N with N base polynomials $r_i(x)$, $i=1,2,\cdots,N$, and the value of $f(a,y)$, where a is a constant, $a \in [0, 1]$, can be approximated by an (M-1)th order polynomial $P_M(y)$ which constitutes an M-dimensional vector space V_M with M base polynomials $s_j(y)$, $j=1,2,\cdots,M$. The value of function $f(x,y)$ can be approximated by the polynomial $Q_{N\times M}(x,y)$ with the form

$$Q_{N\times M}(x,y) = \sum_{i=1}^{N} \sum_{j=1}^{M} \bar{c}_{ij}\cdot x^{i-1}\cdot y^{j-1} \qquad (3.20)$$

where $\bar{c}_{ij}$ is a coefficient

It is clear that $Q_{N\times M}(x,y)$ constitutes a N×M dimensional polynomial vector space $V_{N\times M}$ with respect to the operation of addition and scalar multiplication. It will now be shown that $\Phi_{ij} = r_i(x)\cdot s_j(y)$ constitutes the base polynomials in the vector space $V_{N\times M}$. Since $r_i(x)$, $s_j(y)$ are the base polynomials of $P_N(x)$ and $P_M(y)$, they must be linearly independent, that is

$$\sum_{i=1}^{N} c_i\cdot r_i(x) = 0 \qquad \text{only if} \quad c_i = 0, \quad i=1,2,\cdots,N \qquad (3.21)$$

$$\sum_{j=1}^{M} d_j\cdot s_j(y) = 0 \qquad \text{only if } d_j = 0, \quad j=1,2,\cdots,M \qquad (3.22)$$

Now we see that if

$$\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij}\cdot \Phi_{ij}(x,y) = 0 \quad , \quad \text{i.e.} \quad \sum_{i=1}^{N} (\sum_{j=1}^{M} c_{ij}\cdot s_j(y))\cdot r_i(x) = 0$$

From (3.21) the following equation is obtained

$$\sum_{j=1}^{M} c_{ij}\cdot s_j(y) = 0$$

Finally from (3.22) we obtain $c_{ij} = 0$. Then, $\Phi_{ij}(x,y)$ constitutes the base polynomials in $V_{N\times M}$.

Now it is assumed that the following constrained relations are satisfied for function u(x,y,t) and its first order spatial derivative

$$u_x(x_i,y_j,t) = \sum_{k=1}^{N} a^x_{ik}\cdot u(x_k,y_j,t) \tag{3.23}$$

$$u_y(x_i,y_j,t) = \sum_{k=1}^{M} a^y_{jk}\cdot u(x_i,y_k,t) \tag{3.24}$$

for $i=1,2,\cdots,N$; $j=1,2,\cdots,M$

where a^x_{ik}, a^y_{jk} are the weighting coefficients related to $u_x(x_i,y_j,t)$ and $u_y(x_i,y_j,t)$ respectively. If all the base polynomials $\Phi_{ij}(x,y)$ satisfy (3.23), (3.24), then so does any polynomial in $V_{N\times M}$. Substituting $\Phi_{ij}(x,y)$ into (3.23), (3.24) leads to

$$\sum_{k=1}^{N} a^x_{ik}\cdot r_j(x_k) = r_j'(x_i) \tag{3.25}$$

$$\sum_{k=1}^{M} a^y_{ik}\cdot s_j(y_k) = s_j'(y_i) \tag{3.26}$$

where $r_j'(x_i)$ represents the first order derivative of $r_j(x)$ at x_i and $s_j'(y_i)$ represents the first order derivative of $s_j(y)$ at y_i. From (3.25), (3.26), it is obvious that a^x_{ik} or a^y_{jk} is only related to $r_i(x)$ or $s_j(y)$. Hence the formulation of the one dimensional case can be directly extended to the two dimensional case, that is

$$a^x_{ij} = \frac{M^{(1)}(x_i)}{(x_i-x_j)\cdot M^{(1)}(x_j)}, \qquad \text{for } i\neq j \tag{3.27a}$$

$$a^x_{ii} = -\sum_{j=1,\,j\neq i}^{N} a^x_{ij} \tag{3.27b}$$

for $i,j = 1,2,\cdots,N$

$$a^y_{ij} = \frac{P^{(1)}(y_i)}{(y_i-y_j)\cdot P^{(1)}(y_j)}, \qquad \text{for } i\neq j \tag{3.28a}$$

$$a^y_{ii} = -\sum_{j=1,\,j\neq i}^{M} a^y_{ij} \tag{3.28b}$$

for $i,j = 1,2,\cdots,M$

where

$$M^{(1)}(x_i) = \prod_{j=1,j\neq i}^{N} (x_i - x_j)$$

$$P^{(1)}(y_i) = \prod_{j=1,j\neq i}^{M} (y_i - y_j)$$

Similarly, for the second or higher order derivative the recurrence relationship of the weighting coefficients can be obtained as follows

$$w_{ij}^{(n)} = n\cdot(a_{ij}^{x}\cdot w_{ii}^{(n-1)} - \frac{w_{ij}^{(n-1)}}{x_i - x_j}), \qquad j\neq i \qquad (3.29a)$$

$$w_{ii}^{(n)} = -\sum_{j=1,j\neq i}^{N} w_{ij}^{(n)} \qquad (3.29b)$$

for $n=2,3,\cdots,N-1$; $i,j = 1,2,\cdots,N$

$$\bar{w}_{ij}^{(m)} = m\cdot(a_{ij}^{y}\cdot \bar{w}_{ii}^{(m-1)} - \frac{\bar{w}_{ij}^{(m-1)}}{y_i - y_j}), \qquad j\neq i \qquad (3.30a)$$

$$\bar{w}_{ii}^{(m)} = -\sum_{j=1,j\neq i}^{M} \bar{w}_{ij}^{(m)} \qquad (3.30b)$$

for $m=2,3,\cdots,M-1$; $i,j = 1,2,\cdots,M$

where $w_{ij}^{(n)}$ are the weighting coefficients of the *n*th order derivative of $u(x,y,t)$ with respect to x at x_i, y_j, namely, $u_x^{(n)}(x_i,y_j,t)$, and $\bar{w}_{ij}^{(m)}$ the weighting coefficients of the *m*th order derivative of $u(x,y,t)$ with respect to y at x_i, y_j, namely, $u_y^{(m)}(x_i,y_j,t)$. They satisfy

$$u_x^{(n)}(x_i,y_j,t) = \sum_{k=1}^{N} w_{ik}^{(n)}\cdot u(x_k,y_j,t) \qquad (3.31)$$

$$u_y^{(m)}(x_i,y_j,t) = \sum_{k=1}^{M} \bar{w}_{jk}^{(m)}\cdot u(x_i,y_k,t) \qquad (3.32)$$

for $i=1,2,\cdots,N$; $j=1,2,\cdots,M$

Similar formulations can be obtained for the three dimensional case.

If the functional values at all grid points are obtained, it is easy to determine the functional values in the overall domain in terms of the polynomial approximation, i.e.

$$u(x,y_j) = \sum_{i=1}^{N} u(x_i,y_j)\cdot r_i(x), \qquad j=1,2,\cdots,M \tag{3.33a}$$

$$u(x_i,y) = \sum_{j=1}^{M} u(x_i,y_j)\cdot s_j(y), \qquad i=1,2,\cdots,N \tag{3.33b}$$

$$u(x,y) = \sum_{i=1}^{N} \sum_{j=1}^{M} u(x_i,y_j)\cdot r_i(x)\cdot s_j(y) \tag{3.33c}$$

where $r_i(x)$, $s_j(y)$ are the Lagrange interpolated polynomials along the x and y direction respectively.

3.3.5 Comparison with the Finite Difference Scheme

As stated above, the equation system for the determination of a_{ij} derived from one set of base polynomials is equivalent to that derived from another set of base polynomials. We will choose only one equation system obtained by the base polynomials x^k, $k=0,1,\cdots, N-1$ and prove that this equation system is the same as that given by the finite difference approach.

For the one dimensional case, supposing the whole domain has N grid points, x_1, x_2, $\cdots$, x_N. The (N-1)th order finite difference scheme for the first order spatial derivative can be written as a linear sum of the functional values at N grid points, which has the same form as (3.1) where the weighting coefficients are determined by the Taylor series expansion which is usually used in the design of the low order finite difference schemes. Using a Taylor series expansion, $u(x_j,t)$ can be expressed as

$$u(x_j,t) = u(x_i,t) + u^{(1)}(x_i,t)\cdot(x_j-x_i)+ \cdots +u^{(k)}(x_i,t)\cdot(x_j-x_i)^k/k!$$
$$+ \cdots + u^{(N-1)}(x_i,t)\cdot(x_j-x_i)^{N-1}/(N-1)! + R_N' \tag{3.34}$$

where $u^{(k)}(x_i,t)$ is the *k*th order derivative of u with respect to x at

x_i, R_N' is the truncated error, and can be written as

$$R_N' = u^{(N)}(\xi_i,t)\cdot(x_j-x_i)^N/N! \qquad , \ \xi_i \in [x_i,\ x_j]$$

Substituting (3.34) into (3.1) yields

$$u_x(x_i,t) = \sum_{j=1}^{N} a_{ij}\cdot\Big\{u(x_i,t) + u^{(1)}(x_i,t)\cdot(x_j-x_i) + \cdots + u^{(N-1)}(x_i,t)\cdot(x_j-x_i)^{N-1}/(N-1)! + R_N'\Big\} \tag{3.35}$$

In order to keep the right side of (3.35) consistent with the left side of (3.35) with (N-1)th order accuracy, we set

$$\begin{cases} \sum_{j=1}^{N} a_{ij} = 0 \\ \sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i) = 1 \\ \sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i)^k = 0 \quad , \qquad k = 2,\ 3,\ \cdots,\ N-1 \end{cases} \tag{3.36}$$

for $i = 1,\ 2,\ \cdots,\ N$.

Equation set (3.36) is another equation system for the determination of the weighting coefficients a_{ij} which are derived from the Taylor series expansion. It will now be proved that equation system (3.36) is the same as (3.2) which is derived from the high order polynomial approximation in the overall domain.

It is obvious that the first equation of (3.2) and (3.36) are the same, i.e.

$$\sum_{j=1}^{N} a_{ij} = 0 \tag{3.37}$$

Furthermore, it can be shown that the second equation of the two systems are the same, i.e.

$$\sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i) = \sum_{j=1}^{N} a_{ij}\cdot x_j - \Big(\sum_{j=1}^{N} a_{ij}\Big)\cdot x_i = \sum_{j=1}^{N} a_{ij}\cdot x_j = 1 \tag{3.38}$$

Now, assuming that the first p+1 equations of the two systems are the same, that is

$$\sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i)^k = \sum_{j=1}^{N} a_{ij}\cdot x_j^k - k\cdot x_i^{k-1} = 0 \qquad (3.39)$$

for $k = 0, 1, \cdots, p; \quad i = 1, 2, \cdots, N$

then using the binary formulation

$$(a-b)^p = a^p - c_p^1\cdot a^{p-1}\cdot b + \cdots + (-1)^k\cdot c_p^k\cdot a^{p-k}\cdot b^k + \cdots + (-1)^p\cdot b^p \qquad (3.40)$$

here c_p^k is the combination of p terms taken k at a time,

and setting a = b = 1, the following expression will be obtained.

$$c_p^1 - c_p^2 + \cdots + (-1)^{k+1}c_p^k + \cdots + (-1)^{p+1} = 1. \qquad (3.41)$$

Using (3.40), the (p+2)th equation of (3.36) can be written as

$$\sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i)^{p+1} = \sum_{j=1}^{N} a_{ij}\cdot x_j^{p+1} - c_{p+1}^1\cdot x_i\cdot[\sum_{j=1}^{N} a_{ij}\cdot(x_j^p -$$

$$\frac{1}{2}\cdot c_p^1\cdot x_j^{p-1}\cdot x_i + \cdots + (-1)^p\cdot x_i^{p-1}/(p+1))] \qquad (3.42)$$

Substituting (3.41), (3.39) into (3.42) leads to

$$\sum_{j=1}^{N} a_{ij}\cdot(x_j-x_i)^{p+1} = \sum_{j=1}^{N} a_{ij}\cdot x_j^{p+1} - c_{p+1}^1\cdot x_i\cdot[p\cdot x_i^{p-1} -$$

$$\frac{1}{2}\cdot c_p^1\cdot(p-1)\cdot x_i^{p-1} + \cdots + (-1)^{p-1}\cdot x_i^{p-1}]$$

$$= \sum_{j=1}^{N} a_{ij}\cdot x_j^{p+1} - (p+1)\cdot x_i^p\cdot[c_p^1 - c_p^2 + \cdots + (-1)^{k+1}c_p^k + \cdots + (-1)^{p+1}]$$

$$= \sum_{j=1}^{N} a_{ij}\cdot x_j^{p+1} - (p+1)\cdot x_i^p \qquad (3.43)$$

Equation set (3.43) demonstrates that the (p+2)th equation of the two systems are the same. Since p is an arbitrary integer only if $p \le N-2$, it has been proved that the two systems (3.36) and (3.2) are exactly the same.

Similarly, for the case of higher order derivatives, it is easy to show that the weighting coefficients $w_{ij}^{(m)}$ satisfy the following equation system, derived from the finite difference scheme for the mth order derivative in the overall domain

$$\begin{cases} \sum_{j=1}^{N} w_{ij}^{(m)} = 0 \\ \sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^m = m! \\ \sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^k = 0 \quad , \quad k = 1, 2, \cdots, N-1, \text{ but } k \neq m \end{cases} \tag{3.44}$$

for $i = 1, 2, \cdots, N;$ $m = 2, 3, \cdots, N-1$.

It is clear that the first equation of (3.44) is exactly the same as (3.19) for $m = 2, \cdots, N-1$. To prove that $w_{ij}^{(m)}$, for $2 \le m \le N-1$, satisfies other equations of (3.44), it is supposed that $w_{ij}^{(m-1)}$ satisfies those equations firstly, that is

$$\sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot (x_j - x_i)^k = \begin{bmatrix} (m-1)! & \text{when } k = m-1 \\ 0 & \text{others} \end{bmatrix} . \tag{3.45}$$

Using (3.18), now we have, for $1 \le k \le N-1$

$$\sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^k = m \cdot w_{ii}^{(m-1)} \cdot \sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^k + m \cdot \sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot (x_j - x_i)^{k-1} . \tag{3.46}$$

Substituting (3.45), (3.36), (3.19) into (3.46) leads to

$$\sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^k = \begin{bmatrix} m! & \text{when } k = m \\ 0 & \text{others} \end{bmatrix} \tag{3.47}$$

Since m is an arbitrary integer only if $2 \le m \le N-1$, it has been proved that $w_{ij}^{(m)}$ satisfies (3.44) exactly. Thus it can be concluded that GDQ is an extension of finite difference methods, and is a highest order finite difference scheme. It also provides a new way to develop high order finite difference schemes. It has been shown in Appendix B that the formulation of the first order derivative discretization obtained by GDQ in the interval $[x_{i-1}, x_{i+1}]$ is exactly the same as that given from the second order finite difference scheme.

3.3.6 Specific Results for Typical Distributions of Grid Points

In this section, three specific formulations of the weighting coefficients will be given for three typical distributions of the grid points: uniform grid; the coordinates chosen as the roots of $T_N(\eta)$ or $|T_N(\eta)| - 1$, where $T_N(\eta)$ is an *N*th order Chebyshev polynomial. Since the complete weighting coefficients of the second and higher order derivatives can be calculated from those of the first order derivative, and that for the multi-dimensional cases, each direction can be treated as in the **1D** case, then only the weighting coefficients of the first order derivative in the **1D** case are considered.

Case I: Uniform Grid

By a uniform grid it is meant that the grid has the same step sizes. Thus setting

$$\Delta x = x_2 - x_1 = x_i - x_{i-1} = x_N - x_{N-1}, \text{ etc.,}$$

one can obtain

$$x_j - x_i = (j-i)\Delta x$$

$$M^{(1)}(x_i) = (-1)^{N-i}\cdot(\Delta x)^{N-1}\cdot(i-1)!\cdot(N-i)! \quad , \quad i = 1, 2, \cdots, N$$

Thus

$$a_{ij} = (-1)^{i+j}\cdot\frac{(i-1)!\cdot(N-i)!}{\Delta x\cdot(i-j)\cdot(j-1)!\cdot(N-j)!} \tag{3.48a}$$

for i, j = 1, 2, ···, N, except j≠i

$$a_{ii} = -\sum_{j=1,j\neq i}^{N} a_{ij} \quad , \quad i = 1, 2, \cdots, N \tag{3.48b}$$

Case II: Coordinates Chosen As the Roots of $|T_N(\eta)| - 1$

An *N*th order Chebyshev polynomial can be written as

$$T_N(\eta) = \cos(N\theta) \tag{3.49}$$

with $\eta = \cos\theta$, $-1 \le \eta \le 1$

Setting $|T_N(\eta)| = 1$ yields

$N\theta = j\pi$, $j = 0, 1, \cdots, N$

i.e. $\eta_j = \cos(j\pi/N)$, $j = 0, 1, \cdots, N$

where η_j is the coordinate of the grid point in the domain [1, -1].

In this case, the Lagrange interpolated polynomial can be written as

$$r_j(\eta) = \frac{(-1)^{j+1}\cdot(1-\eta^2)\cdot T_N'(\eta)}{\bar{c}_j\cdot N^2\cdot(\eta-\eta_j)} \quad , \; j = 0, 1, \cdots, N \tag{3.50}$$

where $T_N'(\eta)$ is the first derivative of $T_N(\eta)$, and

$$\bar{c}_j = \begin{cases} 2 & \text{when } j = 0, N \\ 1 & \text{others} \end{cases} .$$

Thus (3.9) can be reduced to

$$a_{ij} = \frac{(-1)^{j+i}\cdot\bar{c}_i}{\bar{c}_j\cdot(\eta_i - \eta_j)} \quad , \quad i, j = 0, 1, \cdots, N, \text{ but } j\ne i \tag{3.51a}$$

$$a_{ii} = -\sum_{j=1, j\ne i}^{N} a_{ij} \quad , \quad i = 0, 1, \cdots, N \tag{3.51b}$$

It can be seen that (3.51) is the same as that deduced from the pseudo-spectral Chebyshev method (Ehrenstein et al, 1989). To analyse this behaviour, it is well known that both spectral methods and finite element methods are based on the principle of the weighted residuals. Spectral collocation methods can be considered as an extension of finite element methods. The difference is that the spectral collocation methods include only one element while finite element methods include many elements. As shown in Appendix B, finite difference methods can also be considered as "finite element" methods which are different from the standard approach in that the elements in a finite difference method are overlapped while the elements in a standard finite element method are patched. But if the whole computational domain is composed of only one element, both finite difference methods, and finite element methods in which the weighting function is taken as the delta function, should give the same results. This is because in this case, one overlapped element and one patched element are the same. From this analysis, it is shown that the GDQ approach should give the same results as the spectral collocation methods if the same distribution of grid points is used, since they can be considered as an extension of the finite difference and finite element methods with only one element. This phenomenon is confirmed in this research shown above.

If the physical domain is not [1, -1], but [a, b], then we need to use the following transformation

$x = 0.5\cdot(b-a)\cdot(1-\eta) + a$, where x is the physical coordinate

The weighting coefficients $\bar{a}_{ij}$ in the physical coordinate system can be written as

$$\bar{a}_{ij} = -2\cdot a_{ij}/(b-a) \quad , \quad i,\ j = 0,\ 1,\ \cdots,\ N. \tag{3.52}$$

Case III: Coordinates Chosen As the Roots of $T_N(\eta)$

Setting $T_N(\eta) = 0$ yields

$N\theta = 0.5\cdot(2j-1)\pi$, i.e. $\eta_j = \cos[0.5\cdot(2j-1)\pi/N]$, $j = 1,\cdots,N$

It should be noted that η_j is in the domain $[\eta_1,\ \eta_N]$, where η_1 =

$\cos(0.5\pi/N)$, $\eta_N = -\eta_1$, and $\eta_1 \neq 1$. In this case, the Lagrange interpolated polynomial can be written as

$$r_j(\eta) = \frac{(-1)^{j+1}\cdot(1-\eta_j^2)^{1/2}\cdot T_N(\eta)}{N\cdot(\eta - \eta_j)} \quad , \quad j = 1, \cdots, N. \tag{3.53}$$

Then (3.9) can be reduced to

$$a_{ij} = \frac{(-1)^{j+i}\cdot(1-\eta_j^2)^{1/2}}{(\eta_i-\eta_j)\cdot(1-\eta_i^2)^{1/2}} \ , \quad i, j = 1, \cdots, N, \text{ but } j\neq i \tag{3.54a}$$

$$a_{ii} = -\sum_{j=1, j\neq i}^{N} a_{ij} \quad , \quad i = 1, 2, \cdots, N. \tag{3.54b}$$

Similarly, if the physical domain is [a, b], using the following transformation

$$\eta = d_2\cdot(x-a)/(b-a) + d_1 \ , \text{ where } d_1 = \cos[0.5\pi/N], \ d_2 = -2\cdot d_1 \ ,$$

the weighting coefficients $\bar{a}_{ij}$ in the physical coordinate system can be written as

$$\bar{a}_{ij} = d_2\cdot a_{ij}/(b-a) = -2\cdot\cos[0.5\pi/N]\cdot a_{ij}/(b-a) \tag{3.55}$$

for i, j = 1, 2, $\cdots$, N.

3.4 Generalized Integral Quadrature (GIQ)

3.4.1 Introduction

In practice, for some problems such as the area of a surface and the volume of a body, it is necessary to know the integration of a function over some domain. In most cases, it is difficult to obtain the value of the integration analytically. As a result, numerical integration techniques are of interest in engineering. The numerical integration of function f(x) over domain [a, b] can usually be written in the form

$$\int_a^b f(x)\cdot dx = \sum_{i=1}^{N} c_i \cdot f(x_i) \tag{3.56}$$

where c_i is the weighted coefficient. There are a lot of conventional rules to determine c_i. The low order Simpson's rule is the most frequently used in obtaining approximate integrals. To obtain accurate results, high order methods are preferrable. Amongst them, the integration rules of Gaussian quadrature are extensively used, where both the weighting coefficients and the coordinates of the grid points are taken as unknowns which are determined by the 2N powers 1, x, $\cdots$, x^{2N-1}. The major advantage of Gaussian quadrature is its high accuracy. But for the general case, it has a disadvantage that the coordinates obtained may be outside the interest region. Thus it is necessary to check for the correct coordinates before numerical integration. Some specific Gaussian quadrature techniques such as the Gauss-Legendre can remove this drawback. Another drawback of Gaussian quadrature is that, since the coordinates are determined by the roots of some functions, they cannot be arbitrarily given. This may provide a difficulty in use for some problems where the functional values are only known at some specific grid points which may not coincide with the coordinates obtained, e.g. the experimental data are given at some selected points. In the case of the solution of a differential-integral equation, the grid points, where the unknowns are set, are usually generated in advance, and may not be the same as the coordinates of Gaussian quadrature. If this method was used for these cases, an interpolation approach needs to be employed. This may introduce additional errors in numerical computations.

If the grid points are given in advance, the weighted coefficients can be determined by the integration rule of interpolatory type. c_j can be written as

$$c_j = \int_a^b \frac{M(x)\cdot dx}{(x-x_j)\cdot M^{(1)}(x_j)} \quad , \quad j = 1, 2, \cdots, N \tag{3.57}$$

The exact expression of (3.57) is very complicated. Thus it is not easy to calculate c_i accurately on the computer using (3.57). As a special example, when the uniform grid is used, (3.57) is reduced to the Newton-Cotes integration formula which has some asymptotic expressions for c_i. For details, see, for example, the book of Davis et al (1975).

In some cases, the function f(x) is continuous in a whole domain containing sufficient grid points, and the integral over a part of the whole domain involving only a few grid points is of interest. When the conventional numerical integral schemes are used to approximate this integral, the results will be of low accuracy since these schemes use a linear combination of the functional values only in the integral domain to approximate the integral, thus only few functional values can be used. In order to improve the accuracy for this special integral, one may raise the question: is it possible to have a method which uses the functional values of the overall domain to approximate the integral over a part of the overall domain? The answer is positive, and will be shown in this section. The weighting coefficients of the integral for the case of given grid points are determined by the inversion of an matrix which is easily obtained on the computer.

3.4.2 One Dimensional Integrals with Specified Grid Points

It is supposed that a function f(x) is continuous in the overall domain [a, b], which can be decomposed into N-1 intervals with grid points as x_1 (=a), x_2, $\cdots$, x_N (=b). Since f(x) is continuous in the whole domain, it can be approximated by an (N-1)th order polynomial. In particular, when the functional values at N grid points are known, f(x) can be approximated by the Lagrange interpolated polynomial which are related to the functional values at all mesh points. As a result, the integral of this approximated polynomial over $[x_i, x_j]$ may involve the functional values outside the integral domain. As a general case, it is assumed that the integral of f(x) over a part of the whole domain can be approximated by a linear combination of the functional values in the overall domain with the form

$$\int_{x_i}^{x_j} f(x)\cdot dx = \sum_{k=1}^{N} c_k^{ij}\cdot f(x_k) \tag{3.58}$$

where x_i, x_j are numbers that can be altered. When x_i=a, x_j=b, (3.58) is a traditional integral. In a similar fashion to the analysis in the previous section, the (N-1)th order polynomial, which is an approximation to f(x), constitutes an N-dimensional linear vector space. Thus if all the base polynomials satisfy (3.58), so does any polynomial in the space. If the Lagrange interpolated polynomials, $r_n(x)$, n = 1, $\cdots$, N, are chosen as the base polynomials, c_k^{ij} can be determined by

$$c_k^{ij} = \int_{x_i}^{x_j} r_k(x)\cdot dx \tag{3.59}$$

The expression of c_k^{ij} is very complicated. Therefore, it is difficult to calculate c_k^{ij} accurately using (3.59). We will turn to another way to determine c_k^{ij}.

The GDQ formulation (3.1) can be written as a vector form

$$\mathbf{U}_x = A\mathbf{U} \tag{3.60}$$

where

$$\mathbf{U} = \left[u(x_1),\ u(x_2),\ \cdots,\ u(x_N)\right]^T$$

$$\mathbf{U}_x = \left[u_x(x_1),\ u_x(x_2),\ \cdots,\ u_x(x_N)\right]^T$$

and A is a matrix composed by a_{ij}. Equation (3.60) is exact when u(x) is a polynomial of degree less than or equal to N-1. Now, if we set

$$f(x) = \frac{du(x)}{dx} \tag{3.61}$$

or $$u(x) = \int_c^x f(x)\cdot dx + u(c) \tag{3.62}$$

where c is a constant,

$$\mathbf{f}^I = \left(\int_c^{X_1} f(x)\cdot dx,\ \int_c^{X_2} f(x)\cdot dx,\ \cdots,\ \int_c^{X_N} f(x)\cdot dx\right]^T \tag{3.63}$$

$$\mathbf{I} = (1,\ 1,\ \cdots\cdot,\ 1)^T \tag{3.64}$$

$$\mathbf{f} = \mathbf{U}_x \tag{3.65}$$

then (3.60) can be written as

$$\mathbf{f} = A\cdot(\mathbf{f}^I + u(c)\cdot\mathbf{I}). \tag{3.66}$$

Setting

$$W^I = A^{-1} \tag{3.67}$$

we obtain

$$\mathbf{f}^I = W^I\cdot\mathbf{f} - u(c)\cdot\mathbf{I} \tag{3.68}$$

The scalar form of (3.68) can be written as

$$\int_c^{X_i} f(x)\cdot dx = \sum_{k=1}^{N} w^I_{ik}\cdot f(x_k) - u(c) \tag{3.69}$$

for i = 1, 2, ⋯, N

Thus

$$\int_{X_i}^{X_j} f(x)\cdot dx = \sum_{k=1}^{N} (w^I_{jk} - w^I_{ik})\cdot f(x_k) \tag{3.70}$$

$$c_k^{ij} = w^I_{jk} - w^I_{ik} \tag{3.71}$$

It is found that the weighting coefficients w^I_{ik} determined by (3.67) are not accurate when u(x) is an (N-1)th order polynomial. From (3.61), it is clear that when u(x) is an (N-1)th order polynomial, f(x) is an (N-2)th order polynomial. Particularly, when the base polynomials for u(x) are taken as 1, x, $\cdots$, x^{N-1}, then the base polynomials for f(x) become 0, 1, $\cdots$, x^{N-2}. Obviously, u(x) = 1 and f(x) = 0 does not always satisfy (3.62). In other words, to hold (3.62) for all cases, $u(x) \neq 1$. But, since the determination of A involves the use of u(x) = 1, it can be concluded that (3.67) is not accurate for W^I. For the purpose of future comparison, we write

$$\tilde{W}^I = A^{-1} \tag{3.72}$$

In order to keep f(x) being a (N-1)th order polynomial and (3.62) held for all cases, u(x) should be an *N*th order polynomial without constant term. Thus u(x) has N terms with the form

$$u(x) = x\cdot(a_0 + a_1\cdot x + \cdots + a_{N-1}\cdot x^{N-1}) \tag{3.73}$$

It is clear, from (3.73), that u(x) constitutes an N dimensional linear vector space. One set of its base polynomials can be chosen as

$$g_k(x) = x\cdot r_k(x) \quad , \quad k = 1, 2, \cdots, N \tag{3.74}$$

where $r_k(x)$ is the Lagrange interpolated polynomial. Comparing with equation (3.1), we can set

$$u_x(x_i) = \sum_{j=1}^{N} \tilde{a}_{ij}\cdot u(x_j). \tag{3.75}$$

Using the same analysis as above, we obtain

$$W^I = (\tilde{A})^{-1} \tag{3.76}$$

where $\tilde{A}$ is a matrix composed of $\tilde{a}_{ij}$. We will discuss how to determine $\tilde{A}$ through two cases.

Case I: The Integral Domain not Including the Origin

It is supposed that the integral domain does not include the origin, i.e. $a > 0$ or $b < 0$. Substituting (3.74) into (3.75) yields

$$\tilde{a}_{ij} = \frac{x_i}{x_j} \cdot a_{ij} \quad , \quad \text{when } i \neq j \tag{3.77a}$$

$$\tilde{a}_{ii} = a_{ii} + 1/x_i \tag{3.77b}$$

for $i, j = 1, 2, \cdots, N$

(3.77) requires $x_i \neq 0$, for $i = 1, \cdots, N$. This is guaranteed by the condition of $a > 0$ or $b < 0$.

Case II: The Integral Domain Including the Origin

In this case, if all the grid points do not include the origin, then (3.77) can still be used, but if one grid point coincides with the origin, (3.77) is singular. This problem can be removed by the following transformation

$$\bar{x} = x + d \tag{3.78}$$

where $\bar{x}$ is the transformed coordinate, and d is a constant which guarantees that the transformed integral domain does not include the origin, that is, $\tilde{a} = a + d > 0$ or $\tilde{b} = b + d < 0$. (3.77) is held in the domain $[\tilde{a}, \tilde{b}]$. Using (3.78), we get $\tilde{a}_{ij}$, in this case, as

$$\tilde{a}_{ij} = \frac{x_i + d}{x_j + d} \cdot a_{ij} \quad , \quad \text{when } i \neq j \tag{3.79a}$$

$$\tilde{a}_{ii} = a_{ii} + 1/(x_i + d) \tag{3.79b}$$

for $i, j = 1, 2, \cdots, N$.

3.4.3 Multi-Dimensional Integrals With Specified Grid Points

Firstly, we consider the two-dimensional case. The whole domain is $x \in [a, b]$, $y \in [c, d]$ which can be decomposed into $(N-1)\times(M-1)$ intervals with N grid points in the x direction x_1 (=a), x_2, ···, x_N (=b), and M grid points in the y direction y_1 (=c), y_2, ···, y_M (=d). We will study the integral over a part of the whole domain. The shaded area as shown in Fig. 3.1 is taken as the integral domain.

If the function f(x,y) is continuous in the overall domain, using the same technique as in the **1D** case and in the GDQ analysis, we can use the functional values in the overall domain to approximate the integral over a part of the whole domain. As a result,

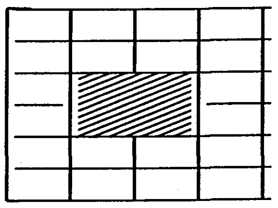


Fig. 3.1

the formulation of the **2D** integral of f(x,y) over the domain $x \in [x_i, x_j]$, $y \in [y_l, y_m]$ can be written as

$$\int_{x_i}^{x_j}\int_{y_l}^{y_m} f(x,y)\cdot dx\cdot dy = \sum_{n=1}^{N}\sum_{k=1}^{M}(w^I_{jn} - w^I_{in})\cdot(\bar{w}^I_{mk} - \bar{w}^I_{lk})\cdot f(x_n,y_k) \qquad (3.80)$$

where w^I_{ij}, $\bar{w}^I_{ij}$ are the weighting coefficients of the one-dimensional integral

$$w^I_{jn} - w^I_{in} = \int_{x_i}^{x_j} r_n(x)\cdot dx \quad , \ i,j,n=1,2,\cdots,N \qquad (3.81)$$

$$\bar{w}^I_{jk} - \bar{w}^I_{ik} = \int_{y_i}^{y_j} s_k(y)\cdot dy \quad , \ i,j,k=1,2,\cdots,M \qquad (3.82)$$

which can be determined by the inversion of the matrix as stated previously. Here $r_n(x)$, $s_k(y)$ are the Lagrange interpolated polynomials respectively. For the three-dimensional integral, it is supposed that the function f(x,y,z) is continuous in the overall domain $x \in [a, b]$, $y \in [c, d]$, $z \in [g, h]$ which can be divided into $(N-1)\times(M-1)\times(L-1)$ intervals with N grid points in the x direction x_1 (=a), x_2, $\cdots$, x_N (=b), M grid points in the y direction y_1 (=c), y_2, $\cdots$, y_M (=d) and L grid points in the z direction z_1 (=g), z_2, $\cdots$, z_L (=h). It is shown that the integral of f(x,y,z) over any interval in the domain can be approximated by the combination of all the functional values in the whole domain. The GIQ formulation of this case can be written as

$$\int_{x_i}^{x_j}\int_{y_l}^{y_m}\int_{z_e}^{z_q} f(x,y,z)\cdot dx\cdot dy\cdot dz =$$
$$\sum_{n=1}^{N}\sum_{k=1}^{M}\sum_{p=1}^{L}(w^I_{jn} - w^I_{in})\cdot(\bar{w}^I_{mk} - \bar{w}^I_{lk})\cdot(\bar{\bar{w}}^I_{qp} - \bar{\bar{w}}^I_{ep})\cdot f(x_n,y_k,z_p) \tag{3.83}$$

where $\bar{\bar{w}}^I_{qp}$ is the weighting coefficient of the **1D** integral in the z direction, which can be obtained by the inversion of a matrix.

$$\bar{\bar{w}}^I_{qp} - \bar{\bar{w}}^I_{ep} = \int_{z_e}^{z_q} t_p(z)\cdot dz \quad , \ e,q,p=1,2,\cdots,L \tag{3.84}$$

where $t_p(z)$ is the Lagrange interpolated polynomial in the z direction.

In all cases discussed in this subsection, if the domain in a particular direction includes a point of origin, the weighting coefficients of the integral in that direction can be obtained using the same way as in the **1D** case.

3.5 Concluding Remarks

Based on the analysis of the high order polynomial approximation in the overall domain, and the analysis of a linear vector space, the generalized differential and integral quadrature techniques, which are global methods, were developed. For the case of GDQ, the weighting coefficients of the first order derivative were determined by a simple algebraic formulation without any restriction on choice of grid points. Furthermore, a recurrence relationship for the determination of the weighting coefficients of the second and higher order derivatives was developed. For the multi-dimensional case, each direction can be treated using the same approach as in the **1D** case. It has been shown that GDQ can be considered as the highest order finite difference scheme, and when the coordinates of grid points are chosen as the roots of a Chebyshev polynomial, the formulation of the first order derivative discretization obtained by GDQ is exactly the same as that given by the Chebyshev Pseudospectral method. For the case of GIQ, if the function is continuous in the whole domain, then the integral of the function over a part of the whole domain (including the case of a whole domain) can be approximated by a linear combination of all the functional values in the overall domain. The weighting coefficients of the integral can be determined from those of the first order derivative discretization. The multi-dimensional integrals can be approximated in the same way as in the **1D** case.

CHAPTER FOUR

ERROR, STABILITY ANALYSIS AND MODEL PROBLEMS

4.1 Introduction

The theory and details of GDQ and GIQ have been described in the previous chapter. In this chapter, the basic properties of these schemes will be analysed. The errors of the approximations for the derivatives and integrals will be estimated in section 4.2. For the stability analysis, it is desirable to take into account the influence of the types and numerical treatments of the boundary conditions on the overall stability of the scheme. This can be done by matrix methods. This approach has an advantage over the Von Neumann method, which can only consider the influence of the periodic boundary conditions, in that the influence of the different types of the boundary conditions can be easily taken into account. In section 4.3, we study the stability condition using matrix methods, firstly for the semi-discrete equations obtained by the spatial discretizations, then for the time discretized equations. Since all the stability conditions are related to the eigenvalues of the spatial discretization matrix, it is valuable to know the properties of these eigenvalues. The eigenvalues of the specific matrices given from GDQ are given in section 4.5, where the influence of the different types of the boundary conditions and the distributions of the grid points are discussed. Section 4.6 shows the application of GDQ and GIQ to model problems. Some comparisons with exact solutions and numerical solutions given from finite difference schemes are also included in this section.

Finally, some concluding remarks are given in section 4.7.

4.2 Error Estimations

4.2.1 The Function Approximation

It is interesting to analyse the errors resulting from the approximation of the function, derivatives and integrals. For the sake of simplicity, the cases of derivatives with respect to x and from one-dimensional integrals are only considered.

Firstly, we will discuss the approximation error when f(x) is approximated by an (N-1)th order polynomial, particularly by the Lagrange interpolation polynomial.

$$P_N f = \sum_{i=1}^{N} f(x_i)\cdot r_i(x) \tag{4.1}$$

We define the approximation error of f(x) as

$$E(f) = f(x) - P_N f\ . \tag{4.2}$$

If it is supposed that the *N*th order derivative of function f(x) is a constant, say K, then using a Taylor expansion, we can obtain

$$\begin{aligned} f(x) &= f(c) + f^{(1)}(c)\cdot(x-c) + \cdots + f^{(k)}(c)\cdot(x-c)^k/k! + \cdots \\ &\quad + f^{(N-1)}(c)\cdot(x-c)^{N-1}/(N-1)! + f^{(n)}(\xi)\cdot(x-c)^N/N! \\ &= m_0 + m_1 x + m_2 x^2 + \cdots + m_{N-1} x^{N-1} + K\cdot x^N/N! \end{aligned} \tag{4.3}$$

where c is a constant, and $\xi \in [x, c]$. Since (4.1) is exactly satisfied for a polynomial of degree less than or equal to N-1, we have

$$E(x^k) = 0, \quad \text{when } k = 0, 1, \cdots, N-1. \tag{4.4}$$

Substituting (4.3) into (4.2) and using (4.4), we obtain

$$E(f) = K\cdot E(x^N)/N! \tag{4.5}$$

where

$$E(x^N) = x^N - \sum_{i=1}^{N} x_i^N \cdot r_i(x) \ . \qquad (4.6)$$

On the other hand, substituting the (N-1)th order polynomial $g(x) = x^N - (x-x_1)\cdot(x-x_2)\cdots(x-x_N) = x^N - M(x)$ into (4.1), we obtain

$$\sum_{i=1}^{N} x_i^N \cdot r_i(x) = x^N - M(x) \qquad (4.7)$$

Finally, we get

$$E(f) = K\cdot M(x)/N! \qquad (4.8)$$

In most cases, the *N*th order derivative of f(x) is not a constant, but may be bounded. In this case, we can turn to another way to analyse E(f). For simplicity, we set $\phi(x) = P_N f$, and define the function F(z) as

$$F(z) = f(z) - \phi(z) - a\cdot M(z) \qquad (4.9)$$

Clearly, when $z = x_1, x_2, \cdots, x_N$, $F(z) = 0$. If we set $F(x) = 0$, we get

$$E(f) = f(x) - P_N f = f(x) - \phi(x) = a\cdot M(x) \qquad (4.10)$$

Since F(z) has N+1 roots in the domain, by repeated application of Rolle's theorem, the *N*th order derivative of F(z), $F^{(N)}(z)$, is found to have at least one root lying between x_1 and x_N. Let ξ denote this point. We have

$$F^{(N)}(\xi) = 0 \ . \qquad (4.11)$$

From (4.9), we obtain

$$a = f^{(N)}(\xi)/N! \ , \qquad (4.12)$$

so, $E(f) = f^{(N)}(\xi)\cdot M(x)/N! \ .$ (4.13)

Generally, ξ is a function of x.

4.2.2 The Derivative Approximation

We define the error for the *m*th order derivative approximation as

$$E_D^{(m)}(f) = \frac{\partial^m f}{\partial x^m} - \frac{\partial^m (P_N f)}{\partial x^m} = \frac{\partial^m f}{\partial x^m} - \frac{\partial^m \phi}{\partial x^m} \tag{4.14}$$

where m = 1, 2, ···, N-1. Generally, $E_D^{(m)}(f)$ can be written as

$$E_D^{(m)}(f) = \frac{1}{N!}\cdot\partial^m[f^{(N)}(\xi)\cdot M(x)]/\partial x^m \tag{4.15}$$

Since ξ is an unknown function of x, it is difficult to estimate $E_D^{(m)}(f)$ using (4.15). As a special case, if we assume that the *N*th order derivative of f(x) is a constant, namely K, then from (4.8), we get

$$E_D^{(m)}(f) = K\cdot M^{(m)}(x)/N! \tag{4.16}$$

Although (4.16) is satisfied for the condition of $f^{(N)}(\xi) = K$, it is useful in the error analysis. Firstly, (4.16) has no restriction on x, in other words, x can be any coordinate in the domain. Secondly, similar to the analysis of the order of the truncated error in a low order finite difference scheme, when the order of the truncated error caused by GDQ is studied, we can only consider the (N+1)th term in the Taylor series expansion though this term is not the exact error. The (N+1)th term of the Taylor series expansion is $f^{(N)}(c)\cdot(x-c)^N/N!$, where c is a constant. So, $f^{(N)}(c)$ can be treated as a constant in this case. Thus the analysis of the function and the derivative approximations is the same as that shown above. For a more general case, we can use a similar method as in the analysis of the function approximation to do it. Since $g(z) = f(z) - \phi(z)$ has N roots in the domain, according to Rolle's theorem, its *m*th order derivative $g^{(m)}(z)$ has at least N-m roots in the domain, namely, $\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_{N-m}$. Thus the function

$$F^{(m)}(z) = g^{(m)}(z) - a\cdot\bar{M}(z) = f^{(m)}(z) - \phi^{(m)}(z) - a\cdot\bar{M}(z) \tag{4.17}$$

where

$$\bar{M}(z) = (z-\bar{x}_1)\cdot(z-\bar{x}_2)\cdots(z-\bar{x}_{N-m}) ,$$

vanishes at $\bar{x}_1$, $\bar{x}_2$, $\cdots$, $\bar{x}_{N-m}$. Now, if we set $F^{(m)}(\bar{x}) = 0$, where $\bar{x}$ is different from $\bar{x}_1$, $\bar{x}_2$, $\cdots$, $\bar{x}_{N-m}$, then $F^{(m)}(z)$ has N-m+1 roots, and

$$E_D^{(m)}(f(\bar{x})) = f^{(m)}(\bar{x}) - \phi^{(m)}(\bar{x}) = a\cdot\bar{M}(\bar{x}) \tag{4.18}$$

Using Rolle's theorem repeatedly (N-m times), the (N-m)th order derivative of $F^{(m)}(z)$ is found to have at least one root ξ, i.e.

$$f^{(N)}(\xi) - a\cdot(N-m)! = 0,$$

so, $a = f^{(N)}(\xi)/(N-m)!$

and

$$E_D^{(m)}(f(\bar{x})) = f^{(N)}(\xi)\cdot\bar{M}(\bar{x})/(N-m)! \tag{4.19}$$

Equation (4.19) is satisfied for $\bar{x} \neq \bar{x}_1$, $\bar{x}_2$, $\cdots$, $\bar{x}_{N-m}$. This is guaranteed if $\bar{x}$ is outside the domain of x_1, x_2, $\cdots$, x_N.

If it is assumed that all the x_i and x are in the interval h, and the *N*th order derivative of function f is bounded, then

$|f^{(N)}(\xi)| \le C$, where C is a positive constant

$|M^{(m)}(x)| \le N\cdot(N-1)\cdots(N-m+1)\cdot h^{N-m}$

$|\bar{M}(\bar{x})| \le h^{N-m}$

so

$$|E_D^{(m)}(f)| \le \frac{C\cdot h^{N-m}}{(N-m)!} \tag{4.20}$$

for $1 \le m \le N-1$

4.2.3 The Integral Approximation

The error of the numerical integral of f(x) in the domain $[x_i, x_j]$ is defined as

$$E_I(f,\ x_i,\ x_j) = \int_{x_i}^{x_j} f(x)\cdot dx - \int_{x_i}^{x_j} \phi(x)\cdot dx$$

$$= \int_{x_i}^{x_j} f(x)\cdot dx - \sum_{k=1}^{N} (w_{jk}^{I} - w_{ik}^{I})\cdot f(x_k) \tag{4.21}$$

where w_{ij}^{I} is the weighting coefficient of the integral described in the previous chapter. For a general case, using (4.13), we get

$$E_I(f,\ x_i,\ x_j) = \frac{1}{N!}\cdot\int_{x_i}^{x_j} f^{(N)}(\xi)\cdot M(x)\cdot dx \quad . \tag{4.22}$$

If the integral domain is $[x_i,\ x_{i+1}]$, then M(x) does not change its sign in $[x_i,\ x_{i+1}]$. By using the second mean-value theorem, (4.22) can be reduced to

$$E_I(f,\ x_i,\ x_{i+1}) = \frac{f^{(N)}(\eta)}{N!}\cdot\int_{x_i}^{x_{i+1}}\cdot M(x)\cdot dx \tag{4.23}$$

Generally, M(x) may change its sign in the domain $[x_i,\ x_j]$, but $|M(x)|$ is always positive in the domain. If it is assumed that $|f^{(N)}(\xi)| \le C$, then (4.22), (4.23) can be rewritten as

$$|E_I(f,\ x_i,\ x_j)| \le \frac{C}{N!}\cdot\int_{x_i}^{x_j} |M(x)|\cdot dx \tag{4.24}$$

$$|E_I(f,\ x_i,\ x_{i+1})| \le \frac{C}{N!}\cdot\left|\int_{x_i}^{x_{i+1}}\cdot M(x)\cdot dx\right| \tag{4.25}$$

4.3 Stability Analysis

Time-dependent problems are usually well-posed by the equation

$$\frac{\partial u}{\partial t} = L(u) \tag{4.26}$$

with proper initial and boundary conditions, where L is an operator which contains the spatial part of the partial differential equations. L is generally a non-linear operator. After discretization by GDQ and linearization of the non-linear terms, (4.26) can be transformed into a set of ordinary differential equations in time

$$\frac{d\mathbf{U}}{dt} = A\mathbf{U} + \mathbf{Q} \tag{4.27}$$

where **U** is the vector of the functional values at interior grid points, **Q** contains non-homogeneous and boundary values and A is a matrix. The stability condition of (4.27) is the same as that for the spatial discretization, which will be discussed in the following subsection, since there is no time discretization in (4.27).

4.3.1 Spatial Discretization

The stability analysis of (4.27) is based on the eigenvalue structure of the matrix A, since the exact solution of (4.27) is directly determined by the eigenvalues and eigenvectors of A. Let λ_i, i=1, $\cdots$, N be the eigenvalues of A, $\mathbf{V}_i$ the associated eigenvectors, and the matrix P formed by the N columns $\mathbf{V}_i$, the diagonal matrix D formed by the eigenvalues. We then get

$$D = P^{-1}AP \ . \tag{4.28}$$

Since the eigenvectors $\mathbf{V}_i$ form a complete set of base vectors in the considered space, the exact solution of (4.27) and vector **Q** can be written as a linear combination of $\mathbf{V}_i$, thus

$$\mathbf{U} = P\bar{\mathbf{U}} \tag{4.29}$$

$$\mathbf{Q} = P\bar{\mathbf{Q}} \tag{4.30}$$

where $\bar{\mathbf{U}}$, $\bar{\mathbf{Q}}$ are vectors. Substituting (4.29), (4.30) into (4.27), we get

$$\frac{d\bar{u}_i}{dt} = \lambda_i\bar{u}_i + \bar{q}_i \tag{4.31}$$

where the $\bar{u}_i$, $\bar{q}_i$ are the components of $\bar{\mathbf{U}}$ and $\bar{\mathbf{Q}}$. The solution of (4.31) can be written as

$$\bar{u}_i = [\bar{u}_i(0) + \bar{q}_i/\lambda_i]\cdot e^{\lambda_i t} - \bar{q}_i/\lambda_i \ . \tag{4.32}$$

Using (4.29) and (4.32), the exact solution of (4.27) is

$$\mathbf{U} = \sum_{i=1}^{N}\left[\bar{u}_i(0)\cdot e^{\lambda_i t} + \frac{\bar{q}_i}{\lambda_i}\cdot(e^{\lambda_i t} - 1)\right]\cdot\mathbf{V}_i \ . \tag{4.33}$$

To keep the solution **U** bounded requires

$$\mathrm{Re}\ (\lambda_i) \le 0 \qquad \text{for all } i \tag{4.34}$$

where Re (λ_i) means the real part of λ_i.

Since the error between the exact solution of (4.27) and the numerical solution of (4.27) always satisfies the homogeneous equation, therefore, in terms of (4.33), the error (vector) at time level $t=n\Delta t$ can be written as

$$\mathbf{E}(n\Delta t) = \sum_{i=1}^{N} \bar{e}_i(0)\cdot e^{\lambda_i n\Delta t}\cdot\mathbf{V}_i \tag{4.35}$$

where $\bar{e}_i(0)$ is the component of the initial error vector $\bar{\mathbf{E}}(0)=P^{-1}\mathbf{E}(0)$.

By defining the amplification factor G as

$$\mathbf{E}(n\Delta t) = G\cdot\mathbf{E}((n-1)\Delta t)$$

we get from (4.35)

$$G = e^{D\Delta t} \ . \tag{4.36}$$

Here G is a diagonal matrix. Equation (4.34) gurantees $|G_i| \le 1$, which means that the error will not be amplified. Equation (4.34) is, therefore, the stability condition for the spatial discretization.

4.3.2 Time Discretization

Since the error between the exact solution and the numerical solution of a full discrete equation always satisfies the homogeneous equation, we will only investigate the stability behaviour of the homogeneous equation. When the Euler explicit scheme is applied to (4.27) with **Q** =

0, we obtain

$$U^{n+1} = (I + \Delta t\cdot A)\cdot U^n = C\cdot U^n = C^{n+1}\cdot U^0 \qquad (4.37)$$

where I is a unit matrix, n means the time level and C is a matrix. The necessary stability condition for (4.37) is

$$\rho(C) \le 1 + O(\Delta t) \qquad (4.38)$$

where $\rho(C)$ is the spectral radius of the matrix C. (4.38) guarantees the solution of (4.37) to be bounded for a finite value of time. In practical application, the stability condition

$$\rho(C) \le 1 \qquad (4.39)$$

is recommended because it makes (4.37) always stable. Since $\mu(C) = 1 + \Delta t\cdot\mu(A)$, where $\mu(C)$, $\mu(A)$ means the eigenvalues of the matrices C and A, equation (4.39) gives

$$|1 + \Delta t\cdot\lambda_i| \le 1 \quad , \ i = 1, 2, \cdots, N \qquad (4.40)$$

where λ_i is the eigenvalue of A.

Equation (4.40) shows, clearly, that all the eigenvalues of A multiplied by the time step size should be within a stable region (a unit circle). Thus for a general case of the stability of the time integration, we can only consider the behaviour of the scalar model equation

$$\frac{dw}{dt} = \lambda\cdot w \qquad (4.41)$$

where λ can be one of the eigenvalues of the spatial discretization matrix. A general multi-step method of order K applied to (4.41) can be written as

$$\sum_{k=0}^{K} \alpha_k\cdot w^{n+k} = \Delta t\cdot \sum_{k=0}^{K} \beta_k\cdot\lambda\cdot w^{n+k} \qquad (4.42)$$

with consistency conditions

$$\sum_{k=0}^{K} \alpha_k = 0 \ , \quad \sum_{k=0}^{K} k\cdot\alpha_k = \sum_{k=0}^{K} \beta_k \ . \qquad (4.43)$$

Introducing the time shifted operator E as

$$w^{n+k} = E^k \cdot w^n \tag{4.44}$$

then (4.42) can be written as

$$P(E) \cdot w^n = 0 \tag{4.45}$$

where $P(E) = \sum_{k=0}^{K} (\alpha_k - \Delta t \cdot \beta_k \cdot \lambda) \cdot E^k$.

It is shown that, the stability condition for (4.42) is to keep all the roots of the characteristic polynomial

$$P(z) = 0 \tag{4.46}$$

being of modulus lower than or equal to one, i.e.

$$|z_k| \le 1, \quad k = 1, 2, \cdots, K . \tag{4.47}$$

When the time integration scheme is specified, α_k, β_k are known numbers. As a result, z_k is the function of $\Delta t \cdot \lambda$. If the Euler explicit scheme is chosen, then K = 1, and $z = 1 + \Delta t \cdot \lambda$. For this

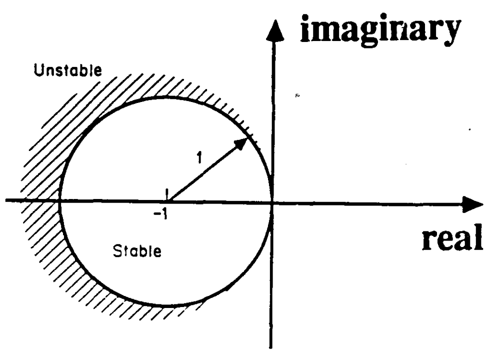


Fig. 4.1 Euler explicit scheme

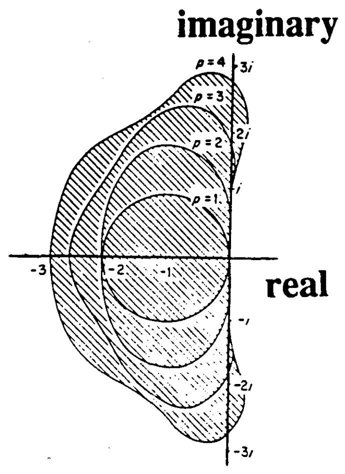


Fig. 4.2 Runge-Kutta scheme

case, (4.47) is exactly the same as (4.40). Fig. 4.1 shows the stability region of the Euler explicit scheme. The stability region of the explicit Runge-Kutta schemes is displayed in Fig. 4.2. Among the Runge-Kutta schemes, the 4-stage scheme is favourable because its high order accuracy in time is consistent with the high order accuracy of GDQ. To reduce the storage required, the standard 4-stage Runge-Kutta scheme for the equation

$$\frac{dw}{dt} = f(w) \tag{4.48}$$

can be revised as (Pike and Roe, 1985)

$$\begin{cases} w = w^n \\ g = f(w) \end{cases} \tag{4.49}$$

$$\begin{cases} w = w^n + 0.25\cdot\Delta t\cdot g \\ g = f(w) \end{cases} \tag{4.50}$$

$$\begin{cases} w = w^n + \Delta t\cdot g/3 \\ g = f(w) \end{cases} \tag{4.51}$$

$$\begin{cases} w = w^n + 0.5\cdot\Delta t\cdot g \\ g = f(w) \end{cases} \tag{4.52}$$

$$w^{n+1} = w^n + \Delta t\cdot g \tag{4.53}$$

4.4 Convergence

According to the equivalence theorem of Lax, (for details, see the book of Richtmyer and Morton (1967)), it has been shown that, for a well-posed initial value problem and a consistent discretization scheme, stability is the necessary and sufficient condition for convergence. This is also true in GDQ discretization because it is consistent. From section 4.2, the order of the truncated error for the *m*th order derivative discretization by GDQ can be written as

$$R^{(m)}_{GDQ} = O[C\cdot h^{N-m}/(N-m)!] \tag{4.54}$$

where O[a] means that its value is the same order as a and m indicates the order of the derivative. For a given m and h, there exists a finite integer L, which has

$$L + 1 > h\ , \quad \text{i.e.} \quad \frac{h}{L+1} < 1 \tag{4.55}$$

Now, we see

$$\frac{C\cdot h^{N-m}}{(N-m)!} = \frac{C\cdot h^{L}}{L!}\cdot \frac{h^{N-m-L}}{(N-m)\cdot(N-m-1)\cdots(L+1)}$$

$$\le \frac{C\cdot h^{L}}{L!}\cdot\left(\frac{h}{L+1}\right)^{N-m-L} \qquad (4.56)$$

Since C, h, L, m are the finite numbers, we have

$$\frac{C\cdot h^{N-m}}{(N-m)!} \longrightarrow 0 \quad , \text{ when } N \longrightarrow \infty \qquad (4.57)$$

So

$$R^{(m)}_{GDQ} \longrightarrow 0 \quad , \text{ when } N \longrightarrow \infty \qquad (4.58)$$

For the general case, the differential operator L may include different orders of spatial derivatives. Its truncated error R_{GDQ} caused by GDQ may be the combination of $R^{(m)}_{GDQ}$. Since every $R^{(m)}_{GDQ}$ tends to zero when N tends to infinity, so, $R_{GDQ} \to 0$, when $N \to \infty$. In other words, GDQ discretization is consistent.

4.5 Eigenvalues of Specific Matrices

From previous analysis, all the stability conditions are related to the eigenvalues of the spatial discretization matrix. We will investigate, in this section, the eigenvalues of some typical matrices obtained by GDQ discretization for model problems, and the influence of the boundary conditions and grid distributions on them.

4.5.1 The Convection Operator

Here $L(u)$ is chosen as a convection operator

$$L(u) = -\frac{\partial u}{\partial x} \qquad \text{on } [0,\ 1] \qquad (4.59)$$

with Dirichlet Boundary condition

$$u(0) = f(x) \qquad (4.60)$$

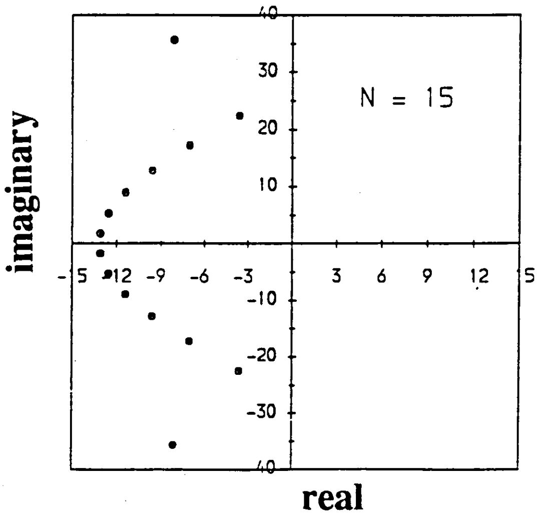
Firstly, we consider the three typical distributions of the grid points given in section 3.6 of previous chapter to study the influence of the grid points.

Case I : basic grid is generated by $|T_N(\eta)| = 1$

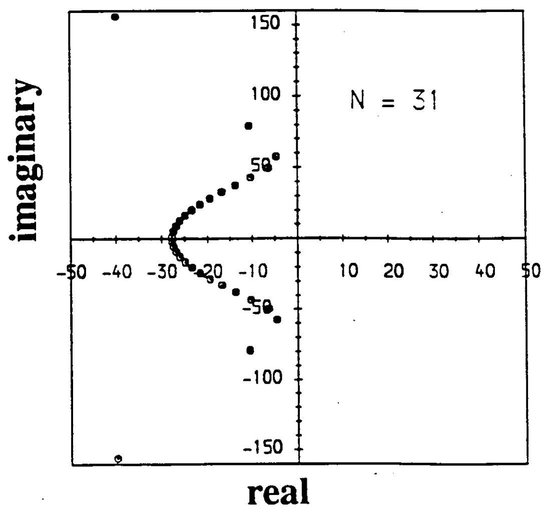
Case II : basic grid is generated by $T_N(\eta) = 0$

Case III : uniform grid

The eigenvalues with grid of case I are plotted in Fig. 4.3. Fig. 4.4 and Fig. 4.5 show the eigenvalues with case II and case III grid respectively. It is clear, from Fig. 4.3, that the real parts of all the eigenvalues of case I are strictly negative. This is not true for cases II and III. In fact, the real part of the maximum eigenvalue for cases II and III is positive although the modulus of the maximum eigenvalue of these two cases is less than that of case I. It is noted that, for cases II and III, the maximum eigenvalue does always lie in the unstable region. This behaviour is independent of the number of grid points used. Thus it seems to be true that the distribution of the grid points can greatly influence the stability behaviour of a global method such as

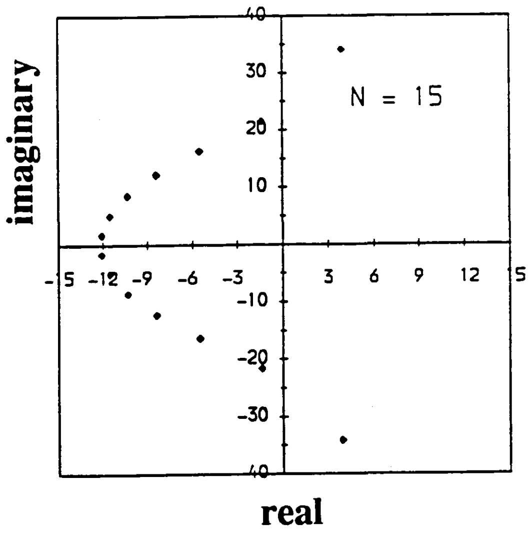


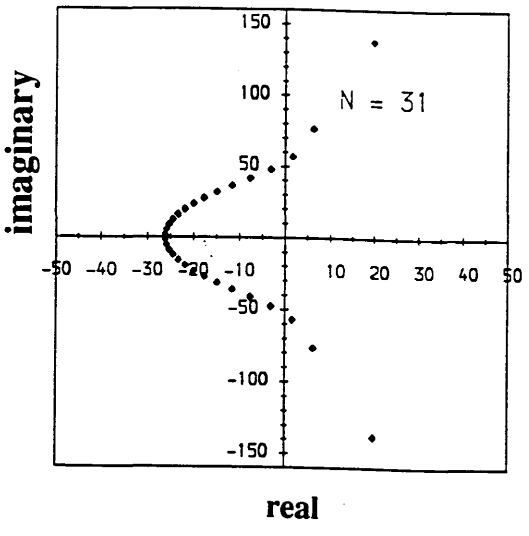
(a) $|\lambda|_{max} = 36.5$, $|\Delta x|_{min} = 0.0125$



(b) $|\lambda|_{max} = 161.1$, $|\Delta x|_{min} = 0.00274$

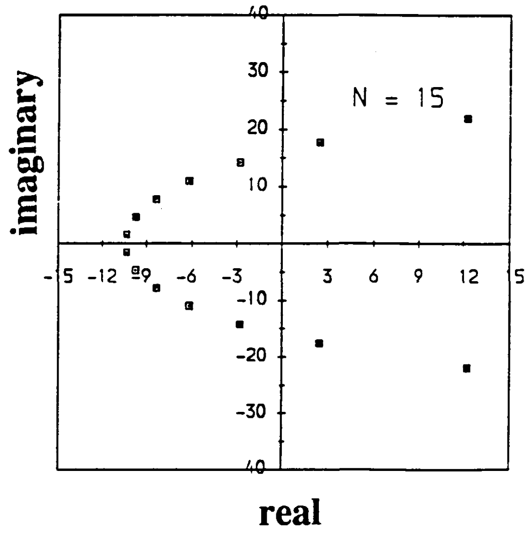
Fig.4.3 Convection Operator Eigenvalues with Grid of Case I

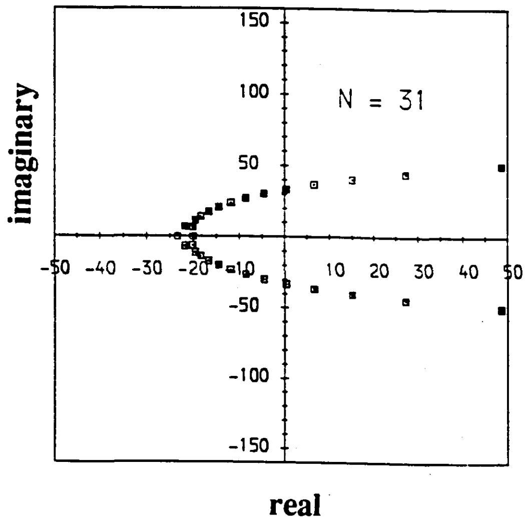




(a) $|\lambda|_{max} = 34.4$, $|\Delta x|_{min} = 0.0219$ (b) $|\lambda|_{max} = 139.4$, $|\Delta x|_{min} = 0.00513$

Fig. 4.4 Convection Operator Eigenvalues with Grid of Case II





(a) $|\lambda|_{max} = 25.1$, $|\Delta x|_{min} = 0.0714$ (b) $|\lambda|_{max} = 69.89$, $|\Delta x|_{min} = 0.03333$

Fig. 4.5 Convection Operator Eigenvalues with Grid of Case III

GDQ. It is also found that the minimum step size near the boundary, for cases II and III, is larger than that for case I.

Here, we pose a question: is the above observation likely to be a major reason to cause stability problems through the use of cases II and III? To study this, we introduce a transformation

$$\bar{x} = (1 - \alpha)\cdot(3\cdot x^2 - 2\cdot x^3) + \alpha\cdot x \quad , \quad \alpha \geq 0 \qquad (4.61)$$

where $\bar{x}$ is the transformed coordinate. When $\alpha \leq 1$, the transformed grid is stretched near the boundary (i.e. grid points are more concentrated near the boundary), when $\alpha > 1$, the transformed grid is relaxed near the boundary. Using (4.61), we can get

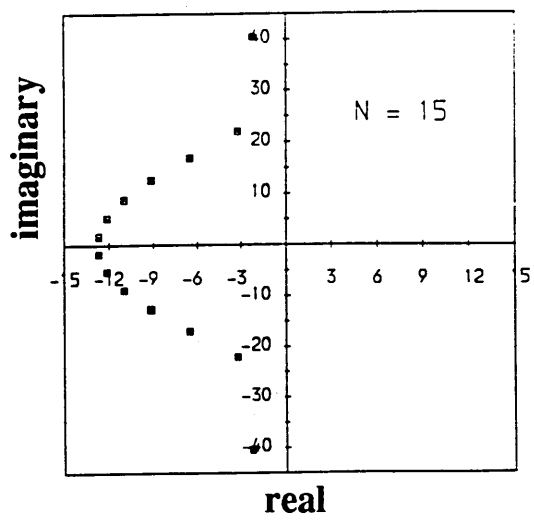
Case IV : Transformed from Case II with $\alpha < 1$

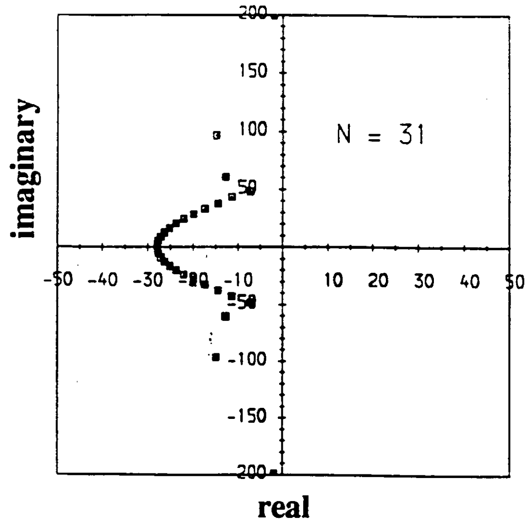
Case V : Transformed from Case III with $\alpha < 1$

To investigate the effect of the minimum step size, under the condition of stability, on the value of the modulus of the maximum eigenvalue, we introduce

Case VI : Transformed from Case I with $\alpha > 1$

Fig. 4.6 - 4.8 display the eigenvalues of cases IV, V and VI. In these cases, the real parts of all the eigenvalues are strictly negative. It is shown in Fig. 4.8 that when the minimum step size is relaxed near the boundary, the value of the maximum eigenvalue is reduced, thus the time step size is relaxed. It can be concluded from Fig. 4.6 - 4.7 that the stretched grid near the boundary can improve the stability. From here, one may have a question : what is the behaviour if the grid is stretched at other points. To study it, we introduce another transformation, which stretches the grid near the middle point





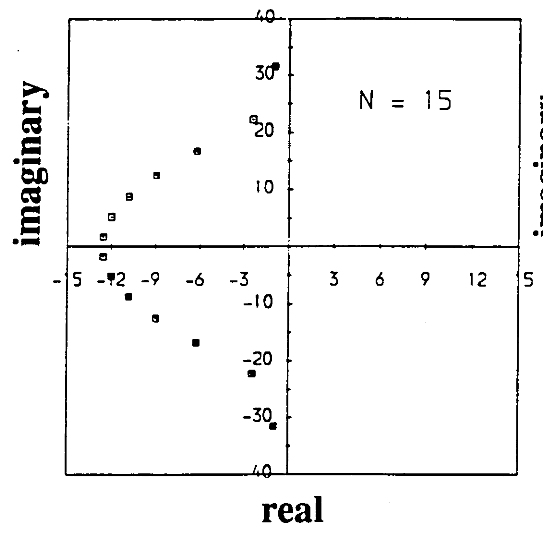
(a) $|\lambda|_{max} = 40.5$, $|\Delta x|_{min} = 0.0161$

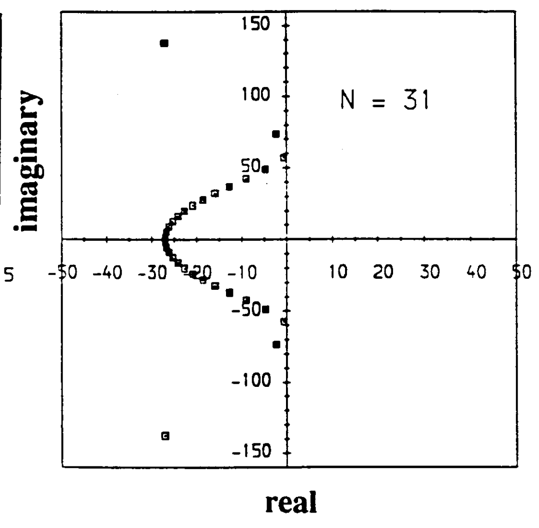
$\alpha = 0.71$

(b) $|\lambda|_{max} = 199.1$, $|\Delta x|_{min} = 0.00336$

$\alpha = 0.65$

Fig.4.6 Convection Operator Eigenvalues with Grid of Case IV





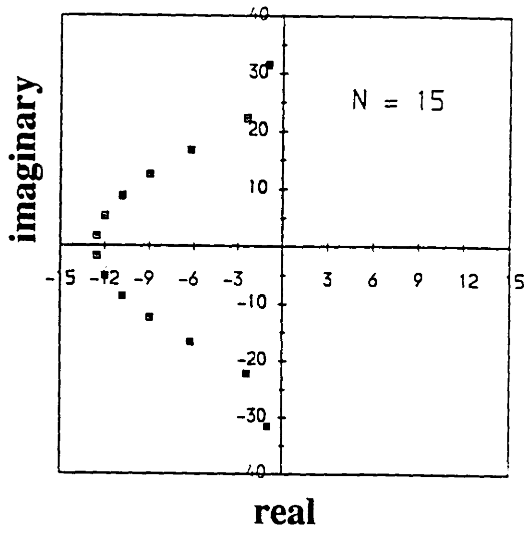
(a) $|\lambda|_{max} = 34.5$, $|\Delta x|_{min} = 0.0163$

$\alpha = 0.03$

(b) $|\lambda|_{max} = 140.4$, $|\Delta x|_{min} = 0.00341$

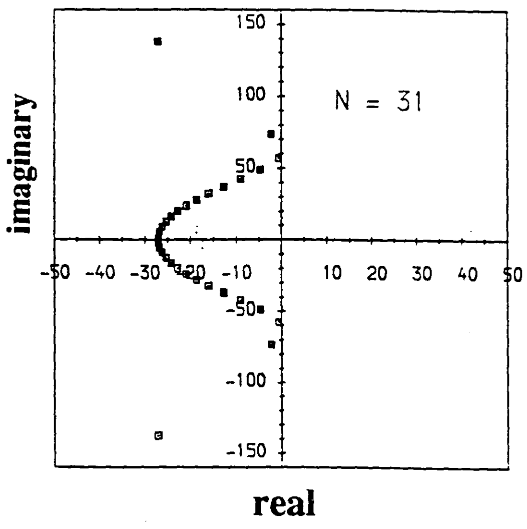
$\alpha = 0.005$

Fig.4.7 Convection Operator Eigenvalues with Grid of Case V



(a) $|\lambda|_{max} = 31.5$, $|\Delta x|_{min} = 0.0159$

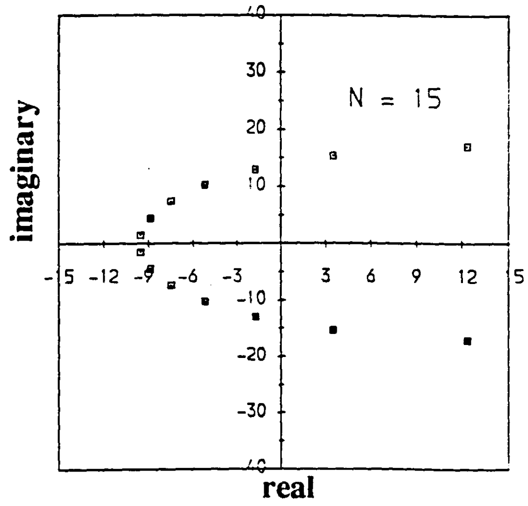
$\alpha = 1.28$



(b) $|\lambda|_{max} = 140.3$, $|\Delta x|_{min} = 0.00315$

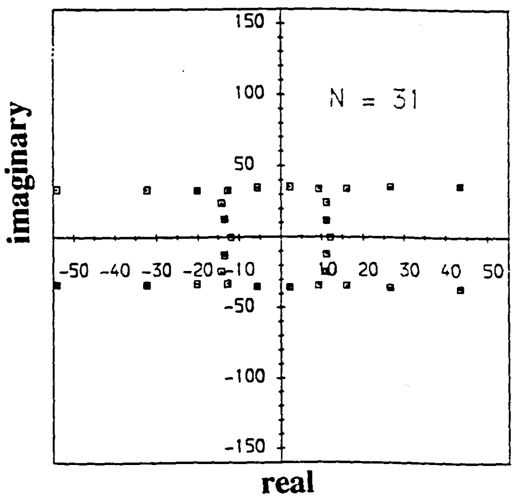
$\alpha = 1.15$

Fig.4.8 Convection Operator Eigenvalues with Grid of Case VI



(a) $|\lambda|_{max} = 21.1$, $|\Delta x|_{min} = 0.01204$

$\beta = 0.03$



(b) $|\lambda|_{max} = 63.81$, $|\Delta x|_{min} = 0.00844$

$\beta = 0.20$

Fig.4.9 Convection Operator Eigenvalues with Grid of Case VII

$$\bar{x} = \xi = 2\cdot(1 - \beta)\cdot(x^2 - x) + \beta\cdot x \quad , \quad \beta \geq 0 \ , \ x \leq 0.5 \qquad (4.62a)$$

$$\bar{x} = 1. - \xi \quad , \ x > 0.5 \qquad (4.62b)$$

Using (4.62), we obtain case VII as

Case VII : Transformed from Case III with $\beta < 1$

Fig. 4.9 shows the eigenvalues of case VII. Obviously, the grid stretched near the middle point does not improve the stability behaviour. Actually, when N is a small number, the structure of the eigenvalue does not change much, but when N becomes a large number, the structure of the eigenvalues changes a lot. It tends to be symmetrical about the origin.

4.5.2 The Diffusion Operator

The diffusion operator is chosen as

$$L(u) = \frac{\partial^2 u}{\partial x^2} \qquad \text{on } [0,\ 1] \qquad (4.63)$$

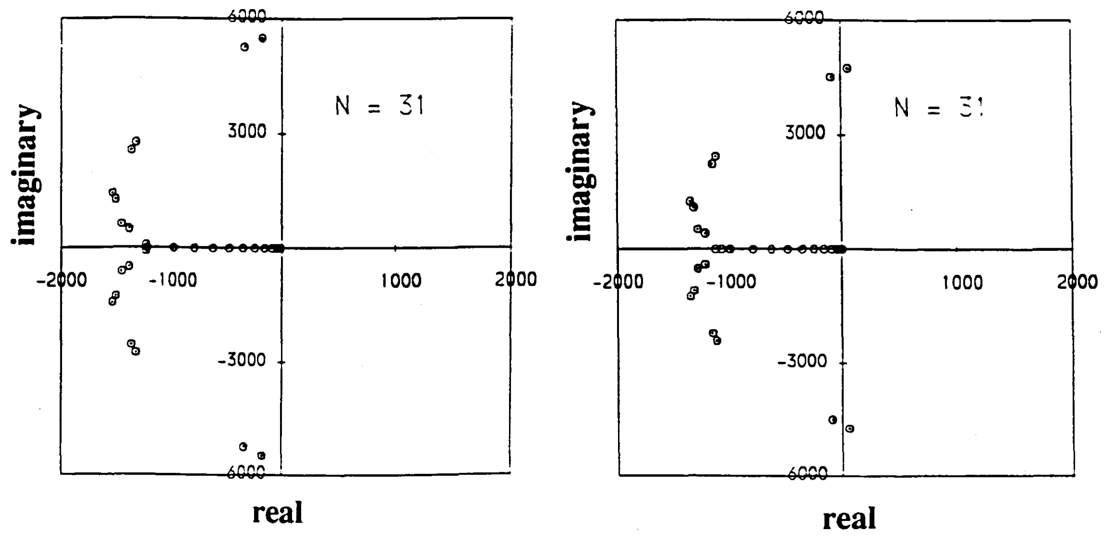
The boundary condition we will impose is of Dirichlet type

$$u(0) = u(1) = 0 \qquad (4.64)$$

or of Neumann type

$$\frac{\partial u}{\partial x}(0) = \frac{\partial u}{\partial x}(1) = 0 \qquad (4.65)$$

When the grid of case I was used, the eigenvalues for both the Dirichlet and Neumann boundary conditions are real numbers. But the Neumann boundary condition can give smaller eigenvalues than the Dirichlet boundary condition, thus the former may allow a larger time step size to be used. For example, the maximum eigenvalue of N = 31, i.e. 1.5443×10^5, for the Dirichlet condition, can be reduced to 4.6665×10^4 for the Neumann condition. This is also the case when the grid of case II was used in which the eigenvalues for the Dirichlet condition are real numbers but are not for the Neumann condition. When N = 31, the maximum eigenvalue is $(-5.5389\times10^4,\ 0)$ for the Dirichlet condition, and $(-1.6801\times10^4,\ 8.0125\times10^3)$ for the Neumann condition. Although the Neumann condition can give smaller eigenvalues, it may cause stability problems. When the grid of case III was used with the Dirichlet condition, the real part of all the eigenvalues are negative, but when the Neumann condition was applied, the real part of the maximum eigenvalue is positive which can cause the computation to be unstable. Fig.4 10 shows the eigenvalues with grid of case III for the Dirichlet and Neumann conditions.



(a) Dirichlet Type Condition (b) Neumann Type Condition

$|\lambda|_{max} = 5.493\times10^3$ $|\lambda|_{max} = 4.734\times10^3$

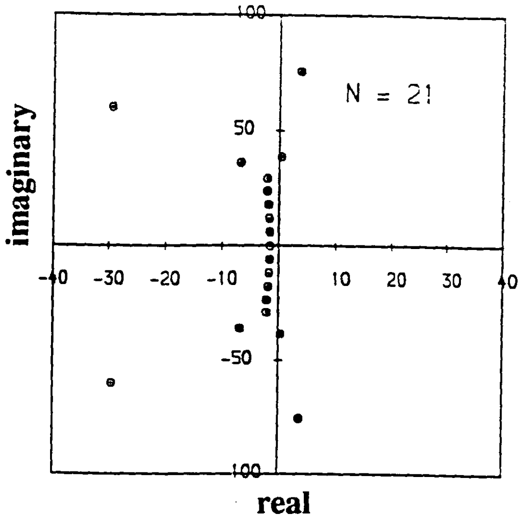
Fig.4.10 Diffusion Operator Eigenvalues with Grid of Case III

4.5.3 The Convection-Diffusion Operator

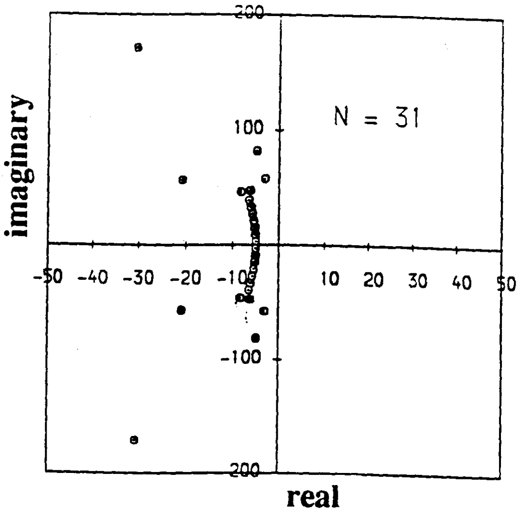
We consider the convection-diffusion operator

$$L(u) = \nu\cdot\frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial x} \quad \text{on } [0,\ 1] \qquad (4.66)$$

with a Dirichlet type boundary condition. When $\nu = O(1)$, this equation is dominated by convection and diffusion. When $\nu \ll O(1)$, this equation is mainly dominated by convection. It is found that when $\nu = O(1)$, the real parts of all the eigenvalues for all of the cases are strictly negative, but when ν is very small, the real parts of some eigenvalues may be positive, leading to a stability problem. It is found that the minimum ν for keeping stability is greatly affected by the distribution of grid points. For example, to obtain a stable solution, the minimum value of ν is around 0.05 for the grid of case III, and 0.0015 for the grid of cases I and II when N=21. Thus for the case of very small ν, the uniform grid is not recommended. The instability problem can be removed by increasing the number of grid points for all the cases when ν is very small. If the number of grid points is kept the same, it is useful to explore the behaviour if the grid is stretched or relaxed near the boundary. We have found that, for the grid of cases II and III, the real parts of all the eigenvalues can be negative if the grid is stretched near the boundary. This is not true for the grid of case I. Fig. 4.11 - 4.12 show the eigenvalues of the convection-diffusion operator with ν = 0.001 for the grid of case I. It is clear from these figures that, when N = 21, the real part of the maximum eigenvalue is positive, but when N = 31, the real parts of all the eigenvalues are strictly negative. Keeping N = 21, when the grid is stretched near the boundary, then more eigenvalues lie in the right half plane, but when the grid is relaxed near the boundary, the real parts of all the eigenvalues are kept in the left half plane. It may be concluded that for the convection-diffusion operator, the stability can be improved by stretching the grid near the boundary for some cases of grid, and by relaxing the grid near the boundary for other cases of grid.

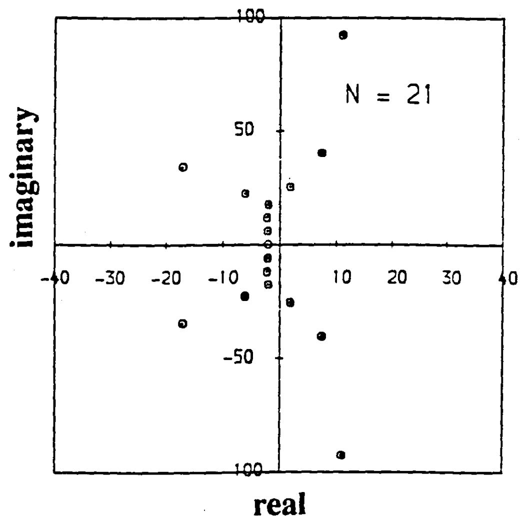


(a) N = 21, ν = 0.001



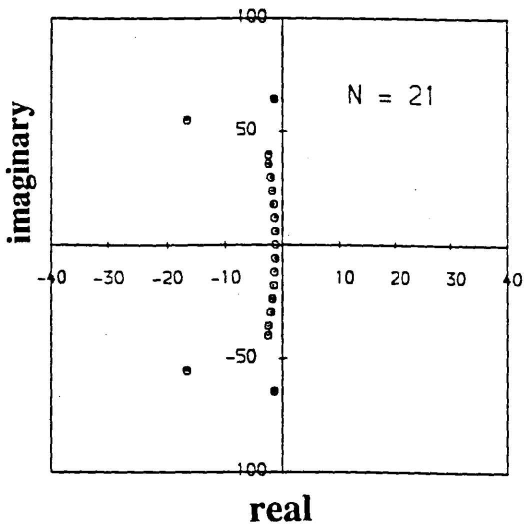
(b) N = 31, ν = 0.001

Fig.4.11 Convection-Diffusion Operator Eigenvalues with Grid of Case I



(a) Stretched near the boundary

α = 0.80, ν = 0.001



(b) Relaxed near the Boundary

α = 1.20, ν = 0.001

Fig.4.12 Convection-Diffusion Operator Eigenvalues with Grid of Case VI

4.6 Applications To Model Problems

4.6.1 Solutions of the Burger's Equations

Firstly, we consider the one-dimensional unsteady problem

$$u_t + u\cdot u_x = \varepsilon\cdot u_{xx} \qquad (4.67)$$

$$x \in [0,\ 1], \quad t \in [0,\ T]$$

with initial condition

$$u(x,\ 0) = f(x) \qquad (4.68)$$

where ε is a constant, T is a specified time. To obtain the analytical solution of (4.67) for comparison purposes, and using the following transformation

$$u(x,\ t) = -2\cdot\varepsilon\cdot W_x(x,\ t)/W(x,\ t) \qquad (4.69)$$

$$f(x) = -2\cdot\varepsilon\cdot g_x(x)/g(x)\ , \qquad (4.70)$$

equation (4.67) can be reduced to a linear heat conduction equation as follows

$$W_t = \varepsilon\cdot W_{xx} \qquad (4.71)$$

with $W(x,\ 0) = g(x)$.

For the test case here, f(x) is chosen as

$$f(x)=-2\varepsilon[b\cdot\pi\cdot\cos(\pi x)+0.5\cdot c\cdot\pi\cdot\cos(0.5\pi x)]/[b\cdot\sin(\pi x)+c\cdot\sin(0.5\pi x)+d]$$

The analytical solution of this can be expressed as

$$W(x,t)=b\cdot\exp(-\varepsilon\pi^2t)\cdot\sin(\pi x)+c\cdot\exp(-0.25\varepsilon\pi^2t)\cdot\sin(0.5\pi x)+d\ ,$$

where b, c, d are the constants and chosen as

$$b = 0.2,\ c = 0.1,\quad d = 0.3,\quad \varepsilon = 0.01\ .$$

After discretization by GDQ, the resulting ordinary differential equations are solved by the 4-stage Runge-Kutta scheme shown in section 4.3. Table I lists the computational results using a uniform grid with different number of grid points, 7, 11, 21. The corresponding analytical results were also included in this Table for comparison. The time step size was chosen as 0.01. Clearly, the numerical solution is very accurate.

Table I Unsteady Solution of Burger's Equation

t	x	Computational			Analytical
		N = 7	N = 11	N = 21	
	0.0	-0.051923	-0.051923	-0.051923	-0.051923
0.1	0.5	-0.003897	-0.003897	-0.003897	-0.003897
	1.0	-0.031138	-0.031126	-0.031126	-0.031127
	0.0	-0.050243	-0.050226	-0.050214	-0.050215
0.5	0.5	-0.003917	-0.003917	-0.003917	-0.003917
	1.0	0.030068	0.029993	0.029995	0.029995
	0.0	-0.048263	-0.048219	-0.048170	-0.048168
1.0	0.5	-0.003939	-0.003939	-0.003939	-0.003939
	1.0	0.028819	0.028628	0.028640	0.028638

Next, we consider the two-dimensional steady problem by solving

$$u_t + c\cdot u_x + d\cdot u_y = \varepsilon\cdot(u_{xx} + u_{yy}) \tag{4.72}$$

with boundary conditions for $t > 0$

$$u(x,0,t)=\{1-\exp[(x-1)\cdot c/\varepsilon]\}/[1-\exp(-c/\varepsilon)],\quad u(x,1,t)=0,\quad 0\le x\le 1$$

$$u(0,y,t)=\{1-\exp[(y-1)\cdot d/\varepsilon]\}/[1-\exp(-d/\varepsilon)],\quad u(1,y,t)=0,\quad 0\le y\le 1\ .$$

The exact solution to (4.72) is

$$u(x,y) = \frac{1-\exp[(x-1)\cdot c/\varepsilon]}{1-\exp(-c/\varepsilon)} \cdot \frac{1-\exp[(y-1)\cdot d/\varepsilon]}{1-\exp(-d/\varepsilon)} \quad . \tag{4.73}$$

Using GDQ, we have employed the grid of cases I, II and III to simulate this problem, and found that, when N = 11, the allowable maximum time step size is 1.1×10^{-3} for case I, 3.20×10^{-3} for case II and 7.10×10^{-3} for case III, and that the converged results for all three cases are nearly the same. This confirms the findings from the stability and eigenvalue analysis in the above section. Table II lists the computational results using the grid of case III with N = 11. Some exact

Table II The Steady Solution of 2D Burger's Equation

$c = 1.0, \quad d = 2.0, \quad \varepsilon = 0.5$

y	0.20	0.40	0.60	0.80
x	Computed by GDQ (N = 11, CPU = 0.44 sec.)			
0.20	0.901911	0.854935	0.750394	0.517749
0.40	0.789693	0.748554	0.657015	0.453316
0.60	0.622288	0.589865	0.517726	0.357209
0.80	0.372555	0.353141	0.309950	0.213851
x	Computed by FD* (N = 51, CPU = 17.15 sec.)			
0.20	0.901928	0.854973	0.750462	0.517836
0.40	0.789720	0.748616	0.657117	0.453437
0.60	0.622318	0.589932	0.517833	0.357328
0.80	0.372578	0.353191	0.310026	0.213933
x	Exact			
0.20	0.901916	0.854945	0.750410	0.517764
0.40	0.789702	0.748575	0.657046	0.453345
0.60	0.622299	0.589890	0.517764	0.357244
0.80	0.372563	0.353160	0.309979	0.213878

* ——— Time-Split MacCormack Finite Difference Scheme

results and numerical results given by a time-split MacCormack scheme are also included in this table. For the finite difference simulation, the allowable maximum time step size was used. The CPU time required on the IBM 3090 are also shown in the table. It is clear that the GDQ results are more accurate than the finite difference results even though fewer grid points are used, and they result from considerably less computation time.

4.6.2 Solution of the Integral Equation

We will use the technique of GIQ developed in Chapter 3 to solve the model integral equation

$$y(x) = 0.5\cdot x\cdot(1-x) + \int_0^1 K(x,\xi)\cdot y(\xi)\cdot d\xi \tag{4.74}$$

with a symmetrical kernel

$$K(x,\xi) = \begin{cases} \xi(1-x) & \text{when } 0\le\xi\le x \\ x(1-\xi) & \text{when } x\le\xi 1 \end{cases} .$$

This equation has an exact solution

$$y(x) = A\cdot\sin(x) + B\cdot\cos(x) - 1 \tag{4.75}$$

where $A = \tan(0.5)$ and $B = 1$

Table III Results of the Integral Equation

x	0.040507	0.118239	0.226900	0.357685	0.500000
Computed by W^I	0.021312	0.057509	0.097403	0.128106	0.139771
Computed by $\tilde{W}^I$	0.021143	0.057211	0.095778	0.128198	0.137014
Exact	0.021303	0.057462	0.097264	0.127974	0.139494

After discretization by GIQ for the integral, the resultant algebraic equations system is solved by a direct method. Table III shows the computed and the exact results using weighting coefficients W^I and $\tilde{W}^I$ from equations (3.76) and (3.72). Since y(x) is symmetrical with respect to x = 0.5, only the results in [0, 0.5] are shown. The solution was obtained with N = 11. From this table, it is clear that the weighting coefficients W^I give more accurate results than $\tilde{W}^I$.

4.7 Concluding Remarks

It has been found that GDQ and GIQ are global methods, which can achieve the same accuracy using just a few grid points as the conventional finite difference scheme using a large number of grid points. It was also shown that GDQ discretization is consistent, the stability conditions for both the semi-discrete equation and the full-discrete equation are dependent on the eigenvalues of the spatial discretization matrix obtained by GDQ. The distribution of the grid points was found to have a considerable influence on the stability condition. Grid stretching near the boundary can improve the stability, but the grid stretching near the middle point makes it worse even though the minimum step size is very small. This is a case of the global method differing from the low order local method. For second order differential equation problems, the types of the boundary conditions can also effect the stability. Comparing with the Neumann type boundary condition, the Dirichlet type boundary condition can causes more stable results, but give larger value of the maximum eigenvalue. This means that it allows a smaller time step size and thus needs more time steps to steady resolution. For the convection-diffusion problem, increasing the number of grid points can improve stability, and stretching the grid near the boundary does likewise.

CHAPTER FIVE

SOLUTIONS OF TWO-DIMENSIONAL INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

5.1 Introduction

In engineering, many fluid flow problems such as encountered in low speed aerodynamics, industrial channel flows, hydraulics can be approximated as incompressible flow. If a Newtonian fluid is considered, these flows are governed, in mathematical terms, by the incompressible Navier-Stokes (N-S) equations which have been discussed in Chapter Two.

The incompressible N-S equations are a mixed set of elliptic-parabolic equations which can be written in several forms. One of the most popular methods for solving the **2D** incompressible N-S equations is through the use of the **vorticity-stream function** approach. This scheme reduces the original equations to a transport equation for vorticity ω, and a Poisson equation for stream function ψ. The successful application of this approach has been the subject of contributions by many researchers such as Burggraf (1966), Osswald et al (1985), Ku et al (1985), Morrison and Napolitano (1988). In the vorticity-stream function approach, however, the implementation of boundary conditions for ψ is not straight forward. For example, two physical boundary conditions for the velocity u, v can give two boundary conditions for ψ, but current numerical techniques normally use only one boundary condition (of the Dirichlet type) for ψ in the calculation. Other researchers (Fasel and Booz 1984, Farouk and Fusegi 1985) have then used the **vorticity-velocity** approach which reduces the original equations to a transport equation for ω, and two Poisson equations for u, v. Compared with the vorticity-stream function approach, this approach needs more computational time.

Another version of the incompressible N-S equations used in the **2D** steady case is the **stream function** approach which reduces the original equations to a 4th order differential equation for ψ. This approach still needs to be justified for a general application because two difficulties appear. One difficulty arises from the solution of a 4th order differential equation, resulting in complexity of the algorithm. Another arises from the implementation of the boundary conditions for the stream function. The stream function-related approaches for solving the incompressible N-S equations lose their attractiveness when applied to a three-dimensional flow because a single scalar stream function does not exist in this case. As a consequence, the **primitive-variable** form is usually used for three-dimensional problems.

One of the schemes for solving the incompressible N-S equations in primitive-variable form is the **artificial compressibility** method of Chorin (1967). In this method, the continuity equation is modified to include an artificial compressibility term which vanishes when the steady state solution is reached. With the addition of this term to the continuity equation, the resultant N-S equations are a mixed set of hyperbolic-parabolic equations which can be solved using a standard numerical approach. A difficulty with this approach lies in the choice of the optimal time step size and the artificial compressibility factor for the general case. In most cases, a value of around 0.5 for the maximum artificial compressibility Mach number can produce a satisfactory result. There are many successful applications of this approach. For details, see, for example, the work of Steger and Kutler (1976), Chang and Kwak (1984), Rizzi and Eriksson (1985). Another scheme in the primitive-variable approach involves using a Poisson equation for pressure in place of the continuity equation. This approach consists of a basic iterative procedure between the velocity and the pressure fields. For an initial approximation of the pressure, the momentum equation is solved to determine the velocity field. The resultant velocity field does not satisfy continuity and has to be corrected. Since this correction has an impact on the pressure field, a related **pressure correction** is defined, obtained by showing that the corrected velocity satisfies the continuity equation. This approach has a wide application in CFD, see, for instance, the work of Patankar and Spalding (1972), Ghia et al (1981), Cebeci et al (1981), Chan et al (1987).

The numerical algorithms described in Chapter 1 can be used to solve the incompressible N-S equations. Chapter Four has shown that GDQ has potential as an attractive technique as a result of the applications to model problems. In this chapter, we will apply GDQ to solve the incompressible **2D** N-S equations, and check out its behaviour to various engineering problems. The vorticity-stream function formulation, and several standard test problems are chosen for demonstration. For application to general problems, the multi-domain GDQ technique is also developed in this chapter. For comparison purposes, the finite difference resolution to the driven cavity flow problem was also included, which was obtained by using a second order time-split

MacCormack finite difference scheme for the vorticity equation, and a SIP approach for the stream function equation.

5.2 Discretization and Boundary Conditions

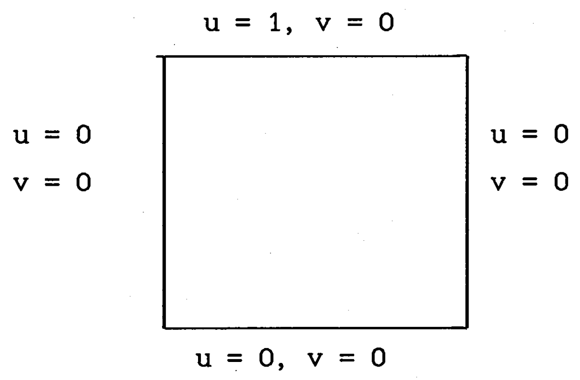


Fig. 5.1 Problem Definition of the Driven Cavity Flow

For demonstration purposes, we will choose the vorticity-stream function formulation in the Cartesian coordinate system to solve the driven cavity flow problem and show the discretization of the governing equations and the treatment of the boundary conditions.

The non-dimensional vorticity-stream function formulation of the **2D** N-S equations is

$$\omega_t + u\cdot\omega_x + v\cdot\omega_y = \frac{1}{Re}\cdot\nabla^2\omega \tag{5.1}$$

$$\nabla^2\psi = \omega \tag{5.2}$$

where ω, ψ, u, v, Re, t, x, y, ∇^2 have the same meaning as shown in Chapter Two. For the driven cavity flow problem, the physical boundary conditions are

$$u = v = 0\ , \quad \text{at } x = 0,\ 1, \quad 0 \le y < 1 \tag{5.3}$$

$$u = v = 0\ , \quad \text{at } y = 0\ , \quad 0 \le x \le 1 \tag{5.4}$$

$$u = 1,\ v = 0, \quad \text{at } y = 1 \ , \quad 0 < x < 1 \tag{5.5}$$

The problem definition is shown in Fig. 5.1. Clearly, the two corner points on the upper wall are singular points which in a numerical technique normally cause difficulties in treating the boundary conditions. Since

$$u = \frac{\partial\psi}{\partial y} \ , \quad v = -\frac{\partial\psi}{\partial x} \tag{5.6}$$

the boundary conditions (5.3)-(5.5) for u, v can be transformed to

$$\psi = \psi_x = 0 \ , \quad \text{at } x = 0,\ 1 \ , \quad 0 \le y < 1 \tag{5.7}$$

$$\psi = \psi_y = 0 \ , \quad \text{at } y = 0 \ , \quad 0 \le x \le 1 \tag{5.8}$$

$$\psi = 0,\ \psi_y = 1 \ , \ \text{at } y = 1 \ , \quad 0 < x < 1 \tag{5.9}$$

for stream function ψ. Thus, there are eight boundary conditions for ψ, each boundary with two types (one Dirichlet and one Neumann).

With the mesh of N grid points in the x direction and M grid points in the y direction, when the derivatives of (5.1) and (5.2) are approximated by GDQ, the discretized forms of (5.1), (5.2) become

$$\frac{d\omega_{ij}}{dt} + u_{ij}\cdot\sum_{k=1}^{N} w_{ik}^{(1)}\cdot\omega_{kj} + v_{ij}\cdot\sum_{k=1}^{M}\bar{w}_{jk}^{(1)}\cdot\omega_{ik} = \frac{1}{Re}\cdot\left[\sum_{k=1}^{N} w_{ik}^{(2)}\cdot\omega_{kj} + \sum_{k=1}^{M}\bar{w}_{jk}^{(2)}\cdot\omega_{ik}\right] \tag{5.10}$$

$$\sum_{k=1}^{N} w_{ik}^{(2)}\cdot\psi_{kj} + \sum_{k=1}^{M}\bar{w}_{jk}^{(2)}\cdot\psi_{ik} = \omega_{ij} \ . \tag{5.11}$$

The boundary conditions for (5.10) can be obtained from (5.2) with the discretized form as

$$\omega_{1j} = \sum_{k=1}^{N} w_{1k}^{(2)}\cdot\psi_{kj} \ , \quad j = 1,\ 2,\ \cdots,\ M \tag{5.12}$$

$$\omega_{Nj} = \sum_{k=1}^{N} w_{Nk}^{(2)}\cdot\psi_{kj} \ , \quad j = 1,\ 2,\ \cdots,\ M \tag{5.13}$$

$$\omega_{i1} = \sum_{k=1}^{M}\bar{w}_{1k}^{(2)}\cdot\psi_{ik} \ , \quad i = 2,\ 3,\ \cdots,\ N-1 \tag{5.14}$$

$$\omega_{iM} = \sum_{k=1}^{M} \bar{w}_{Mk}^{(2)} \cdot \psi_{ik} , \qquad i = 2,\ 3,\ \cdots,\ N-1 . \tag{5.15}$$

Using (5.6), the velocity can be determined by

$$u_{ij} = \sum_{k=1}^{M} \bar{w}_{jk}^{(1)} \cdot \psi_{ik} , \quad i=2,3,\cdots,N-1,\ \ j=2,3,\cdots,M-1 \tag{5.16}$$

$$v_{ij} = - \sum_{k=1}^{N} w_{ik}^{(1)} \cdot \psi_{kj} , \quad i=2,3,\cdots,N-1,\ \ j=2,3,\cdots,M-1 . \tag{5.17}$$

The four boundary conditions for ω can also be obtained from the velocity

$$\omega_{1j} = - \sum_{k=1}^{N} w_{1k}^{(1)} \cdot v_{kj} , \qquad j = 1,\ 2,\ \cdots,\ M \tag{5.18}$$

$$\omega_{Nj} = - \sum_{k=1}^{N} w_{Nk}^{(1)} \cdot v_{kj} , \qquad j = 1,\ 2,\ \cdots,\ M \tag{5.19}$$

$$\omega_{i1} = \sum_{k=1}^{M} \bar{w}_{1k}^{(1)} \cdot u_{ik} , \qquad i = 2,\ 3,\ \cdots,\ N-1 \tag{5.20}$$

$$\omega_{iM} = \sum_{k=1}^{M} \bar{w}_{Mk}^{(1)} \cdot u_{ik} , \qquad i = 2,\ 3,\ \cdots,\ N-1 . \tag{5.21}$$

Four Dirichlet boundary conditions for (5.11) can be written as

$$\psi_{i1} = \psi_{iM} = \psi_{1j} = \psi_{Nj} = 0 . \tag{5.22}$$

Another four Neumann boundary conditions, after discretizing by GDQ, can be combined to give

$$\psi_{2,j} = [\sum_{k=1,k\neq 2,N-1}^{N} (w_{1,k}^{(1)} \cdot w_{N,N-1}^{(1)} - w_{N,k}^{(1)} \cdot w_{1,N-1}^{(1)}) \cdot \psi_{k,j}]/AXN \tag{5.23}$$

$$\psi_{N-1,j} = [\sum_{k=1,k\neq 2,N-1}^{N} (w_{N,k}^{(1)} \cdot w_{1,2}^{(1)} - w_{1,k}^{(1)} \cdot w_{N,2}^{(1)}) \cdot \psi_{k,j}]/AXN \tag{5.24}$$

$$\psi_{i,2} = [\bar{w}_{1,M-1}^{(1)} + \sum_{k=1,k\neq 2,M-1}^{M} (\bar{w}_{1,k}^{(1)} \cdot \bar{w}_{M,M-1}^{(1)} - \bar{w}_{M,k}^{(1)} \cdot \bar{w}_{1,M-1}^{(1)}) \cdot \psi_{i,k}]/AYM \tag{5.25}$$

$$\psi_{i,M-1} = [-\bar{w}_{1,2}^{(1)} + \sum_{k=1,k\neq 2,M-1}^{M} (\bar{w}_{M,k}^{(1)} \cdot \bar{w}_{1,2}^{(1)} - \bar{w}_{1,k}^{(1)} \cdot \bar{w}_{M,2}^{(1)}) \cdot \psi_{i,k}]/AYM \tag{5.26}$$

where

$$AXN = w_{N,2}^{(1)} \cdot w_{1,N-1}^{(1)} - w_{1,2}^{(1)} \cdot w_{N,N-1}^{(1)}$$

$$AYM = \bar{w}^{(1)}_{M,2} \cdot \bar{w}^{(1)}_{1,M-1} - \bar{w}^{(1)}_{1,2} \cdot \bar{w}^{(1)}_{M,M-1} .$$

Thus there are two boundary conditions for ψ on each boundary.

With the boundary conditions (5.12)-(5.15) or (5.18)-(5.21), the set of (N-2)×(M-2) ordinary differential equations for ω, (5.10), can be solved by the 4-stage Runge-Kutta scheme which was given in Chapter Four. With eight boundary conditions, the set of (N-4)×(M-4) algebraic equations, (5.11), can be solved by LU decomposition which is shown in Appendix E. Noting that the Laplacian operator is a linear operator, we need only decompose the matrix of the equations system (5.11) once and store the inverted matrix elements for all the following time steps.

5.3 Single-Domain Results

In this section, four standard test problems, which have been extensively studied by many researchers using conventional numerical techniques, are chosen to validate the GDQ approach.

5.3.1 Driven Cavity Flow

This flow problem, often chosen as a test case for checking new numerical techniques, has been simulated very extensively. There are a variety of numerical results available for comparison. For example, the vortex centre and the velocity profile through the geometrical centre are presented by Ghia et al (1982) and Ku et al (1985). For numerical simulation, the solutions were obtained in the Reynolds number range from 100 to 1000. The grid of **Case IV** shown in Chapter 4 was used for the GDQ simulation. Particularly, mesh sizes of 13×13, 17×15, 21×17 and 23×21 for Reynolds number of 100, 200, 400, 1000 were used respectively. The initial values for all variables in the interior points are chosen to be zero. For direct comparison of GDQ with conventional numerical techniques, numerical results using a second order time-split MacCormack finite difference scheme (shown in Appendix C) for vorticity equation and a preconditioning technique of SIP (shown in Appendix D) for stream function, are also obtained for a uniform grid of mesh size of 51×51. By numerical experiment, the allowable maximum time step size was used.

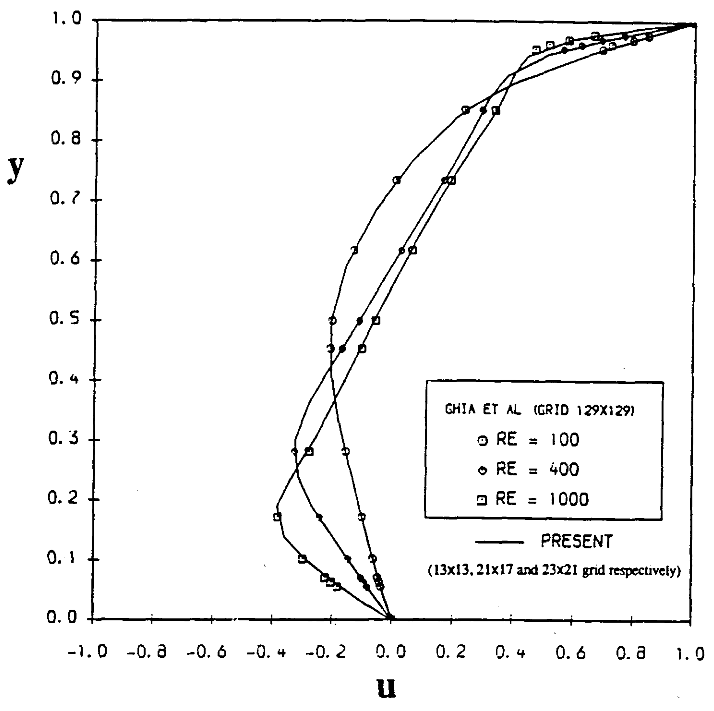
Table I Parameters of Vortex Center for Driven Cavity Flow

Re	Reference	Grid	x	y	ψ	ω
	Ghia et al	129×129	0.6172	0.7344	-0.1034	3.1665
100	Present (GDQ)	13×13	0.615	0.735	-0.1035	3.1547
	Present (FD)	51×51	0.620	0.740	-0.1030	3.1915
	Ku et al	25×15	0.6023	0.6657	-0.1071	2.6345
200	Present (GDQ)	17×15	0.600	0.665	-0.1089	2.6686
	Present (FD)	51×51	0.600	0.660	-0.1072	2.6673
	Ghia et al	129×129	0.5547	0.6055	-0.1139	2.2947
400	Present (GDQ)	21×17	0.555	0.605	-0.1131	2.2794
	Present (FD)	51×51	0.560	0.600	-0.1105	2.2428
	Ghia et al	129×129	0.5313	0.5625	-0.1179	2.0497
1000	Present (GDQ)	23×21	0.530	0.565	-0.1184	2.0649
	Present (FD)	51×51	0.540	0.560	-0.1103	1.9326

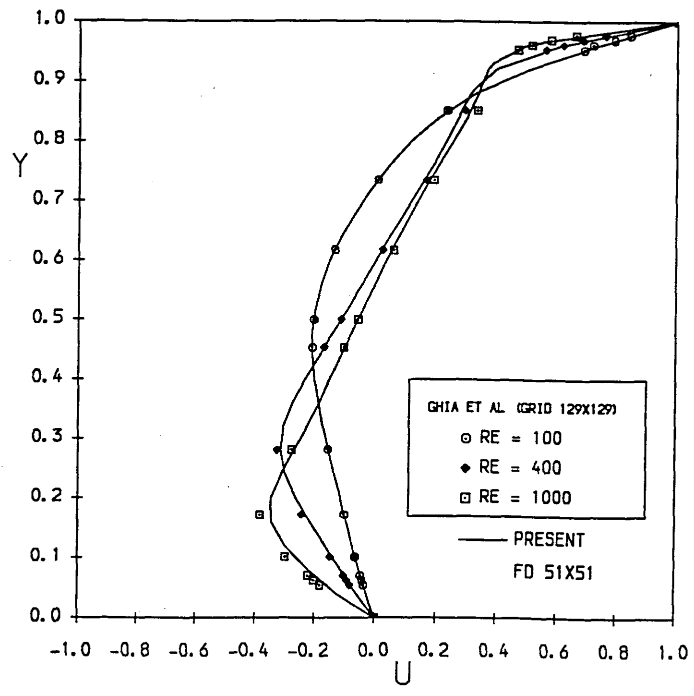
Table II The CPU Time Taken by Driven Cavity Flow Simulation

Re	100	200	400	1000
CPU time (GDQ) seconds	4.27	6.69	16.99	33.79
CPU time (FD) seconds	442.73	536.98	601.50	732.90

As is well known, during the transition from fluid motion governed mainly by viscosity to one where inertia forces dominate the flow with increasing Reynolds number, the core of the primary vortex centre seems to behave as a solid boundary. One indication of this behaviour is that, the location of the primary vortex center moves to the geometric center of the cavity with increasing Reynolds number, which is shown clearly in Table I. Table I includes the results of GDQ approximation, the finite difference approximation and results given by Ghia et al (1982) and Ku et al (1985). It is clear from Table I that the GDQ approximation is very accurate even though just a few grid points were used, compared with the finite difference approximations using a large number of grid points. Table II shows the CPU time on the IBM 3090 by the GDQ approximation and the time-split MacCormack finite difference approximation. The mesh size used is the same as Table I. We see, clearly, that the GDQ approach requires much less CPU time for accurate results. The computed horizontal velocity profiles along the vertical line through the geometric center of the cavity, are displayed in Fig. 5.2, and the vertical velocity profiles along the horizontal line through the geometric center of the cavity are shown in Fig. 5.3. Also included in Fig. 5.2 and 5.3 are the results of Ghia et al. Fig 5.4 and 5.5 show the streamlines computed by GDQ and the MacCormack finite

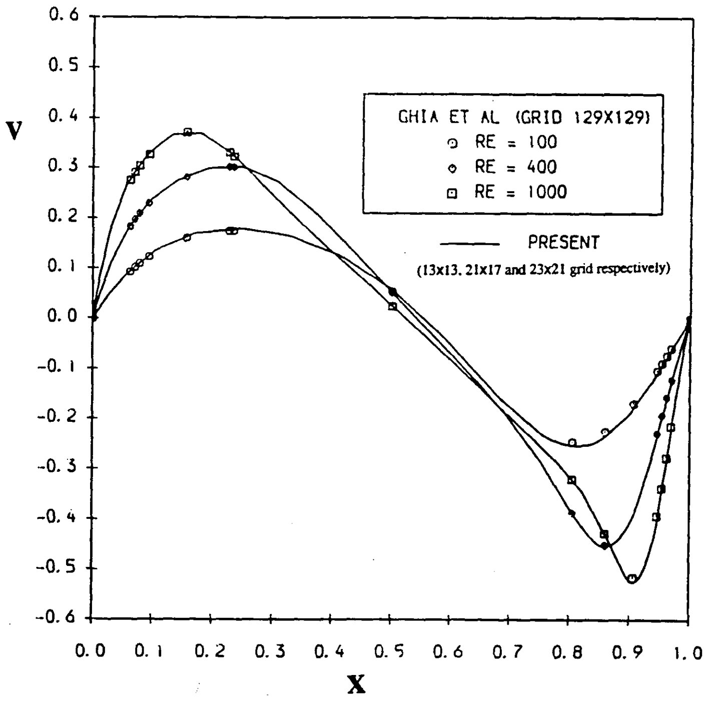


(a) Results from GDQ Approach

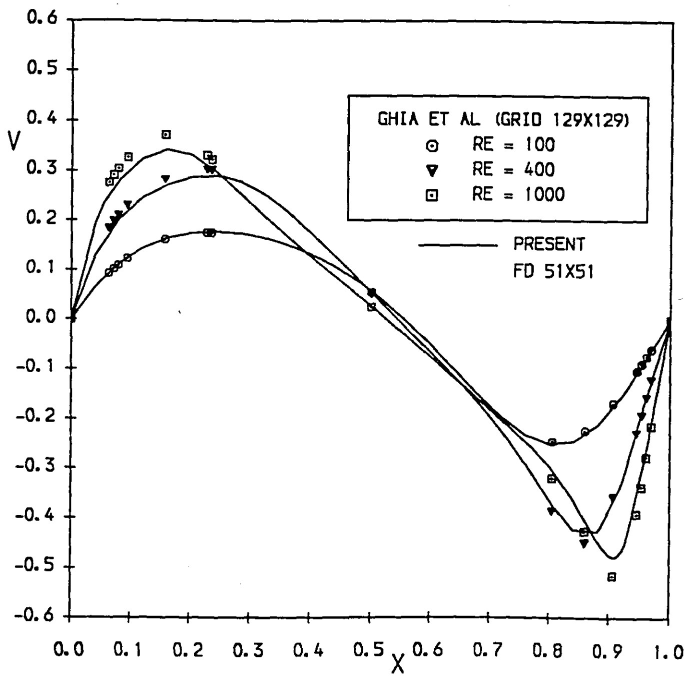


(b) Results from the MacCormack Time-Split Finite Difference Scheme

Fig. 5.2 Horizontal Velocity past the Geometric Center of the Cavity

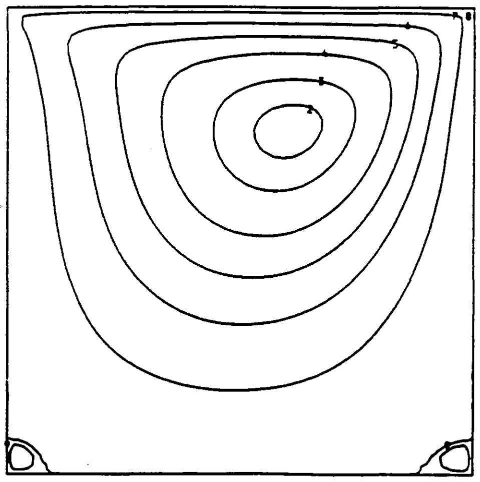


(a) Results from GDQ Approach

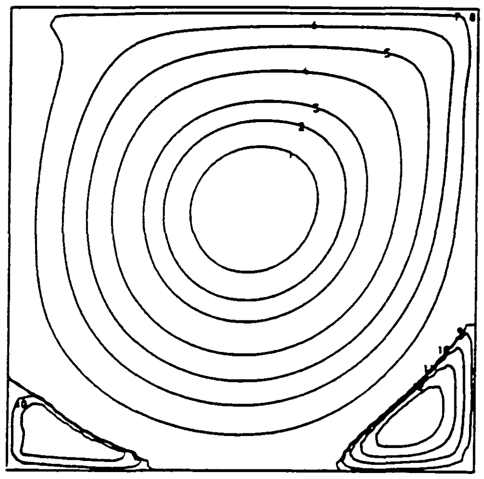


(b) Results from the MacCormack Time-Split Finite Difference Scheme

Fig. 5.3 Vertical Velocity past the Geometric Center of the Cavity

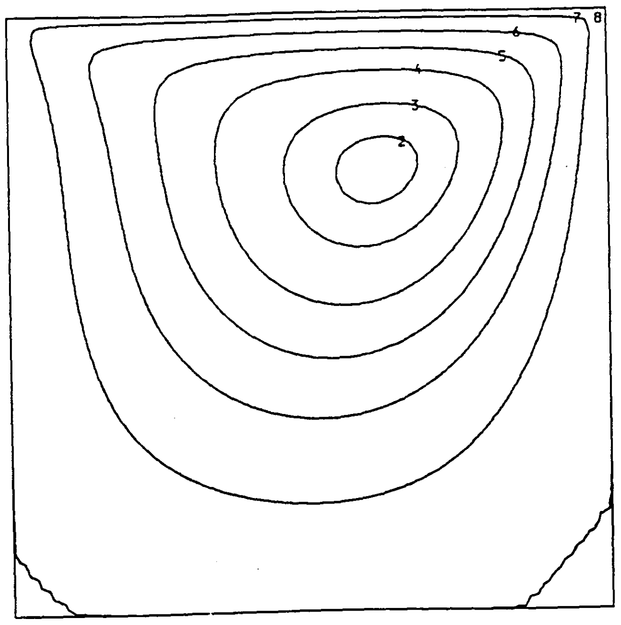


(a) Re = 100, grid 13X13

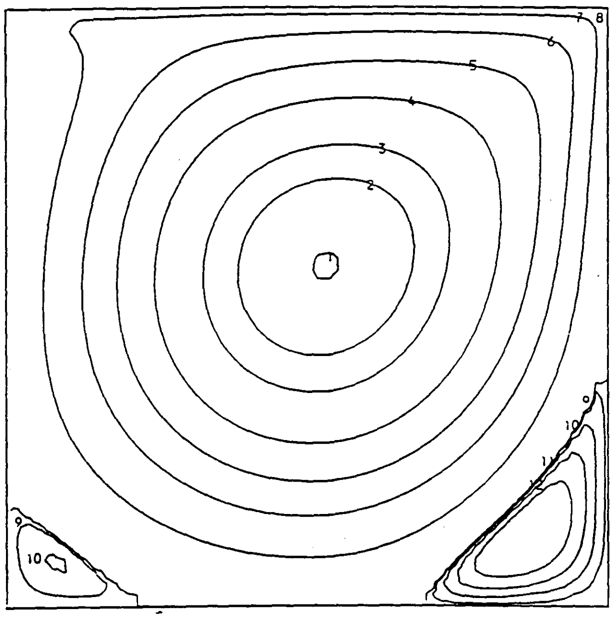


(b) Re = 1000, grid 23X21

Fig. 5.4 Streamlines for Driven Cavity Flow (GDQ Results)



(a) Re = 100, grid 51X51



(b) Re = 1000, grid 51X51

Fig. 5.5 Streamlines for Driven Cavity Flow (FD Results)

difference scheme for the Reynolds number cases of 100, 1000. The values of the streamlines are listed in Table III. Fig. 5.2-5.5 show clearly that the GDQ results using few grid points are more accurate than the finite difference results using a lot of grid points.

Table III Values of Streamlines in Fig. 5.4 and Fig. 5.5

Contour number	Value of ψ	Contour number	Value of ψ
1	-1.10×10^{-1}	8	-1.00×10^{-6}
2	-1.00×10^{-1}	9	5.00×10^{-5}
3	-9.00×10^{-2}	10	2.00×10^{-4}
4	-7.00×10^{-2}	11	5.00×10^{-4}
5	-5.00×10^{-2}	12	1.00×10^{-3}
6	-3.00×10^{-2}	13	2.00×10^{-3}
7	-1.00×10^{-2}		

5.3.2 Natural Convection in a Square Cavity

The buoyancy driven flow in a square cavity with vertical sides which are differentially heated is a suitable vehicle for testing and validating numerical methods used for a wide variety of practical problems. This problem has been extensively studied by many researchers such as Phillips (1984), and as illustrated in the paper of G. de Vahl Davis and I.P. Jones (1983) which outlined numerous contributed results reported at the 2nd Conference on Numerical Methods in Thermal Problems, compared with a bench mark solution. The problem being considered here is that of the two-dimensional flow of a Boussinesq fluid of Prandtl number 0.71 in an upright square cavity described in non-dimensional terms by $0\le x\le1$, $0\le y\le1$ with y vertically upwards. The governing equations are (2.21)-(2.23) shown in Chapter Two, and the problem definition and

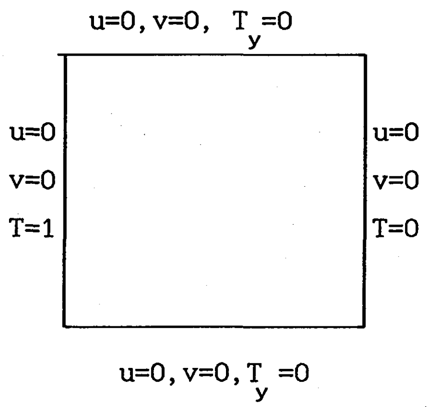


Fig. 5.6 Problem of the Natural Convection in A Square Cavity

the boundary conditions are displayed in Fig. 5.6. Using the same approach as for the driven cavity flow, the resultant set of (N-2)×(M-2) ordinary differential equations for vorticity ω and for temperature T are solved by the 4-stage Runge-Kutta scheme , and the set of (N-4)×(M-4) algebraic equations for ψ are solved by the LU decomposition technique. The treatment of the boundary condition for ω and ψ are the same as for the driven cavity flow problem. Two Dirichlet type boundary conditions in the horizontal direction, and two Neumann type boundary conditions in the vertical direction, are used for T. The grid of case IV was used for GDQ simulation. Numerical results for the values of Ra of 10^3, 10^4, 10^5, 10^6, which have been communicated (1990), were obtained by using the mesh sizes of 13×13, 15×15, 21×17, 21×17 respectively. When the values of the fluid variables are known at the grid points, the full flow field can be determined by equation (3.33). All the following results are based on the interpolated values. For the comparison, the following quantities are calculated

$|\psi|_{mid}$ the stream function at the mid-point of the cavity

$|\psi|_{max}$ the maximum absolute value of the stream function (together with its location)

Table IV GDQ Results of The Natural Convection Problem

	Ra			
	10^3	10^4	10^5	10^6
mesh size	13×13	15×15	21×17	21×17
$\lvert\psi_{mid}\rvert$	1.175	5.075	9.115	16.33
$\lvert\psi\rvert_{max}$	–	–	9.617	16.82
x, y	–	–	0.285, 0.600	0.150, 0.550
u_{max}	3.649	16.19	34.73	64.36
y	0.815	0.825	0.855	0.850
v_{max}	3.697	19.61	68.63	221.80
x	0.180	0.120	0.065	0.035
$\overline{Nu}$	1.1178	2.2454	4.524	8.797
$Nu_{1/2}$	1.1179	2.245	4.526	8.745
Nu_0	1.1179	2.250	4.524	8.837
Nu_{max}	1.506	3.548	7.751	17.13
y	0.090	0.145	0.080	0.045
Nu_{min}	0.6914	0.5860	0.7240	0.9260
y	1	1	1	1

u_{max}	the maximum horizontal velocity on the vertical mid-plane of the cavity (together with its location)
v_{max}	the maximum vertical velocity on the horizontal mid-plane of the cavity (together with its location)
$\overline{Nu}$	the average Nusselt number throughout the cavity
$Nu_{1/2}$	the average Nusselt number on the vertical mid-plane of the cavity
Nu_0	the average Nusselt number on the vertical boundary of the cavity at x = 0
Nu_{max}	the maximum value of the local Nusselt number on the boundary at x = 0 (together with its location)

Nu_{min} the minimum value of the local Nusselt number on the boundary at x = 0 (together with its location)

Table IV lists the GDQ results and Table V and Table VI list the bench mark solutions and the second order finite difference results given from the work of G.de Vahl Davis (1983). Clearly, the GDQ results are very accurate, and nearly the same as the bench mark solutions. It is also found that the GDQ results are more accurate than the finite difference results even though fewer mesh points are used. Table VII shows the CPU time cost on the IBM 3090 by GDQ resolutions. Figure 5.7-5.11 shows isotherms, streamlines, vorticity contours, horizontal velocity contours and the vertical velocity contours.

Table V Bench Mark Solutions of The Natural Convection Problem

	Ra			
	10^3	10^4	10^5	10^6
$\|\psi_{mid}\|$	1.174	5.071	9.111	16.32
$\|\psi\|_{max}$	–	–	9.612	16.75
x,y	–	–	0.285,0.601	0.151,0.547
u_{max}	3.649	16.178	34.73	64.63
y	0.813	0.823	0.855	0.850
v_{max}	3.697	19.617	68.59	219.36
x	0.178	0.119	0.066	0.0379
$\overline{Nu}$	1.118	2.243	4.519	8.800
$Nu_{1/2}$	1.118	2.243	4.519	8.799
Nu_0	1.117	2.238	4.509	8.817
Nu_{max}	1.505	3.528	7.717	17.925
y	0.092	0.143	0.081	0.0378
Nu_{min}	0.692	0.5860	0.7290	0.9890
y	1	1	1	1

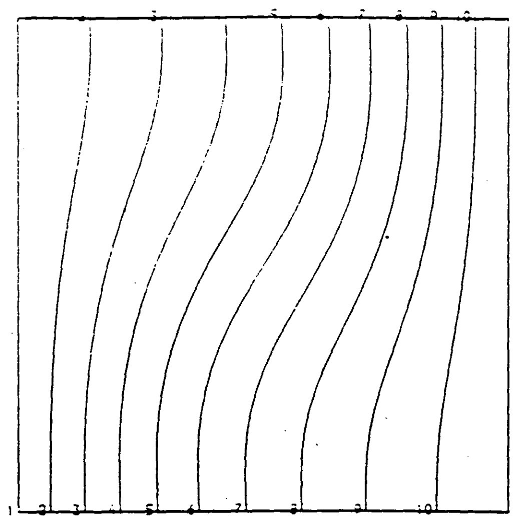
Table VI FD Results of The Natural Convection Problem

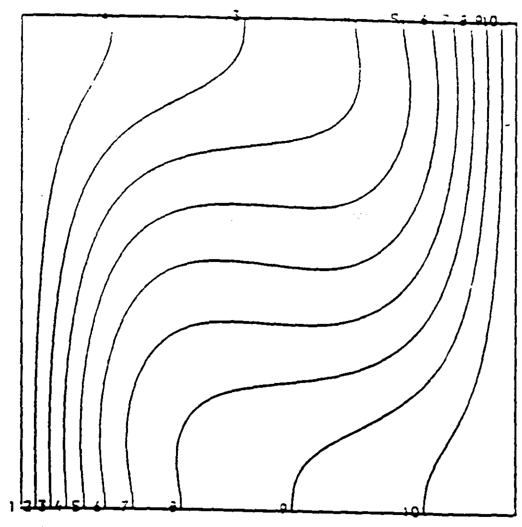
Results are from the work of G.de Vahl Davis

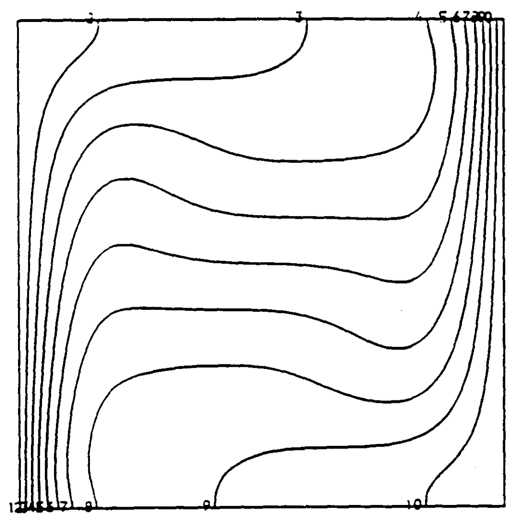
	Ra			
	10^3	10^4	10^5	10^6
mesh size	41×41	41×41	81×81	81×81
$\lvert\psi_{mid}\rvert$	1.174	5.098	9.142	16.53
$\lvert\psi\rvert_{max}$	–	–	9.644	16.961
x,y	–	–	0.285,0.602	0.151,0.543
u_{max}	3.634	16.182	34.81	65.33
y	0.813	0.823	0.855	0.851
v_{max}	3.679	19.509	68.22	216.75
x	0.179	0.120	0.066	0.0387
$\overline{Nu}$	1.116	2.234	4.510	8.798
$Nu_{1/2}$	1.117	2.235	4.512	8.816
Nu_0	1.116	2.242	4.523	8.928
Nu_{max}	1.501	3.545	7.761	18.076
y	0.087	0.149	0.085	0.0456
Nu_{min}	0.694	0.5920	0.7360	1.005
y	1	1	1	1

Table VII CPU Time Taken by Natural Convection Problem Simulation

Ra	10^3	10^4	10^5	10^6
CPU (seconds)	16.01	25.00	93.65	78.85


(a) Ra = 10^3


(b) Ra = 10^4


(c) Ra = 10^5

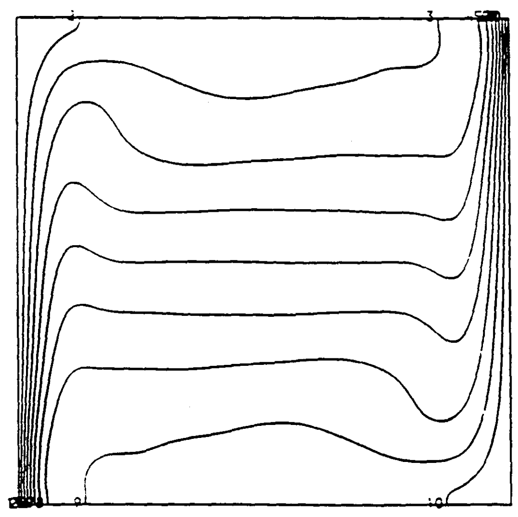
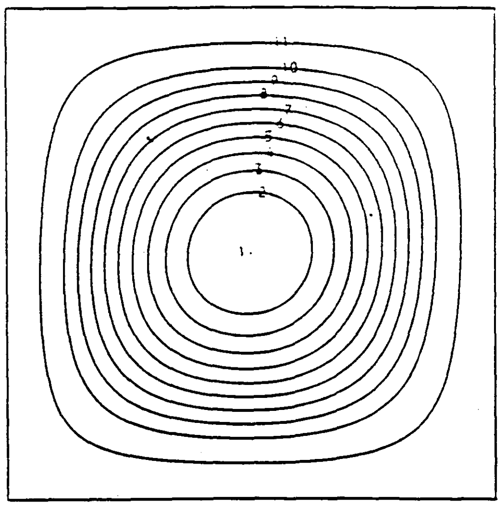

(d) Ra = 10^6

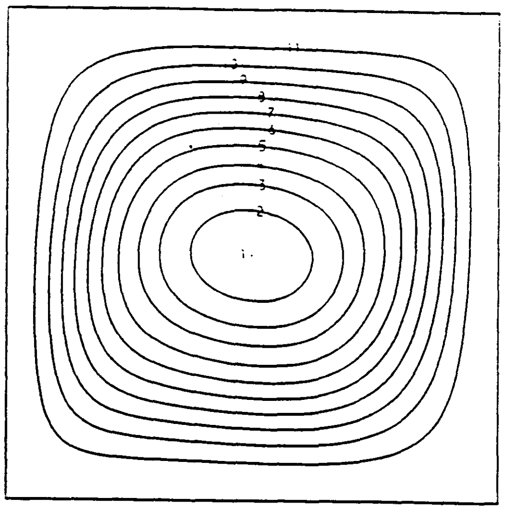
Fig. 5.7 Contour Maps of temperature T

Contours at 0(0.1)1 in each case



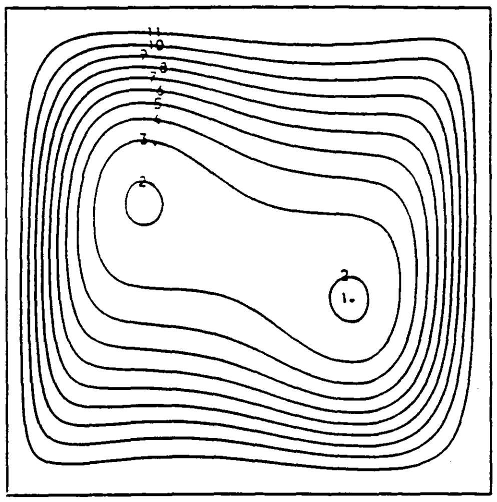
(a) Ra = 10^3; contours at

-1.175, -1.05(0.1)-0.1



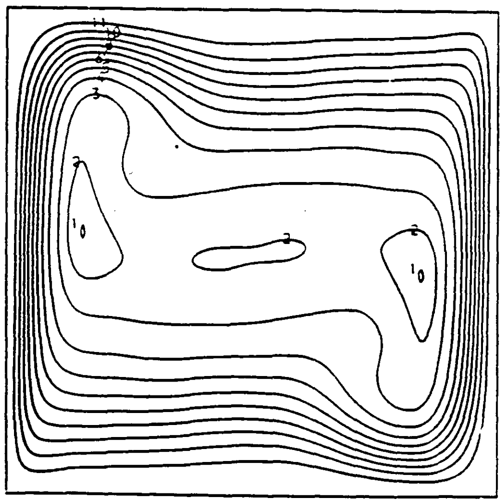
(b) Ra = 10^4; contours at

-5.075,-4.80,-4.5(0.5)-0.50



(c) Ra = 10^5; contours at

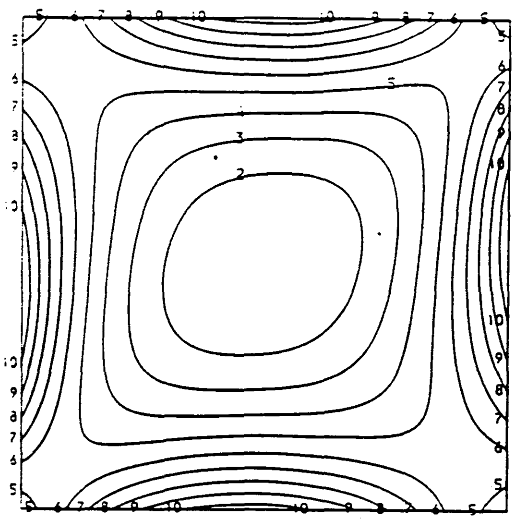
-9.60, -9.47,-8.648(0.9607)-0.96



(d) Ra = 10^6; contours at

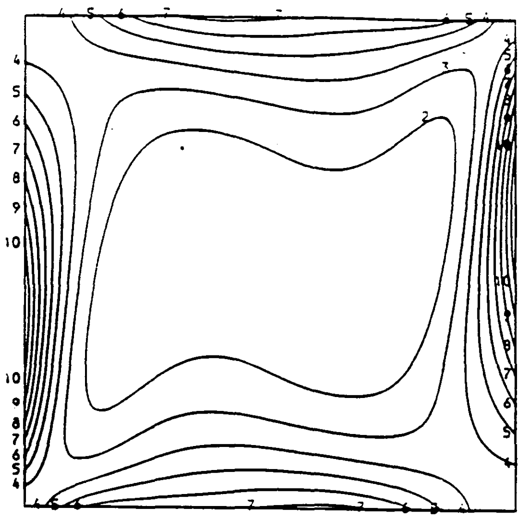
-16.75,-16.00,-15.07(1.675)-1.67

Fig. 5.8 Contour Maps of Stream Function ψ



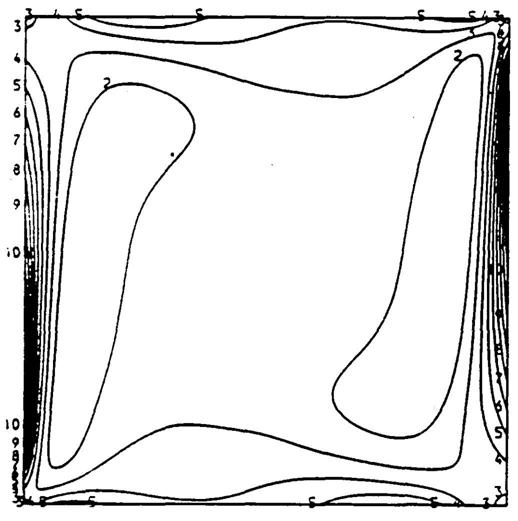
(a) Ra = 10^3; contours at

-32.01(8.328)51.27



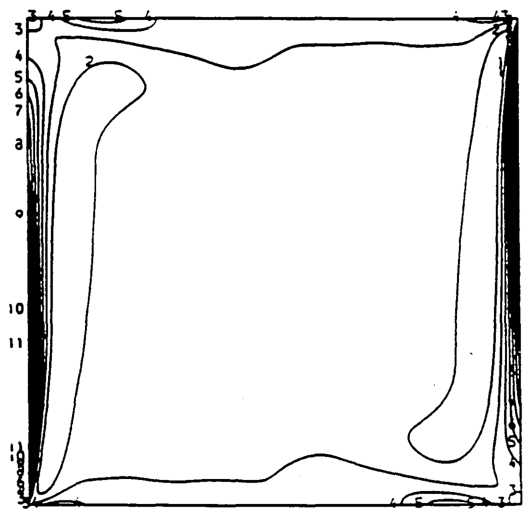
(b) Ra = 10^4; contours at

-124.8(55.17)426.9



(c) Ra = 10^5; contours at

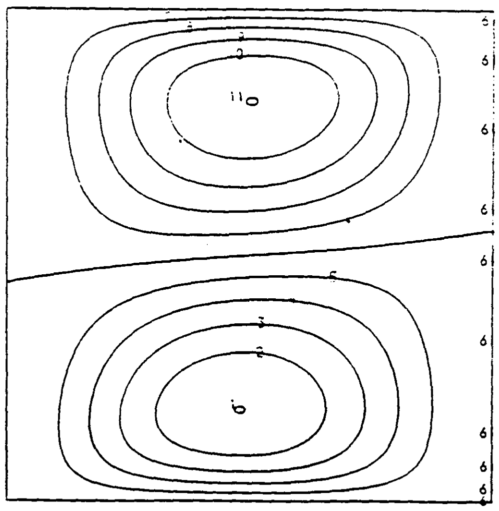
-600.0(322.6)2626.0



(d) Ra = 10^6; contours at

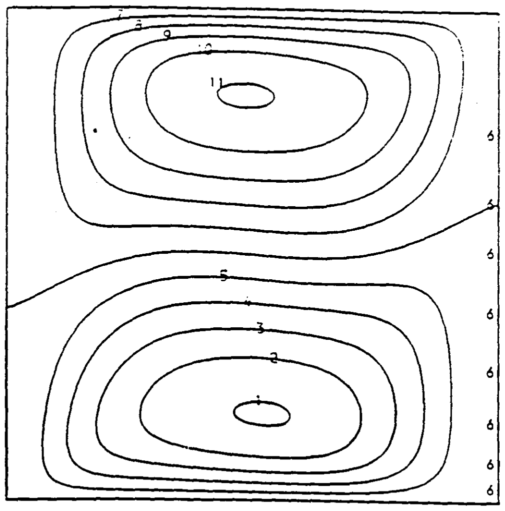
-3178(1847.1)15293

Fig. 5.9 Contour Maps of Vorticity ω



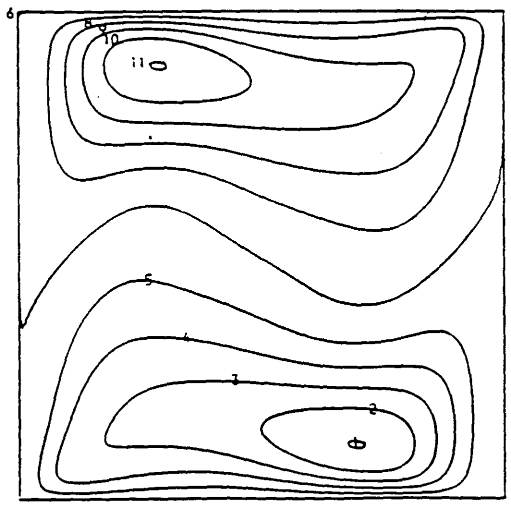
(a) Ra = 10^3; contours at

-3.637(0.7274)3.637



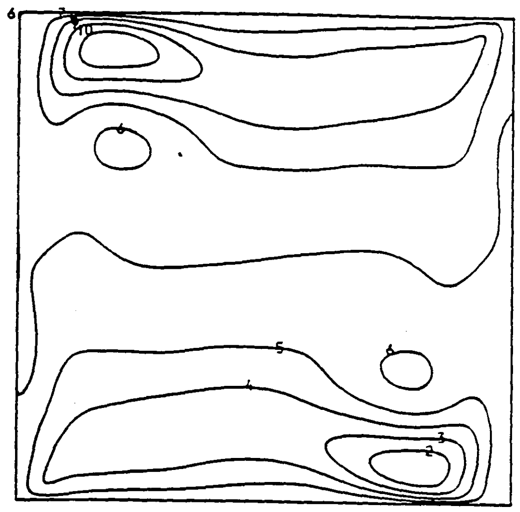
(b) Ra = 10^4; contours at

-16.00(3.20)16.00



(c) Ra = 10^5; contours at

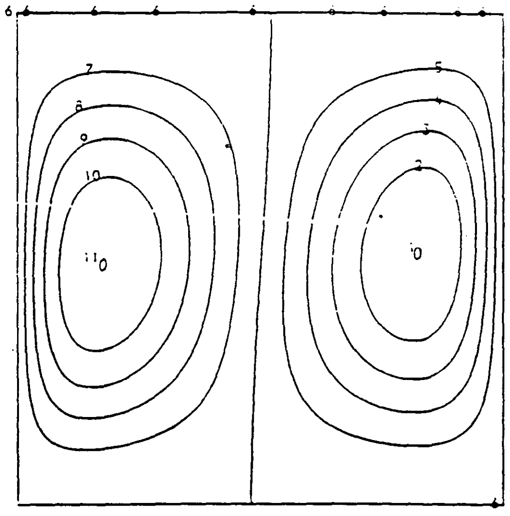
-43.59(8.719)43.59



(d) Ra = 10^6; contours at

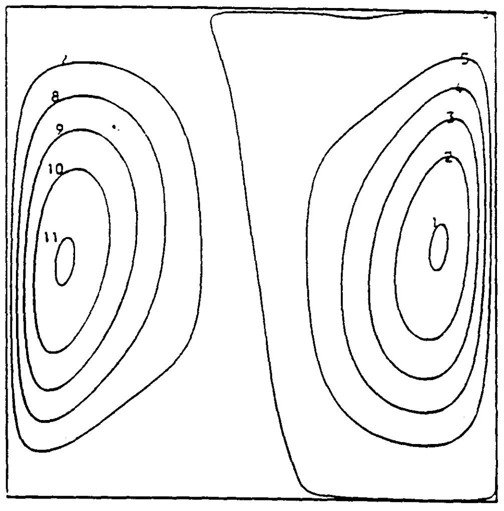
-125.5(25.10)125.5

Fig. 5.10 Contour Maps of Horizontal Velocity u



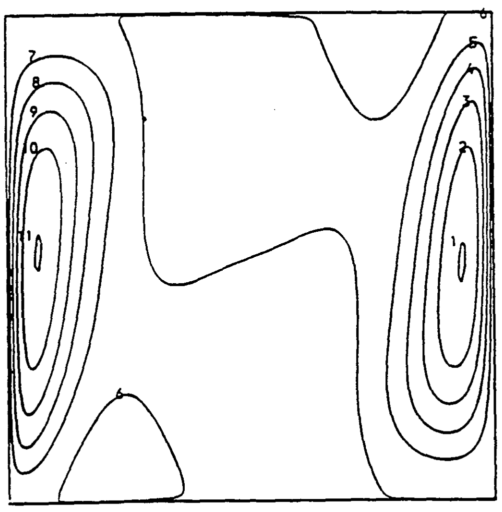
(a) Ra = 10^3; contours at

-3.663(0.7327)3.663



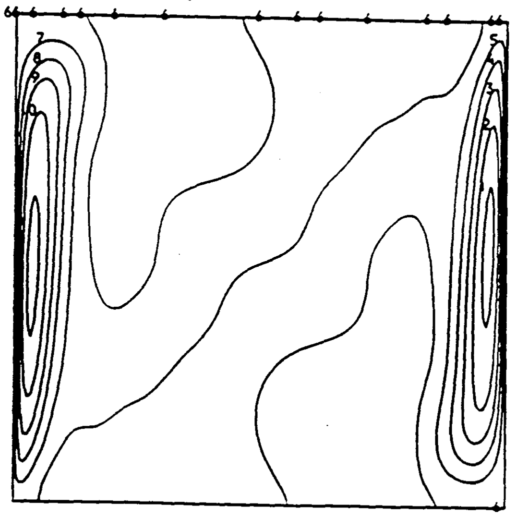
(b) Ra = 10^4; contours at

-19.39(3.877)19.39



(c) Ra = 10^5; contours at

-67.96(13.59)67.96



(d) Ra = 10^6; contours at

-207.6(41.52)207.6

Fig. 5.11 Contour Maps of Vertical Velocity v

5.3.3 The Flow past a Circular Cylinder

For the flow past a circular cylinder, using the following transformation, the physical domain can be mapped into a rectangular domain

$$x = e^{\eta}\cdot\cos\xi , \qquad y = e^{\eta}\cdot\sin\xi \tag{5.27}$$

where the function e^{η} assures an appropriately clustered grid point distribution close to the cylinder surface. The governing equations for this problem are (2.37), (2.38) shown in Chapter Two. To avoid having to deal with the large values of ψ occurring in the far field and also to facilitate the numerical implementation of the far field boundary conditions, the stream function ψ is decomposed into two parts such as

$$\psi = \psi_{in} + \bar{\psi}$$

where, ψ_{in} is chosen as the value of the inviscid flow, i.e.

$$\psi_{in} = (e^{\eta} - e^{-\eta})\cdot\sin\xi$$

Thus the governing equations can be written as

$$e^{2\eta}\cdot\omega_t + (\bar{\psi}_{\xi} + v_1)\cdot\omega_{\eta} - (\bar{\psi}_{\eta} + u_1)\cdot\omega_{\xi} = 2(\omega_{\xi\xi} + \omega_{\eta\eta})/\mathrm{Re} \tag{5.28}$$

$$\bar{\psi}_{\xi\xi} + \bar{\psi}_{\eta\eta} = e^{2\eta}\cdot\omega \tag{5.29}$$

with

$$u_1 = (e^{\eta} + e^{-\eta})\cdot\sin\xi$$

$$v_1 = (e^{\eta} - e^{-\eta})\cdot\cos\xi$$

On the surface of the body, the no-slip boundary conditions are

$$\bar{\psi} = 0 , \quad \bar{\psi}_{\eta} = -2\sin\xi \qquad \text{on } \eta = 0 \tag{5.30a}$$

$$\omega = \bar{\psi}_{\eta\eta} \qquad \text{on } \eta = 0 \tag{5.30b}$$

and the boundary conditions at infinity become

$$\bar{\psi} = 0 , \quad \bar{\psi}_{\eta} = 0 \qquad \text{on } \eta = \infty \tag{5.31a}$$

$$\omega = e^{-2\eta}\cdot\bar{\psi}_{\eta\eta} \qquad \text{on } \eta = \infty . \tag{5.31b}$$

For numerical simulation, the infinite boundary in the η direction can be truncated to a finite distance which is far enough from the cylinder to allow the far boundary conditions to be satisfied accurately. For steady state resolution of the problem, the most sensitive parameter to check the accuracy of numerical simulation is the calculation of the parameters defining the structure of the wake behind the cylinder. The cylinder and the geometrical parameters of the closed wake is shown in Fig. 5.12. After discretization by GDQ, the reduced set of (N-2)×(M-2) ordinary differential equations for ω are solved by the 4-stage Runge-Kutta scheme, and the set of (N-2)×(M-4) algebraic equations are solved by LU decomposition as before.

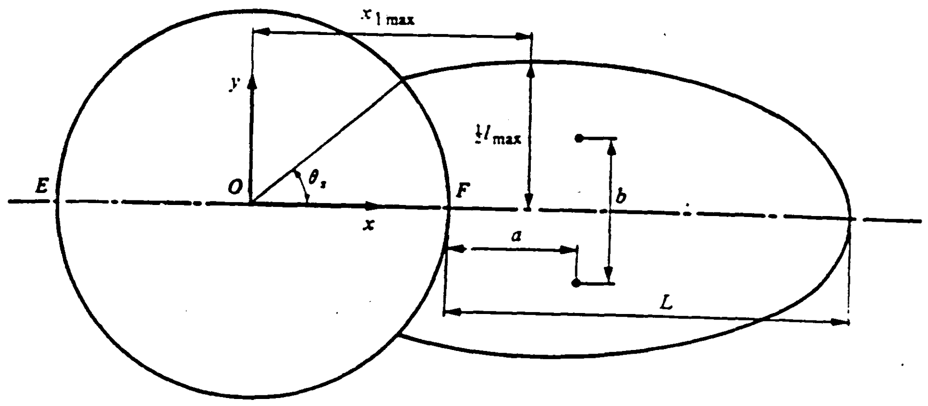
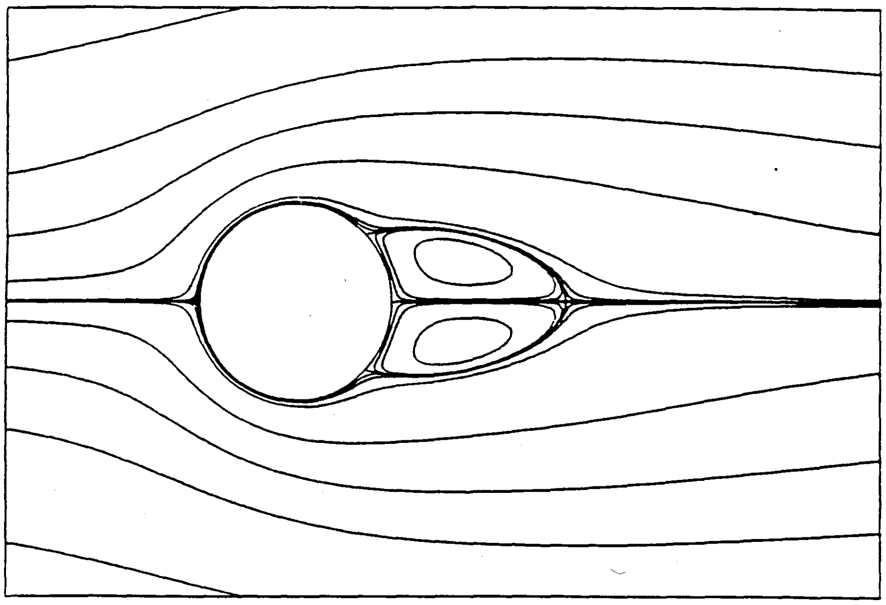
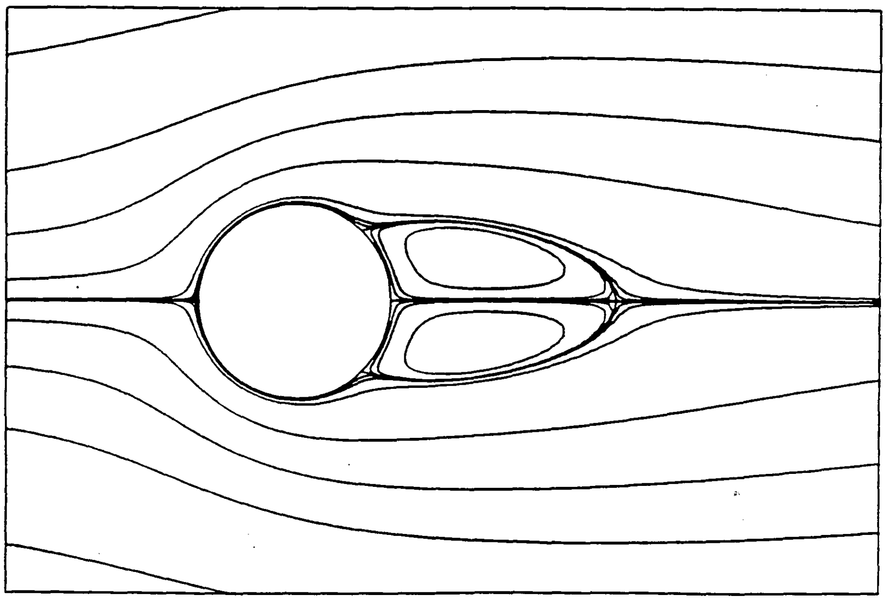


Fig. 5.12 Geometrical parameters of the closed wake behind cylinder

For the numerical simulation here, the $\eta_{\max}$ is chosen as 3.0, and the grid of case IV is used. It is known that the accurate simulation of the flow past a circular cylinder has demonstrated sensitivity in the imposition of the boundary conditions. The key factors may be the implementation of reasonable conditions at the far field boundary and the boundary conditions at the surface of the cylinder. In the present



(a) Re = 20



(b) Re = 25

Fig. 5.13 Streamlines Past A Circular Cylinder

computation, the Neumann boundary conditions for ω and ψ on the surface of the cylinder were treated with (M-1)th order accuracy, where M is the total number of grid points in the η direction. On the outer boundary, the inviscid flow (u=1, v=0) was assumed to provide two boundary conditions for $\bar{\psi}$, where the Neumann boundary condition was treated with (M-1)th order accuracy, and the boundary condition for ω was examined by two cases: one is to assume the outer boundary being in the inviscid region which yields $\omega=0$; another is to compute ω from the definition $\omega = u_y - v_x$, which is discretized with (M-1)th order accuracy. Numerical results for Re of 20, 25 show that both cases demonstrate nearly the same solutions. This further demonstrates that the outer boundary is in the inviscid region for these low Reynolds numbers. For the steady state resolution, the treatment of the boundary condition along the cut line (from the rear point of the cylinder to the outer boundary) was examined using two cases. One is to use the symmetric boundary conditions, namely, $\psi=0$, $\omega=0$, the other is to use the patching technique which enforces ω, ψ and their first derivatives with respect to the normal direction of the cut line to be continuous. Numerical experiment showed that both cases achieve nearly the same results but require different time steps for satisfying the given convergence criterion. Recommended is the use of $\omega=0$, $\psi=0$ at the cut line since this requires less time steps without losing accuracy. Fig. 5.13 shows the streamlines for Re = 20, 25, the values of the streamlines being ±3.0, ±2.0, ±1.0, ±0.5, ±0.15, $\pm5.0\times10^{-3}$, $\pm5.0\times10^{-4}$, $\pm1.0\times10^{-4}$, 0.0. The symmetric eddy pair is clearly shown in the Figure. For the results of Fig. 5.13, the outer boundary condition was set to the value of inviscid flow, the boundary condition on the cut line was set to be $\psi=0$, $\omega=0$, and the mesh size used is 25×21.

Table VIII Geometric Parameters of the Closed Wake behind A Cylinder

Re	Reference	L	a	b	x_{1max}	l_{max}	θ_s	C_D
20	experiment*	0.93	0.33	0.47	0.66	0.80	44.8^0	2.1243
	present	0.92	0.352	0.41	0.68	0.74	43.7^0	2.1220
	Dennis et al	0.94					43.7^0	2.0450
25	experiment*	1.21	0.44	0.51	0.75	0.85	48.0^0	1.8176
	present	1.21	0.424	0.475	0.73	0.82	46.6^0	1.8336
	Gresho et al	1.15	038	0.47	0.67	0.81	43.7^0	2.2600

Table VIII gives the details of the parameters of the wake eddy pair. Also included in Table VIII are the experimental data (Tritton 1959, Coutanceau and Bouard 1977) and other numerical results (Gresho et al 1984, Dennis and Chang 1970). It is shown from Table VIII that the present results are closer to the experimental data than those of Gresho et al although these authors put the outer boundary further away from the surface than the present work and use a larger number of grid points. The present results were thus more accurate than other numerical results even though the outer boundary was closer to the cylinder surface and fewer grid points were used. It is seen that on the one hand, GDQ appears to be a robust, efficient numerical technique, and on the other hand, the treatment of the boundary condition on the surface of the cylinder may be critically important in numerical simulation. The

* —— C_D (drag coefficient) is from the work of Tritton, other parameters are from the work of Coutanceau and Bouard with $\lambda=0$, where λ is the ratio between the cylinder and the tank diameter

major difference between the present approach and other numerical approaches is the treatment of the Neumann boundary conditions, with high order accuracy in the present approach and low order accuracy in other approaches.

5.3.4 The Flow past A Backward Facing Step

The flow past a backward facing step in a channel, shown in Fig. 5.14, is a challenging problem which had been chosen by the organizers of a GAMM workshop (Morgan et al 1984) as a test case for validating numerous solutions of the incompressible N-S equations. The governing equation for this problem is the same as for the driven cavity flow, but with different boundary conditions. As studied by other researchers (Thomas et al 1981), the computational domain in this case is limited to the interior of the channel immediately to the right of the step. A fully developed parabolic velocity profile (Couette flow) is used as the

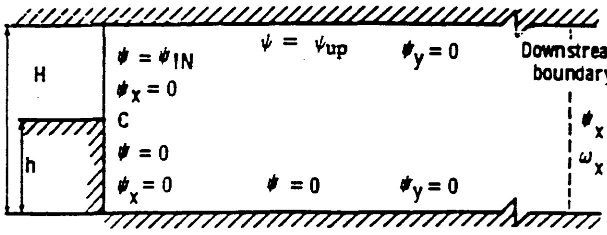


Fig. 5.14 Problem Definition of the Flow past A Backward Facing Step

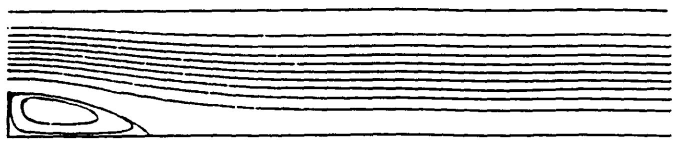
boundary condition in the upper part of the left boundary, the downstream boundary condition, being set at a distance from the step equal to 10, is implemented by the natural channel condition (zero gradient). For the case here, the non-dimensional height of the channel H is equal to 1.5, the height of the step is 0.5, and the maximum value of the non-dimensional longitudinal velocity at the inlet is equal to 1.0. Thus the boundary conditions can be written as

$\psi = 4(0.5\cdot y^2 - y^3/3)$, $\psi_x = 0$, at the inlet

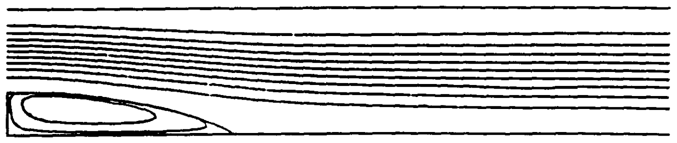
$\psi_x = 0$, $\omega_x = 0$, at the outlet

$\psi = 0$, $\psi_n = 0$, on the wall

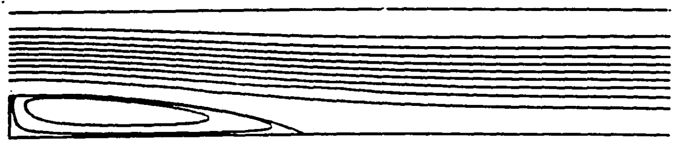
with 7 boundary conditions for ψ , 3 of which are in the x direction and 4 in the y direction, and 4 boundary conditions for ω. The resultant set of $(N-2)\times(M-2)$ ordinary differential equations for ω are solved by the 4-stage Runge-Kutta scheme, and the set of $(N-3)\times(M-4)$ algebraic equations for ψ are solved by LU decomposition using the same approach as for the driven cavity flow. For the numerical simulation, the sharp corner of the step (point C in Fig. 5.14) was chosen as the origin of the coordinate system, and the grid size used was 23×19. Numerical results for Reynolds number range from 50 to 450 were obtained. Figure 5.15 shows the streamlines for different Reynolds numbers, where the values of the streamlines (ψ/ψ_{max}) are 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0, -0.1, -0.01, -0.001 and the window in the x direction for plotting these streamlines is from x = 0.0 to x = 8.0. Figure 5.16, 5.17 show the vorticity distributions for different Reynolds numbers along the lower and upper walls. The dashed line included in these figures are the results which can be obtained theoretically when the downstream boundary is far enough so that the



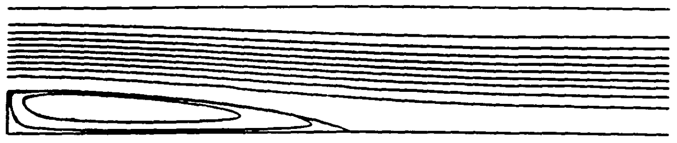
(a) Re = 100



(b) Re = 200



(c) Re = 300



(d) Re = 400

Fig. 5.15 Streamlines past A Backward Facing Step

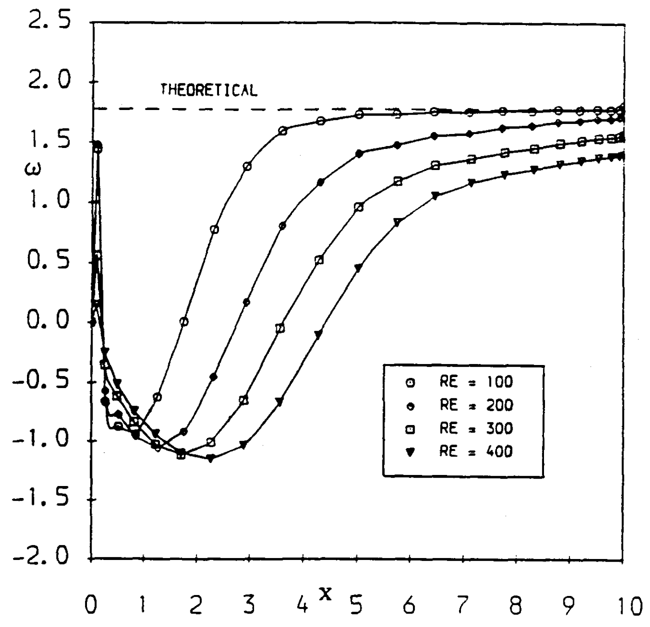


Fig. 5.16 Vorticity Distributions on the Lower Wall

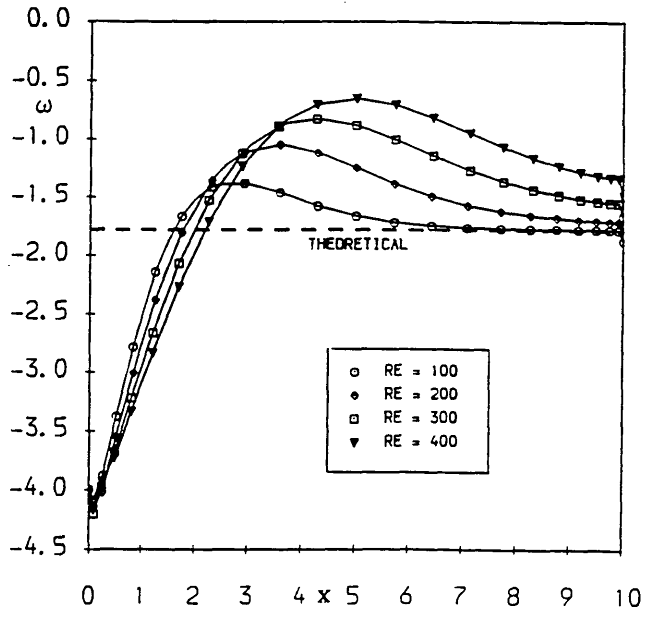


Fig. 5.17 Vorticity Distributions on the Upper Wall

fully developed parabolic velocity profile is obtained. From Fig. 5.16, 5.17, one can see clearly that the outflow boundary conditions have not been imposed far enough downstream except for a Reynolds number of 100. It is, therefore, demonstrated that the outflow boundary conditions of Neumann type may be useful for reasonable solutions with a short distance downstream. It is also shown in Fig. 5.16 and 5.17 that the vorticity near the step gives some minor spurious results. This behaviour may be caused by the discontinuity of ψ_{yy} at the corner C. In the present calculation, the value of ψ_{yy} at the corner C, is evaluated analytically, with the point C being considered as the part of the inlet flow domain ($\psi_{yy} = 4$), and this is not true for the vertical wall of the step. Thus on the boundaries of the computational domain, there is a point of singularity, namely C, which is not at a corner of the computational domain. On the other hand, GDQ is indeed a global method, which makes the variables at all interior grid points to be related to the variables on the boundary. In all cases computed, no convergence problems were encountered, but a small wiggle of the vorticity isolines near the step was observed. Also observed in Fig. 5.16 and 5.17 is that, as Reynolds number increases, the spurious values were weakened. The reason may be that a larger amount of diffusion is produced due to the sharp vorticity gradient around the corner, and as the Reynolds number increases, the convective term plays a more important role in the flow field than the diffusive term. Thus the influence of the point C on the results is weakened. Fig. 5.18 shows the reattachment length of the primary vortex vs Reynolds number. Some experimental results (Kueny and Binder 1984) were also included in Fig. 5.18 for comparison.

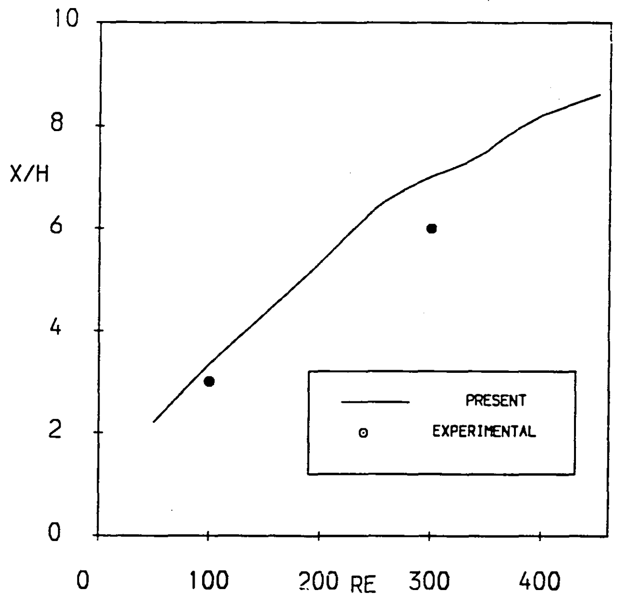


Fig. 5.18 Length of Recirculation Zone vs Reynolds Number

5.4 Multi-Domain Results

It is shown in Chapter Four that, as the number of grid points increases, the eigenvalues of the spatial discretization matrix, obtained by GDQ, increase very quickly. Thus the allowable time step size becomes very small for a large number of grid points and needs many time steps for converged results. On the other hand, GDQ generally requires the computational domain to be rectangular in the same way as for the spectral method, but in practical applications, the physical domain is usually complex, leading to difficulties in numerical simulations. These difficulties can be alleviated by the choice of grid generation and multi-domain techniques. In addition, a multi-domain technique is suited to the case in which there are geometrical singularities such as corners and sharp edges as tackled in the following simulation or the case where the computational domain may be divided into several regions described by different differential equations, e.g. the viscous region near the surface of a solid body is given by the N-S equations, and the inviscid region far from the solid boundary by the Euler equations. This section is devoted to the presentation and application of a multi-domain GDQ technique for solving the incompressible N-S equations in vorticity-stream function formulation. This approach combines the geometric capabilities of the multi-domain technique with the potential for accuracy of GDQ.

5.4.1 The Concept of Domain Decomposition

The physical domain of the problem can be represented by Ω, and the boundary by Γ. The multi-domain technique, firstly, decomposes the domain Ω into several subdomains Ω_i, $i = 1, 2, \cdots, K$, where K is the number of the subdomains. In each subdomain, a local mesh can be generated with stretching near the boundaries and a local GDQ technique can be used in the same fashion as the application of GDQ in a single domain. In the case of solving the incompressible N-S equations, this means that the resultant ordinary differential equations for vorticity and the algebraic equations for stream function are to be solved in each subdomain. Each subdomain may have a different number of grid points. The solutions for interior grid points are independent for each subdomain, thus they can easily be computed in parallel. Globally, the information exchange between the subdomains is required. This can be done across the interface of the subdomains. Since any complex geometry can be transformed into a rectangular domain or a combination of the rectangular subdomains by the technique of grid generation. Here we only consider the rectangular domain for demonstration without losing generality.

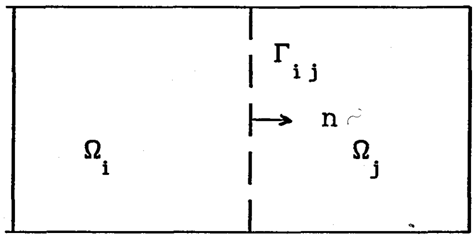


Fig. 5.19

Supposing Γ_{ij} is the interface of the subdomains Ω_i and Ω_j, that is, $\Gamma_{ij} = \Omega_i \cap \Omega_j$. The patching condition is enforced at the interface Γ_{ij} so that both the function and its first derivative normal to Γ_{ij} are continuous along the normal direction of the interface, i.e.

$$f(x_N^i) = f(x_1^j) \qquad \text{on } \Gamma_{ij} \tag{5.32}$$

$$f_n(x_N^i) = f_n(x_1^j) \qquad \text{on } \Gamma_{ij} \tag{5.33}$$

where $f(x_N^i)$, $f(x_1^j)$ represent the values of the function f at the interface of the subdomains Ω_i and Ω_j, and $f_n(x_N^i)$, $f_n(x_1^j)$ the values of the derivative of f with respect to n at the interface. For the cases selected for study, each subdomain is rectangular. Then the normal direction to the interface is parallel to one coordinate axis in the local coordinate system. For simplicity, this coordinate axis can be assumed as the x axis, and in this direction, there are N grid points in the subdomain Ω_i, and M grid points in the subdomain Ω_j. The weighting coefficients of the first order derivative along the x direction are written as a_{mn}^i in the Ω_i and a_{mn}^j in the Ω_j. Thus using the technique of GDQ, (5.33) can be written as

$$\sum_{k=1}^{N} a_{Nk}^i \cdot f(x_k^i) = \sum_{k=1}^{M} a_{1k}^j \cdot f(x_k^j) \tag{5.34}$$

Using (5.32), and setting $f(x_N^i) = f(x_1^j) = \bar{f}$, we obtain

$$\bar{f} = (\sum_{k=1}^{N-1} a_{Nk}^i \cdot f(x_k^i) - \sum_{k=2}^{M} a_{1k}^j \cdot f(x_k^j))/(a_{11}^j - a_{NN}^i) \tag{5.35}$$

where $\bar{f}$ is the value of the approximation to the function f at the interface Γ_{ij}, which exchanges the information between the subdomains, and the $f(x_k^i)$, $f(x_k^j)$ represent the values of the function f at x_k^i in the subdomain Ω_i and x_k^j in the subdomain Ω_j. For the solution of the incompressible N-S equations in the vorticity-stream function formulation, $\bar{f}$ can be the vorticity and the stream function, and (5.35) is used as the Dirichlet boundary condition for them. Equation (5.35) is suitable for rectangular domains. If the computational domain is a non-rectangular domain, it should be transformed into several rectangular subdomains firstly, then (5.35) can be used.

5.4.2 The Flow past A Backward Facing Step

This problem is the same as that described in the section 5.2.4 with a difference that the the inlet is located at a distance upstream of the step. For numerical simulation here, the computational domain is divided into 3 subdomains. The problem definition and the computational domain are shown in Fig. 5.20, where the expansion ratio is 1:1.5. All the lengths have been normalized by the inlet width D, and the velocities by the maximum value of the longitudinal velocity at inlet, u_{max}. The location of the inlet is chosen as 6D upstream of the step, and the outlet is located at 12D downstream of the step. The implementation of the boundary condition can be treated using the same method as in the section 5.2.4. In each subdomain, one Dirichlet boundary condition for vorticity was used on each boundary and the interface, but for stream function, two boundary conditions (one being of Dirichlet type, another of Neumann type) on each boundary, and one Dirichlet boundary condition on the interface, were used. To verify that the solution is independent of grid, different mesh sizes have been tested. Numerical experiment shows that this case can be accurately simulated using few grid points. The length of recirculation zone computed by the mesh size of 15×13 for domain I, 23×13 for domain II, and 23×11 for domain III is less than

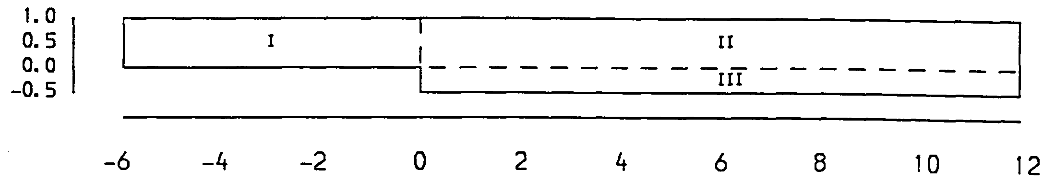


Fig. 5.20 Problem Definition and the Computational Domain

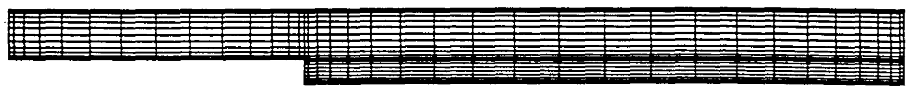
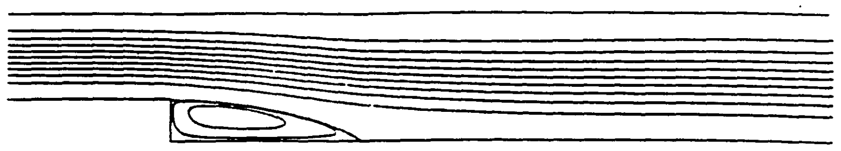


Fig. 5.21 Meshes for Flow past A Backward Facing Step

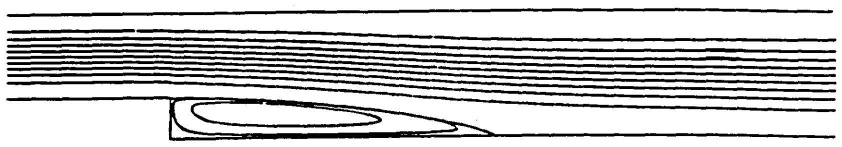
three per cent different from that computed by the mesh size of 15×17 for domain I, 23×17 for domain II, and 23×19 for domain III for all Reynolds numbers. In this section, all the results are based on the mesh size of 15×13 for domain I, 23×13 for domain II, and 23×11 for domain III, which is shown in Fig. 5.21.

As stated in the section 5.2.4, the sharp corner of the step is a singularity, and since this point is on the boundary, not at the corner, of the computational domain, small spurious deviations were produced near the step. On the other hand, the computation for Reynolds number over 500 becomes difficult because of the effect of the singularity. The singularity can be removed by techniques such as the interpolation method, asymptotic expansion, etc. For the case here, the multi-domain technique was used to avoid dealing with the sharp corner singularity since the sharp corner of the step is exactly at the corner of the subdomain.

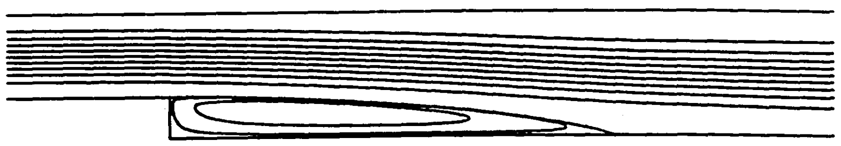
Numerical results for Reynolds numbers range from 100 to 1000 were obtained. Fig. 5.22 illustrates the computed streamlines for Reynolds numbers of 200, 400, 600, 800, 1000, where the values of the streamlines are the same as shown in the section 5.2.4, and the window for plotting the streamlines in the x direction is from x = -2.0 to x = 8.0. Fig. 5.23, 5.24 show the vorticity distributions for different Reynolds numbers along the lower and upper walls. Fig. 5.25 and 5.26 compare the single-domain results with the multi-domain results for the vorticity distributions along the walls. The dashed lines included in these figures represent the results of the fully developed parabolic profile which would be obtained if the downstream boundary is located at an infinite distance from the step. From these figures, it is clear that no spurious deviations in the vicinity of the step were found for the multi-domain results; the single domain results differ from the multi-domain results near the step because of the effect of the singularity, but in the downstream, the single domain results agree well with the multi-domain results; the flow does not become fully developed at the high Reynolds number cases. Fig. 5.27 shows the horizontal velocity profiles for Reynolds number of 100 and 900 at the step. The dashed line



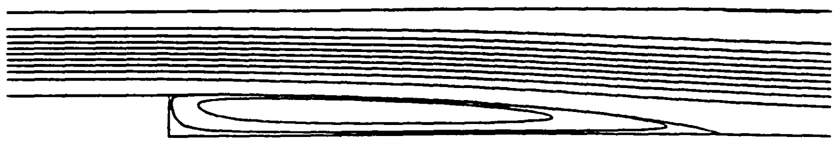
(a) Re = 200



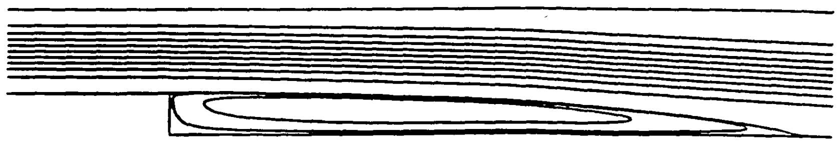
(b) Re = 400



(c) Re = 600



(d) Re = 800



(e) Re = 1000

Fig. 5.22 Streamlines of the Flow past A Backward Facing Step

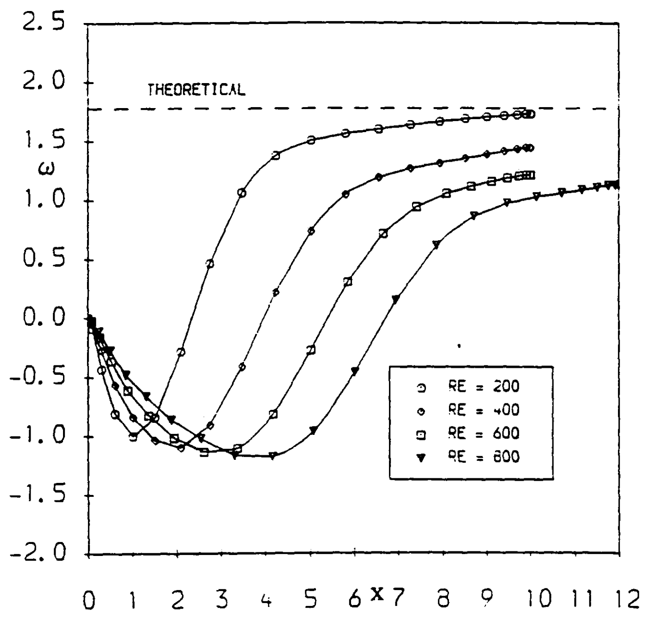


Fig. 5.23 Vorticity along the Lower Wall (Multi-Domain Results)

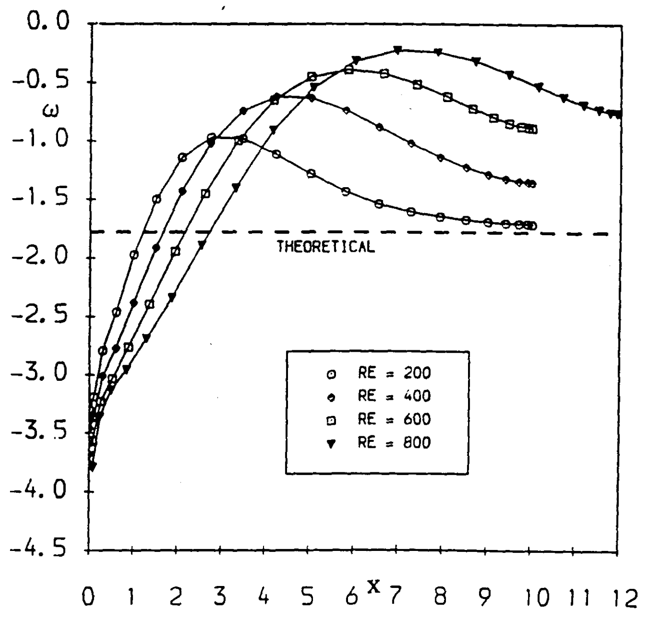


Fig. 5.24 Vorticity along the Upper Wall (Multi-Domain Results)

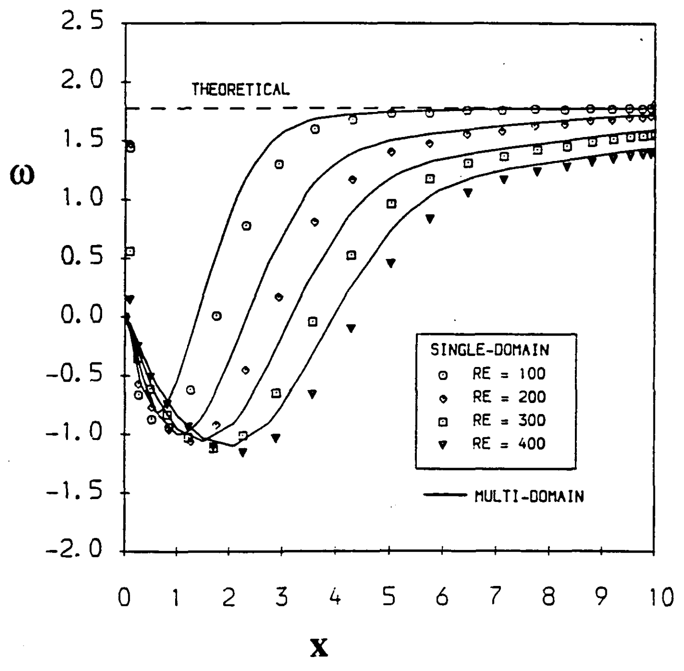


Fig.5.25 Comparison of Single-Domain Results with Multi-Domain Results

Vorticity along the Lower Wall

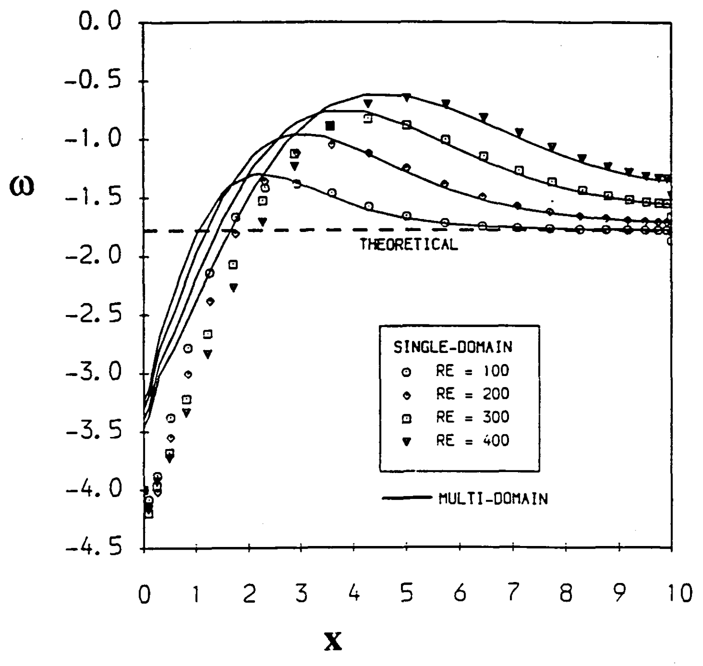


Fig.5.26 Comparison of Single-Domain Results with Multi-Domain Results

Vorticity along the Upper Wall

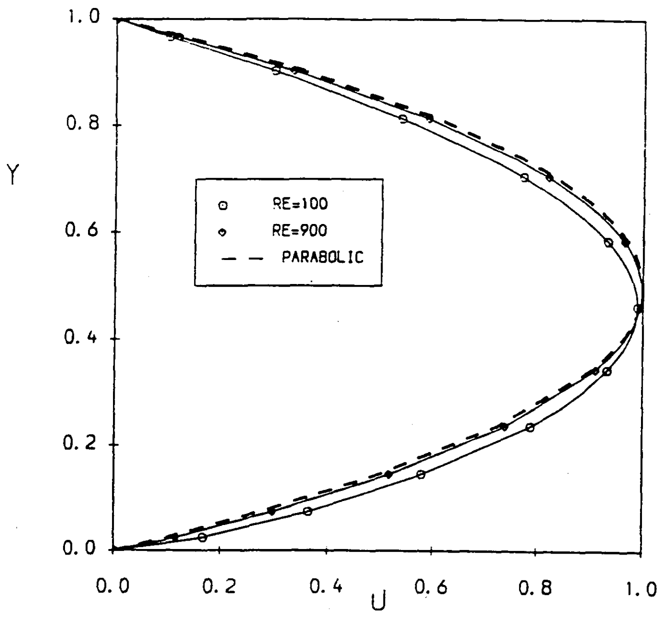


Fig. 5.27 Horizontal Velocity Profiles at the Step

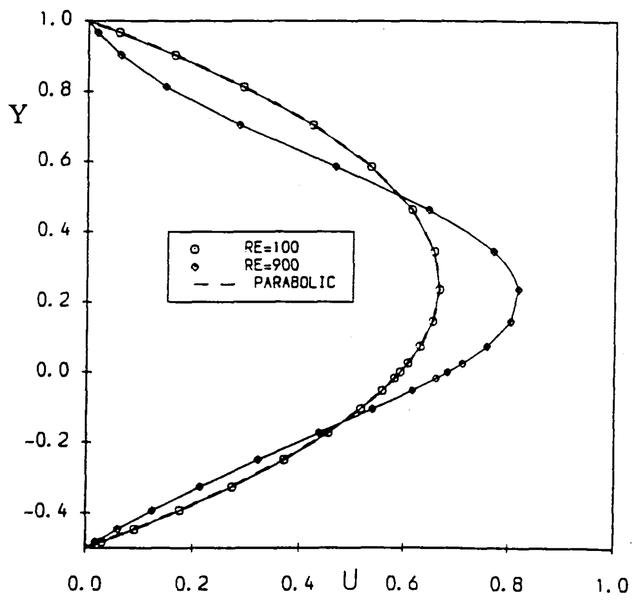


Fig. 5.28 Horizontal Velocity Profiles at the Outlet

in this figure represents the parabolic velocity profile imposed at the inlet. It is seen that the velocity profile for the low Reynolds number case was close to the parabolic profile, except for a systematical small deviation due to the pressure gradient enforced by the step. However, the velocity profile at the high Reynolds number case tended towards the parabolic profile due to the convective term playing a greater role in the flow field. Thus it is suggested that, for accuracy, the inlet should be a reasonable distance before the step for low Reynolds numbers, but can be imposed at the step for high Reynolds numbers. Figure 5.28 shows the velocity profiles at the outlet compared with the fully developed parabolic profile represented again by the dashed line. It is, again, shown that at high Reynolds number the velocity profile at

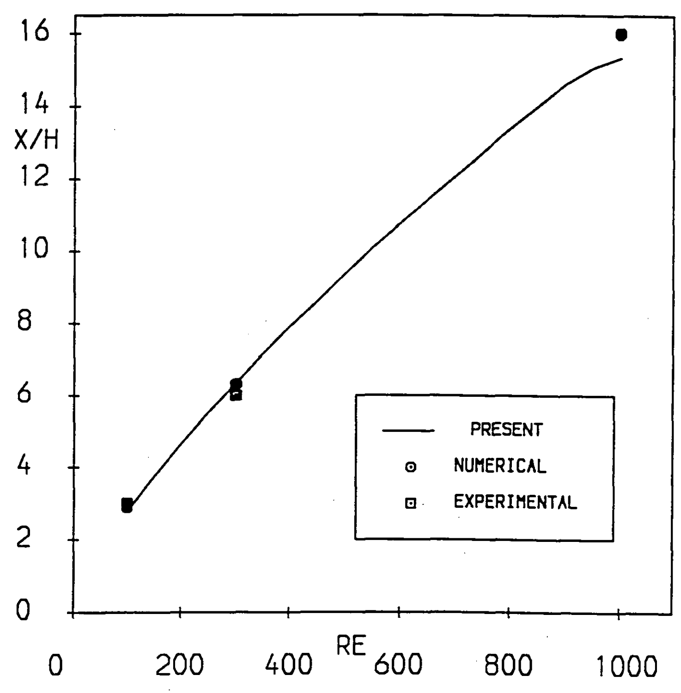


Fig. 5.29 Length of Recirculation Zone vs Reynolds Number

the outlet is not fully developed. This demonstrates that the outflow boundary condition of the Neumann type may give reasonable solutions for a short distance downstream. Figure 5.29 gives the length of the recirculation zone for different Reynolds numbers compared with the experimental data (Kueny and Binder 1984) and other numerical results (Bredif 1984). The present results agree well with the experimental data.

5.4.3 The Flow past A Square Step

Now considered is the flow in a channel containing a square step in which the step is located fairly close to the inlet. The flow past a square step with a "flat" inlet velocity distribution rather than a fully developed parabolic profile, is a more challenging problem for numerical simulation since in this case, not only the two sharp corners of the step produce vorticity singularities, but also the boundary condition at the inlet introduces other vorticity singularities. Hughes et al (1979) presented several results using FEM and claimed that the conventional Galerkin method produced spurious wiggles in the velocity vectors upstream of the step. They suggested an upwind method which then generated the solution without wiggles. Leone and Gresho (1981) studied this problem exhaustively using a velocity-pressure formulation and the conventional Galerkin method, and claimed that the spurious wiggles of the solution may be caused by a combination of the following factors: (1) too coarse a grid to resolve the steep gradient occurring in the flow direction; (2) inlet boundary conditions and the resulting leading edge singularities; (3) proximity of the inlet region to the step; (4) the sharp edge singularity at the leading corner of the step. They firstly studied Stokes flow and found that the inlet wiggles are caused by the leading edge singularities (high pressures are generated at the corners where the fluid decelerates and converges toward mid-channel). They then studied viscous flows through the N-S equations and claimed that, when a coarse mesh is used the inlet wiggles may be caused more by the presence of the step than the leading edge singularity, and when the finer mesh is used, most of the inlet wiggles disappear, only small deviations appearing near the top singularity of the inlet leading edge. They thus suggested that this difficult problem should be solved on a fine grid. Also some results exist in the work of Yang and Atluri (1984) where a mixed finite element method is used.

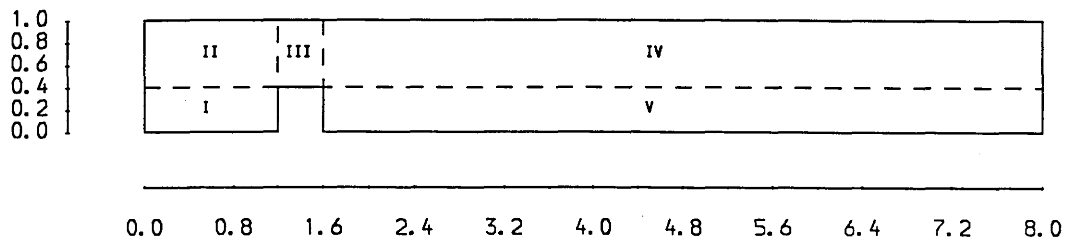


Fig. 5.30 Square Step Problem Definition and Computational Domain

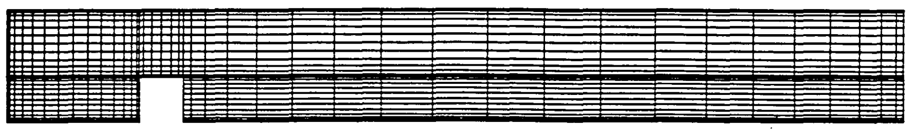


Fig. 5.31 Meshes for Flow past A Square Step

Following the work of Hughes et al, it is attempted to simulate the developing flow in a one unit high (the characteristic length for defining the Reynolds number) channel containing a square step located at 1.2 units from the inlet which is 0.4 units high and 0.4 units across. The problem definition and the computational domain are shown in Fig. 5.30, where the whole domain is decomposed into 5 subdomains with 4 interfaces. The inlet boundary condition is a "flat" velocity profile, u=1 and v=0, except that the no-slip condition, u=0, occurs on the top and bottom surfaces, which gives

$$\psi = y\ ,\quad \psi_x = 0, \qquad \text{at the inlet} \tag{5.36}$$

and the boundary condition

$$\psi_x = 0\ ,\quad \omega_x = 0 \tag{5.37}$$

is imposed at the outlet. On the walls and the surface of the step, the no-slip boundary condition gives

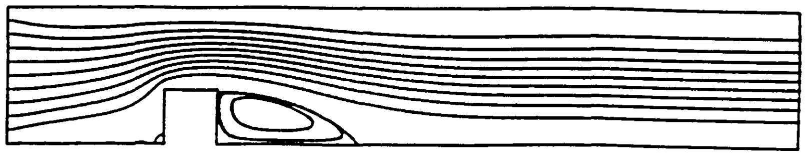
$$\psi = \begin{cases} 1 & \text{on the upper wall} \\ 0 & \text{others} \end{cases} \tag{5.38a}$$

$$\psi_n = 0, \text{ where n is normal to the surface} \tag{5.38b}$$

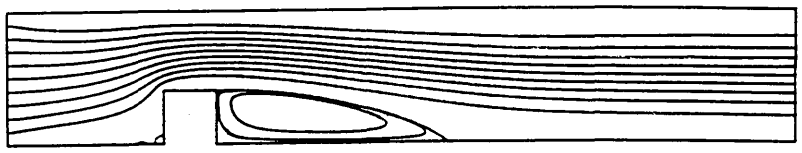
For the present calculation, the outlet location is chosen as 8 units from the inlet. Numerical experiment shows that the accurate results can be obtained by using the mesh sizes of 15×13 for domain I and II, 7×13 for domain III, 21×13 for domain IV and V. This configuration is shown in Figure 5.31. Using the same approach as for the backward facing step, the multi-domain solutions of this problem for Reynolds numbers range from 50 to 250 were obtained. Fig. 5.32 shows the streamlines for Reynolds numbers of 50, 100, 150, 200, 250, where the values of the streamlines are 1.0, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0, -0.1, -0.01, -0.001 and the window for plotting these streamlines in the x direction is from x=0.0 to x=6.0. Clearly, it is shown that no wiggles appear in the flow field except for very small deviations caused by the top singularity of the channel leading edge, which appear near the top corner of the inlet (streamlines have a small contraction near mid-channel). This agrees well with the analysis of Leone et al and demonstrates that the mesh sizes used are fine enough to get accurate results. Fig. 5.33 shows the vorticity distributions along the lower wall before the step for different Reynolds numbers. The plots display clearly a large influence of the lower singularity of the leading edge on the flow near the inlet. In the region close to the bottom corner of the inlet, the flow is dominated by the high pressure gradient produced mainly by the singularity rather than by viscosity, since in this region, the vorticity is independent of the Reynolds number. Fig. 5.34 shows the vorticity distribution along the lower wall behind the step for different Reynolds numbers. The dashed line included in this figure is the result of the fully developed parabolic profile which would be obtained if the outlet is placed at an infinite distance from the inlet. One can see from Fig. 5.34 that most cases except for Re=50 do not achieve a fully developed parabolic velocity profile at the outlet. This demonstrates that the Neumann type boundary condition imposed at the outlet can provide reasonable solutions when the outlet is placed only a short distance downstream. Fig. 5.35 shows the vorticity distributions along the surface of the step. The two singularities at the sharp corners are shown clearly. Figure 5.36 displays the vorticity distributions along the upper wall for different Reynolds numbers. It demonstrates that the flow near the upper singularity of the inlet is dominated by the high pressure gradient, produced mainly by the singularity since the vorticity in this region is shown to be independent of the Reynolds number. This shows that the small deviations in streamlines occurring near the upper corner of the inlet is indeed caused by the singularity of the inlet leading edge. The dashed line included in Figure 5.36 is again the result of the fully developed parabolic profile. Fig. 5.37 gives the velocity profiles at the outlet for Reynolds number cases of 50 and 250. The plots show that the velocity for the low Reynolds number case achieves a nearly parabolic profile at the outlet. This is not the case for the high Reynolds number case. The lengths of the upstream and downstream separated zone for the various Reynolds numbers are shown in Table IX, where x_{up} and x_{do} represent the lengths of the upstream and downstream separation zones, and $\bar{x}_{up} = x_{up}/h$, $\bar{x}_{do} = x_{do}/h$, h is the height of the step.

Table IX Lengths of the Separation Zone for A Square Step Problem

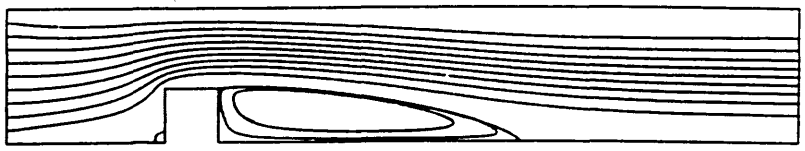
Re	25	50	85	100	150	200	250
$\bar{x}_{up}$	0.1749	0.1749	0.1749	0.1757	0.1777	0.5771	0.5846
$\bar{x}_{do}$	1.5771	2.6701	3.8876	4.3501	5.7549	7.0636	7.9824



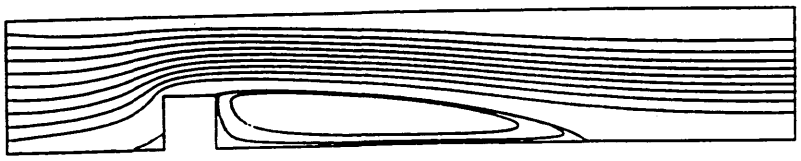
(a) Re = 50



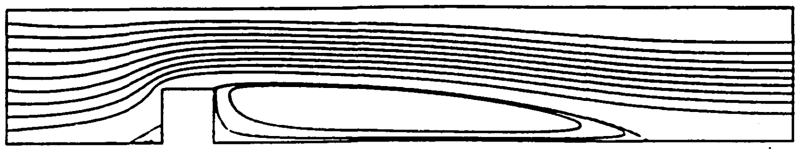
(b) Re = 100



(c) Re = 150



(d) Re = 200



(e) Re = 250

Fig. 5.32 Streamlines of the Flow past A Square Step

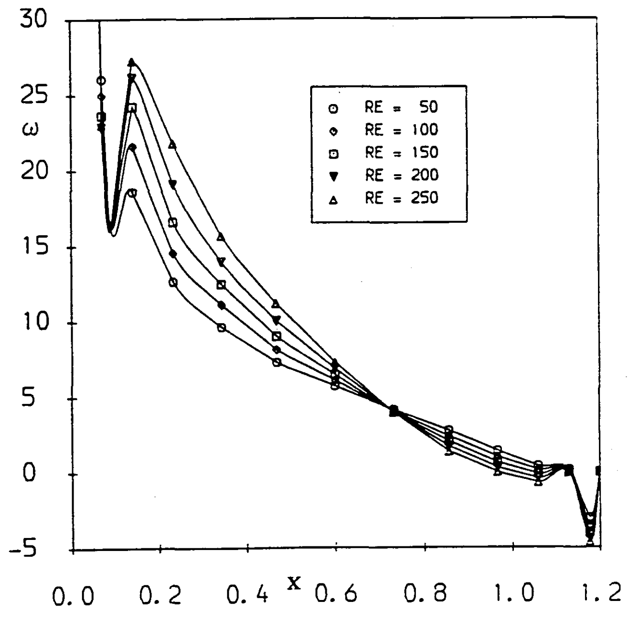


Fig. 5.33 Vorticity along the Lower Wall before the Square Step

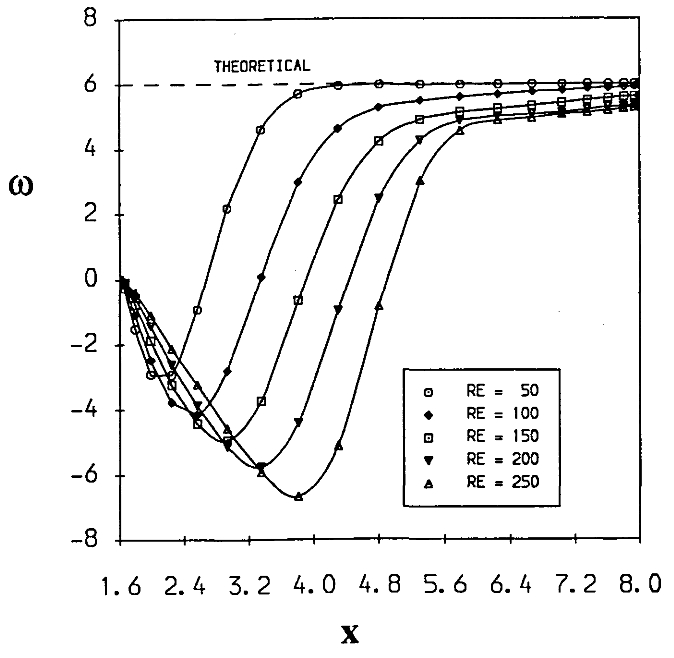


Fig. 5.34 Vorticity along the Lower Wall behind the Square Step

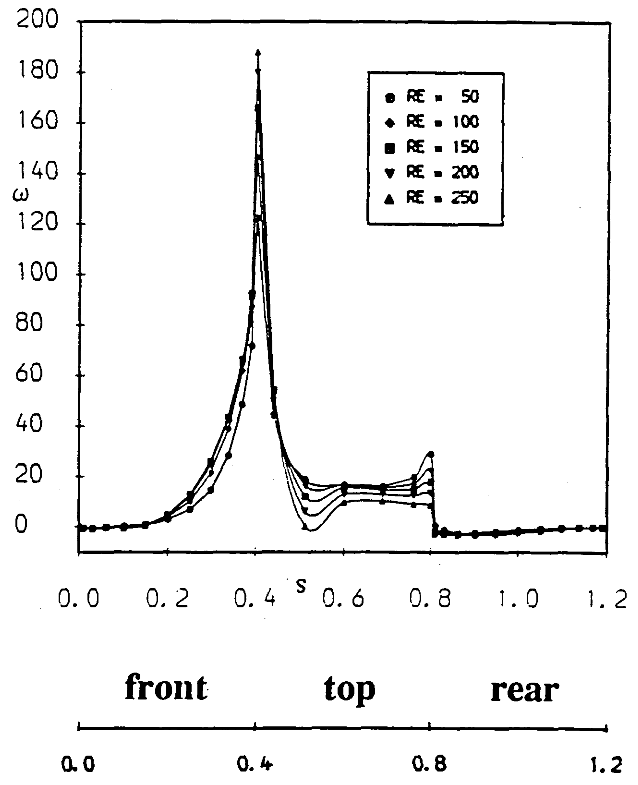


Fig. 5.35 Vorticity along the Surface of the Square Step

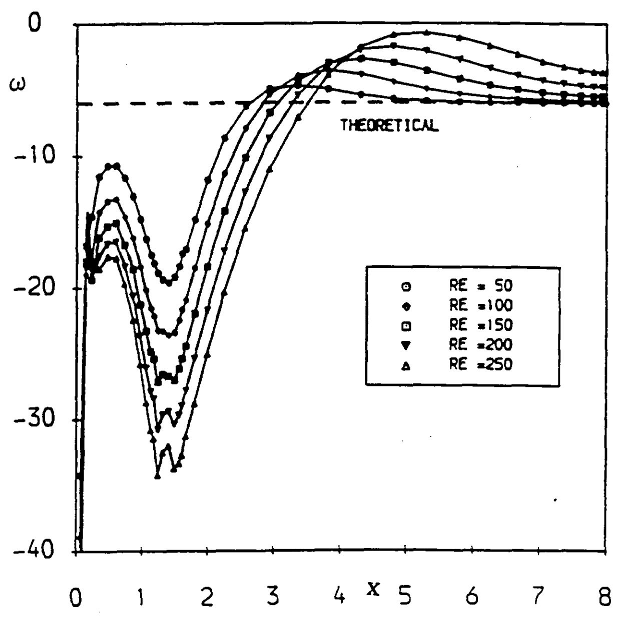


Fig. 5.36 Vorticity along the Upper Wall for Flow past A Square Step

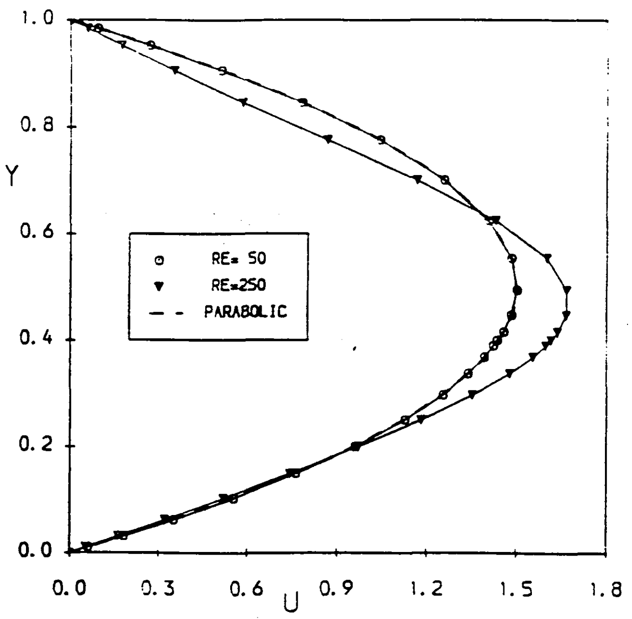


Fig. 5.37 Horizontal Velocity Profiles at the Outlet

5.5 Concluding Remarks

The GDQ and multi-domain GDQ techniques for the solution of the incompressible N-S equations have been shown in this chapter. Numerical results, obtained by GDQ using just a few grid points, are very accurate, and need less storage and computational time, compared with the conventional numerical techniques such as the finite difference methods using a large number of grid points. The reason for GDQ results needing much less CPU time are twofold. One is that the GDQ results are obtained using few grid points, thus fewer degrees of freedom are involved in the solution of resultant equation systems. Another is that GDQ is a global method, which has a global convergence. In other words, it needs less time steps for convergence.

CHAPTER SIX

SOLUTIONS OF BOUNDARY LAYER EQUATIONS

6.1 Introduction

The boundary layer approximation provides a useful mathematical model for some engineering problems which include the jet and wake flows, channel and tube flows and wall boundary layers. Another aspect of the boundary layer approximation currently of interest is the simulation of the flow with a small separation region which can be carried out by the concept of viscous-inviscid interaction, where the calculation of the viscous part is obtained by boundary layer computation. This approach can greatly reduce the computational effort compared with a Navier-Stokes solver. The classical numerical methods for the solution of the boundary layer equations can generally be categorized as (1) integral methods and (2) differential methods. The integral methods transform the boundary layer equations into ordinary differential equations by integrating the differential equations in the normal direction after making assumptions about the general form of the velocity and temperature profiles. The advantage of integral methods is their simplicity as well as small computational effort. But for a general problem, the application of integral methods is not as straightforward as for differential methods. Furthermore, the numerical results are very sensitive to the form of the velocity and temperature profiles in the three-dimensional case. Most schemes using differential methods involve finite differencing. The Crank-Nicolson finite difference scheme was found to work well for a general boundary layer calculation, and has been adopted by many researchers (Werle and Bertke, 1972, etc.). Among others, the Keller-box method (1974) for parabolic partial differential equations has been successfully applied by Keller and Cebeci (1970), for boundary layer resolution. For the unsteady or three-dimensional boundary layer calculation, if the flow has a separation region, the Zig-Zag-box scheme (Cebeci, 1979) should be used since the Keller-box scheme fails to work in this case. Recently, the application of finite element methodology (Chung, 1978), and the spectral method (Streett, Zang and Hussaini, 1984) to the boundary layer equations has been reported.

Boundary layer equations can be solved when they are expressed in physical coordinates or in transformed coordinates. Generally, the transformed form is favourable because it can remove the singularity occurring at the leading edge of the surface. Using transformed coordinates, most researchers prefer to use the stream function as the dependent variable. The major advantage of this is that the continuity equation can be dropped from the solution procedure. Accordingly, the order of the differential equations is increased by one, which may create difficulties in dealing with the boundary conditions. Some other researchers favour the use of the primitive-variable (velocity) as the dependent variable enabling 2D methods to be extended to the 3D case directly. The difficulty then is the coupling of the continuity equation with the momentum and energy equations. The reason for not using the integral form of the continuity equation is that the normal velocity obtained by integrating the equation along the normal coordinate is less accurate because of accumulated round-off errors.

As will be shown in this chapter, the GIQ technique can provide a promising way to obtain the normal velocity accurately by an explicit formulation derived from the integration of the continuity equation in the normal direction. As stated in Chapter 3, the integral over a part of the overall domain can be accurately approximated by a linear combination of all the functional values in the overall domain. Thus, the determination of the normal velocity at any mesh point involves all the functional information of the normal coordinate direction of that mesh point, and has the same order of accuracy for all mesh points along the normal direction. On the other hand, it has been shown in Chapter 5 that the GDQ scheme is a robust, efficient technique to discretize spatial derivatives, which can achieve results of high accuracy using just a few grid points. We will use both the GDQ and GIQ techniques in the normal direction for discretizing the derivatives and the integrals. The mesh points chosen for GDQ and GIQ are the same. In the streamwise or the crossflow direction, both GDQ and finite difference schemes can be used. We will show that the GDQ-GIQ technique for the boundary layer resolution is a general one, which can be used for both primitive variables and stream function or other functional variables.

6.2 GDQ-GIQ Approach

We will consider the GDQ-GIQ approach to the solution of the boundary layer equations using two cases of dependent variable. These are the stream function variable and the primitive variable. Although two specific test problems are chosen for demonstration, the GDQ-GIQ approach can be used for general boundary layer problems.

6.2.1 Stream Function Chosen As Dependent Variable

For simplicity, we consider the two-dimensional Howarth boundary layer problem. The governing equation (Keller and Cebeci, 1970) is

$$\frac{\partial^3 f}{\partial \eta^3} + f\cdot \frac{\partial^2 f}{\partial \eta^2} + \beta(\xi)\cdot\left[1-(\frac{\partial f}{\partial \eta})^2\right] = 2\xi\left[\frac{\partial f}{\partial \eta}\cdot \frac{\partial^2 f}{\partial \xi \partial \eta} - \frac{\partial^2 f}{\partial \eta^2}\cdot\frac{\partial f}{\partial \xi}\right] \tag{6.1}$$

where $f(\xi, \eta)$ is the dimensionless stream function and is subject to the boundary conditions

$$f(\xi,0) = \frac{\partial f}{\partial \eta}(\xi,0) = 0 \tag{6.2}$$

$$\frac{\partial f}{\partial \eta}(\xi,\eta) \longrightarrow 1, \text{ when } \eta \longrightarrow \infty \tag{6.3}$$

Here (ξ,η) are the Levy-Lees coordinates; ξ increases in the free stream direction and η increases away from the wall. And $\beta(\xi)$ is given by

$$\beta(\xi) = \frac{\xi}{\xi - 4}\,. \tag{6.4}$$

Setting $u = \frac{\partial f}{\partial \eta}$, then the 3rd order equation (6.1) can be reduced to

$$\frac{\partial^2 u}{\partial \eta^2} + f\cdot \frac{\partial u}{\partial \eta} + \beta(\xi)\cdot[1 - u^2] = 2\xi\left[u\cdot\frac{\partial u}{\partial \xi} - \frac{\partial u}{\partial \eta}\cdot\frac{\partial f}{\partial \xi}\right] \tag{6.5}$$

$$f = \int_0^{\eta} u\cdot d\eta + f(0) \tag{6.6}$$

For numerical simulation, the infinite interval in the η direction can be truncated to the finite interval $[0, \eta_\infty]$. Using GDQ and GIQ in the domain $[0, \eta_\infty]$, we have

$$(\frac{\partial^2 u}{\partial \eta^2})_{i,j} = \sum_{k=1}^{M} \bar{w}_{jk}^{(2)}\cdot u_{ik} \tag{6.7a}$$

$$(\frac{\partial u}{\partial \eta})_{i,j} = \sum_{k=1}^{M} \bar{w}_{jk}^{(1)}\cdot u_{ik} \tag{6.7b}$$

$$f_{i,j} = \sum_{k=1}^{M} (w_{jk}^{I} - w_{1k}^{I})\cdot u_{ik} + f_{i,1} \tag{6.7c}$$

where $\bar{w}_{ij}^{(m)}$ are the weighting coefficients of the *m*th order derivative of the function with respect to η, w_{ij}^{I} are the weighting coefficients of the integral along the η direction, and the boundary conditions (6.2), (6.3) become

$$u_{i,1} = 0 , \quad u_{i,M} = 1 , \quad f_{i,1} = 0 \tag{6.8}$$

which is easily implemented in the solution procedure. It is clear that there are two boundary conditions for u in the η direction for this case. As we will show, another boundary condition for u in the η direction can be implemented if it is necessary. Referring to equation (3.69), the discretization of (6.6) can also be written as

$$f_{i,j} = \sum_{k=1}^{M} w_{jk}^{I} \cdot u_{ik} - f(c) + f_{i,1} = \sum_{k=1}^{M} w_{jk}^{I} \cdot u_{ik} \tag{6.9}$$

where c is chosen as c = 0, thus $f(c) = f_{i,1} = 0$. Equation (6.9) can provide another boundary condition for u, i.e.

$$\sum_{k=1}^{M} w_{1k}^{I} \cdot u_{ik} = f_{i,1} = 0 \quad . \tag{6.10}$$

It is noticed that if (6.9) is used, then (6.10) should be implemented as another boundary condition for u.

If a second order finite difference scheme is used at $(\xi_{i-1/2} , \eta_j)$ for the discretization of the derivative of u or f with respect to ξ, after linearization of the non-linear terms, the resultant algebraic equations can be written as

$$\mathbf{A} \cdot \mathbf{U} = \mathbf{b} \tag{6.11}$$

where

$\mathbf{U} = (u_{i,2}, u_{i,3}, \cdots, u_{i,M-1})^T$ when two boundary conditions are used

or $\mathbf{U} = (u_{i,3}, u_{i,4}, \cdots, u_{i,M-1})^T$ when three boundary conditions are used.

A is a full matrix and **b** is a known vector. Using (6.11), the boundary layer solution in the whole domain can be obtained by a marching technique along the ξ direction.

If the GDQ scheme is used in the ξ direction, after linearization of the non-linear terms, we can get a similar form of the algebraic equation system to (6.11), but the vector **U** includes all the interior functional values $u_{i,j}$. Thus the marching technique cannot be used in this case. If only steady state resolution is of interest, we can turn to another method. Introducing an unsteady term in (6.5), we can write (6.5) as

$$\frac{\partial^2 u}{\partial \eta^2} + f\cdot \frac{\partial u}{\partial \eta} + \beta(\xi)\cdot[1 - u^2] = \frac{\partial u}{\partial t} + 2\xi[u\cdot\frac{\partial u}{\partial \xi} - \frac{\partial u}{\partial \eta}\cdot\frac{\partial f}{\partial \xi}] \ . \tag{6.12}$$

The spatial discretization of (6.12) leads to a set of ordinary differential equations which can be solved by the 4-stage Runge-Kutta scheme. Equation (6.12) is not a true unsteady boundary layer equation, and is only used for steady state resolution.

6.2.2 Primitive Variable Chosen As Dependent Variable

For demonstration, we consider the two-dimensional unsteady flow past a circular cylinder started impulsively from rest. The non-dimensional form of the governing equations (Cebeci 1979, Liakopoulos 1988) is

$$u_x + v_y = 0 \tag{6.13}$$

$$u_t + u\cdot u_x + v\cdot u_y = u_e\cdot \frac{du_e}{dx} + u_{yy} \tag{6.14}$$

with initial condition

$$u(x,y,0) = u_e(x) = \sin x \ , \quad (y \neq 0) \tag{6.15}$$

and boundary conditions

$$u(x,0,t) = v(x,0,t) = 0 \tag{6.16}$$

$$u(x,\infty,t) = u_e(x) = \sin x \tag{6.17}$$

$$u(0,y,t) = u(\pi,y,t) = 0 \,. \tag{6.18}$$

The computational domain in the y direction can be obtained by truncating the infinite domain to $[0, \eta_\infty]$. Using GDQ and GIQ, the u_y, u_{yy} discretization formula are the same as (6.7a), (6.7b), but v is given by

$$v_{i,j} = -\sum_{k=1}^{M} (w^I_{jk} - w^I_{1k})\cdot(u_x)_{ik} + v_{i,1} \tag{6.19}$$

or by

$$v_{i,j} = -\sum_{k=1}^{M} w^I_{jk}\cdot(u_x)_{ik} \tag{6.20}$$

$$-\sum_{k=1}^{M} w^I_{1k}\cdot(u_x)_{ik} = v_{i,1} \tag{6.21}$$

In a similar fashion to the above subsection, when the x-related derivatives are discretized by a second order finite difference scheme at $(x_{i-1/2},y_j)$, the solution procedure can be marched along the x direction, but when GDQ is used in both the x and y direction, the marching technique is invalid. In this case, the 4-stage Runge-Kutta scheme can be used for the solution of the resultant ordinary differential equations.

The use of GDQ in the x direction is still attractive although it may increase the storage. Since GDQ can achieve the same accuracy using few grid points as a finite difference scheme using a large number of grid points, the total number of the degrees of freedom can be greatly reduced if GDQ and GIQ are used in all the coordinate directions. Thus the total storage and the computational operations required may be reduced. If the GDQ is used in the x direction, we recommend the employment of (6.19) rather than (6.20) and (6.21), since the implementation of (6.21) is very complicated. For simplicity, we will use GDQ and GIQ to discretize the spatial derivatives and the integral in all cases, and use the 4-stage Runge-Kutta scheme to solve the resultant ordinary differential equations in the following calculations.

6.3 Steady Boundary Layer Solutions

In this section, the **1D**, **2D**, and **3D** steady state boundary layer resolutions obtained by the GDQ-GIQ approach will be demonstrated. Each case includes one test problem.

6.3.1 Blasius Boundary Layer

Firstly, we consider the classical Blasius boundary layer, which is governed, in mathematics, by

$$\frac{\partial^3 f}{\partial \eta^3} + f \cdot \frac{\partial^2 f}{\partial \eta^2} = 0 \ . \tag{6.22}$$

Setting u= $\frac{\partial f}{\partial \eta}$ and introducing an unsteady term, (6.22) can be written as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial \eta^2} + f \cdot \frac{\partial u}{\partial \eta} \tag{6.23}$$

$$f = \int_0^\eta u \cdot d\eta + f(0) \ . \tag{6.24}$$

For numerical simulation, the computational domain is truncated to [0, 3], and the grid is stretched near η = 0. Using the technique shown in subsection 6.2.1, we have studied the difference between the use of (6.7c) and (6.9), (6.10). It is found that when (6.9) and (6.10) are used, that is, the three boundary conditions are employed in the η direction, the allowable time step size is much larger than that when (6.7c) is used, that is, only two boundary conditions implemented. For example, when N = 12, the allowable time step size is 1.30×10^{-2} if (6.9) and (6.10) are used, and is 1.0×10^{-3} if (6.7c) is used. As a result, for the convergence criterion of the maximum residual of less than 1.0×10^{-4}, (6.9) and (6.10) require 385 time steps and 1.03 seconds of CPU time on the IBM 3090, but (6.7c) needs 5119 time steps and 12.92 seconds of CPU

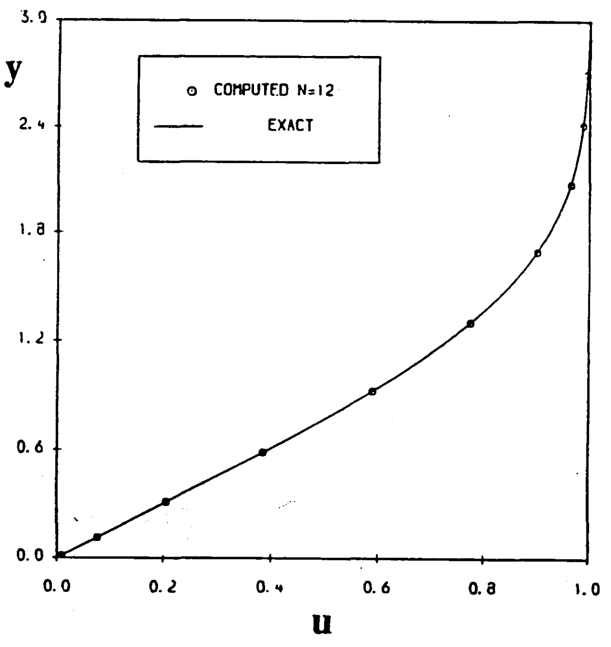
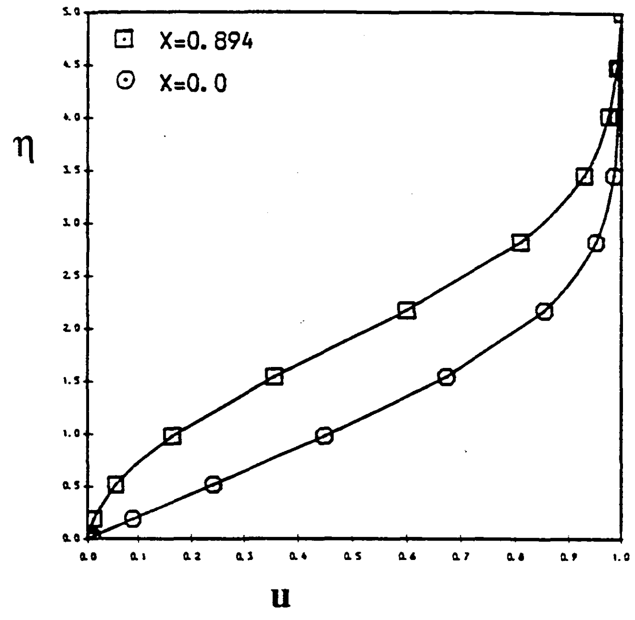


Fig. 6.1 Velocity Profile of The Blasius Boundary Layer

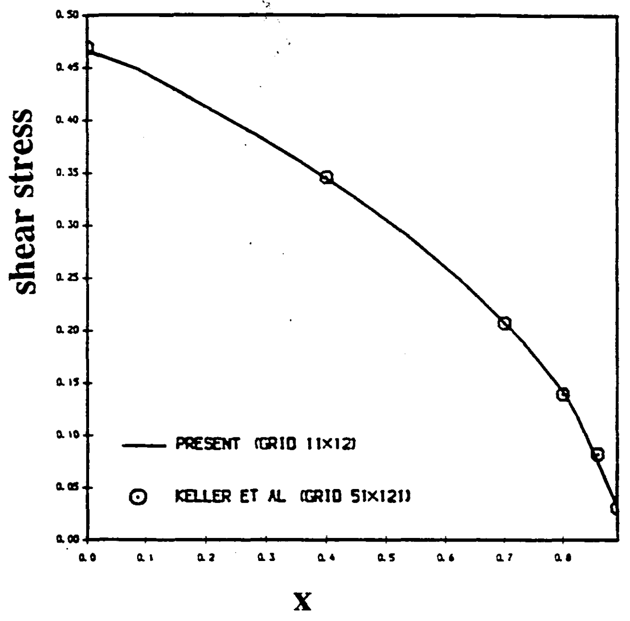
time on the same computer. In addition, it is found that (6.9) and (6.10) can give more accurate results than (6.7c). For the test problem, the exact value of the wall shear stress is 1.3284. Equation (6.9) and (6.10) give 1.3286 using N =12 and (6.7c) gives 1.3298 using N = 12. Figure 6.1 shows the computed and the exact velocity profile of the Blasius boundary layer. The computed results are obtained by using (6.9) and (6.10) and N = 12.

6.3.2 Two-Dimensional Problem

As a test example, the Howarth boundary layer is chosen for study. The governing equations are shown in subsection 6.2.1. This flow problem has a separation point at $\xi \cong 0.901$. So, the computational domain in ξ direction should be [0, b] where $b < 0.901$ because of the Goldstein singularity. In the η direction, the infinite domain is truncated to [0, 5.0]. The GDQ and GIQ techniques are used in both the ξ and the η direction, and (6.9) is used for calculating the normal velocity $v_{i,j}$. Three boundary conditions (6.8), (6.10) are used for u in the η direction. It is found that the GDQ-GIQ approach is very sensitive to the choice of b when b is close to the separation point. Actually, when b is taken as 0.90, the computation will diverge quickly after a few time steps, but when b is chosen as less than or equal to 0.894, the steady state resolution can be obtained accurately. The convergence rate is very fast when b is far from the point of 0.901, and is slow when b is very close to the point of 0.901. Fig. 6.2 displays the computational results of the Howarth boundary layer using the mesh size of 11×12, and $b = 0.894$. Clearly, the current numerical results are very close to those given by the Keller-box finite difference scheme using a large mesh of 51×121.



(a) Velocity Profile



(b) Shear Stresses at Wall

Fig. 6.2 Solutions of the Howarth Boundary Layer

6.3.3 Three-Dimensional Problem

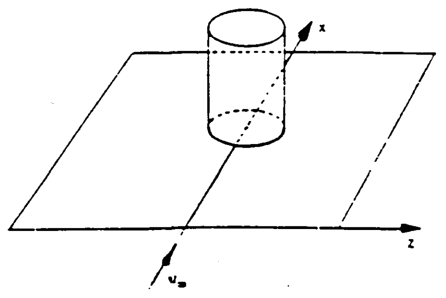


Fig. 6.3 Flow past A Flat Plate with Attached Cylinder

For **3D** boundary layer simulations, we choose the **3D** laminar flow past a flat plate with attached cylinder as a test example. The problem definition is shown in Figure 6.3. It will be assumed that the thin plate does not affect the inviscid flow around the cylinder. We will then simulate the **3D** boundary layer flow on the flat plate due to the inviscid flow created by the cylinder. This flow problem has been computed extensively and accurately by Cebeci (1975), Dwyer (1968), etc. The governing equations are (2.46)-(2.48) shown in Chapter 2, and the inviscid velocity distribution is given by

$$U_e = u_\infty[1 + a^2 \cdot \Delta_2/\Delta_1^2] \tag{6.25}$$

$$W_e = -2u_\infty \cdot a^2 \cdot \Delta_3/\Delta_1^2 \tag{6.26}$$

where

$$\Delta_1 = (x-x_0)^2 + z^2$$

$$\Delta_2 = -(x-x_0)^2 + z^2$$

$$\Delta_3 = (x-x_0) \cdot z$$

Here u_∞ is a reference velocity, a is the cylinder radius, and x_0 denotes the distance of the cylinder axis from the leading edge, x=0. To make a direct comparison between our computed results and with those obtained by Cebeci, we have chosen u_∞=30.50 m/sec, a=0.061m, x_0=0.457m, η_∞=8.0.

As we know, the boundary layer equations pose a combined initial and boundary value problem. The boundary conditions are usually obtained from the independent inviscid flow equations, whereas the initial conditions must be obtained somehow from the boundary layer equations themselves. For the test problem, since the primitive variable is chosen as the dependent variable, then for simplicity, only two boundary conditions in the η direction are implemented for F and G, that is

$$F = 0 \ , \quad G = 0 \ , \quad \text{at } \eta = 0 \qquad (5.27)$$

$$F = 1 \ , \quad G = 1 \ , \quad \text{at } \eta = \eta_\infty \qquad (5.28)$$

And V can be determined by

$$v_{i,j,k} = -\sum_{m=1}^{M} (w^I_{jm} - w^I_{1m})\cdot\Phi_{i,m,k} + v_{i,1,k} \qquad (6.29)$$

where

$$\Phi_{i,m,k} = \left[x\cdot \frac{\partial F}{\partial x} + 0.5F(1+K_3) + x\cdot We\cdot \frac{\partial G}{\partial z} /Ue + G(K_5-0.5K_4)\right]_{i,m,k}$$

K_3, K_4, K_5 were defined in Chapter 2.

The governing equations along the x=0 line of the flat plate reduce to the Blasius equation for both F and G. Thus the initial condition at x=0 can be obtained from the Blasius boundary layer solutions. In addition, the initial condition along some z equal to a constant line should be given. For the test case here, the initial condition along the line of symmetry (z=0) is first obtained. Since, along the line of symmetry, w, We are zero, we define the variable G_z as

$$G_z = \frac{\partial w/\partial z}{\partial(We)/\partial z} \ .$$

Thus the governing equations along the line of symmetry are reduced to

$$x\cdot\frac{\partial F}{\partial x}+\frac{\partial V}{\partial \eta}+0.5F(1+K_3)+G_z\cdot K_5=0 \qquad (6.30)$$

$$\frac{\partial F}{\partial T}+xF\cdot\frac{\partial F}{\partial x}+V\cdot\frac{\partial F}{\partial \eta}=\frac{\partial^2 F}{\partial \eta^2}+K_3\cdot(1-F^2) \qquad (6.31)$$

$$\frac{\partial G_z}{\partial T}+xF\cdot\frac{\partial G_z}{\partial x}+V\cdot\frac{\partial G_z}{\partial \eta}=\frac{\partial^2 G_z}{\partial \eta^2}+K_5\cdot(1-G_z^2)+K_7\cdot(1-F\cdot G_z) \qquad (6.32)$$

where $K_7 = \frac{x}{We,z}\cdot\frac{\partial We,z}{\partial x}$, $We,z = \partial(We)/\partial z$

After obtaining the initial values along the line of x=0, the initial values along the line of symmetry can be found by solving the equations (6.30)-(6.32). With the boundary and initial values, the full boundary layer solutions can be obtained by solving the equations (2.46)-(2.48), using the approach as shown in the section 6.2.

For the GDQ-GIQ simulation, the mesh size used is 11 grid points in the x and z direction, and 13 grid points in the η direction. And the computational domain in the z direction is chosen as between z=0 and z=0.20. Since the cylinder causes an adverse pressure gradient in front of itself, it is expected that the streamwise velocity will reverse in direction along a line in front of the cylinder. Since the adverse pressure gradient is a maximum along the line of symmetry, the flow reversal in the boundary layer will first occur in the line of symmetry. Thus the computational domain in the x direction should be before the separation point in the line of symmetry because of the Goldstein singularity. It is found that the GDQ-GIQ approach is very sensitive to the Goldstein singularity. When the computational domain in the x direction is taken as [0.0, 0.25970], the steady resolution can be obtained very quickly (17.76 seconds CPU time on the IBM 3090) and accurately. But when the computational domain is taken as [0.0, 0.25975], the computation will diverge after a few time steps. Thus it appears that the separation point in the line of symmetry is between 0.25970 and 0.25975, which agrees well with other researcher's results. This also demonstrates that the GDQ-GIQ approach is very accurate and efficient. Table I lists the computed values of $(\partial F/\partial \eta)_w$ and $(\partial G/\partial \eta)_w$ at some specific points. Also included in Table I are the results of Cebeci (1975) which were obtained by the Keller-box finite difference scheme with Richardson extrapolation processing. Clearly, the current results agree well with Cebeci's results. Fig. 6.4 and 6.5 show the non-dimensional streamwise and crossflow velocity profiles along the line of symmetry. Figure 6.6 displays the streamlines of the inviscid flow, and figure 6.7 shows the wall shear lines of the boundary layer.

Table I Comparison of Current Results with Cebeci's Results

		z = 0		
	Cebeci	Present	Cebeci	Present
x	$(\partial F/\partial \eta)_w$	$(\partial F/\partial \eta)_w$	$(\partial G/\partial \eta)_w$	$(\partial G/\partial \eta)_w$
0	0.332066	0.331898	0.332066	0.331898
0.0488	0.324951	0.325169	0.702488	0.702674
0.0976	0.312821	0.312574	1.124300	1.124564
0.1464	0.290184	0.290472	1.624250	1.623227
0.1952	0.243524	0.243612	2.250740	2.247190
0.2440	0.125972	0.123533	3.126830	3.119115

		z = 0.0488		
	Cebeci	Present	Cebeci	Present
x	$(\partial F/\partial \eta)_w$	$(\partial F/\partial \eta)_w$	$(\partial G/\partial \eta)_w$	$(\partial G/\partial \eta)_w$
0	0.332066	0.331898	0.332066	0.331898
0.0488	0.325184	0.325653	0.717599	0.695967
0.0976	0.314423	0.314859	1.124210	1.106794
0.1464	0.295233	0.295767	1.605920	1.585971
0.1952	0.259005	0.259305	2.191930	2.171027
0.2440	0.181979	0.177060	2.959350	2.940921

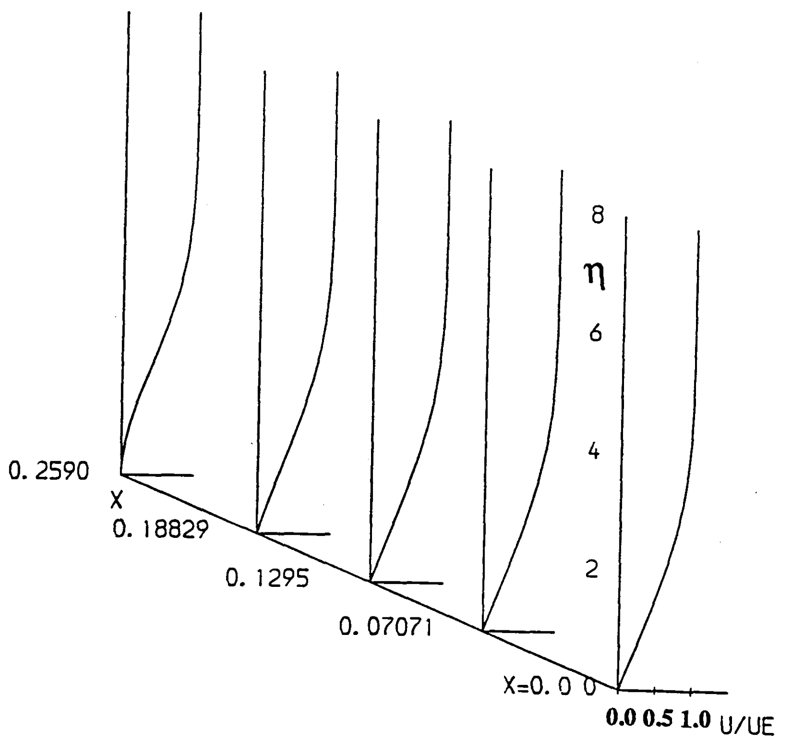


Fig. 6.4 u-Velocity Profiles along the Line of Symmetry

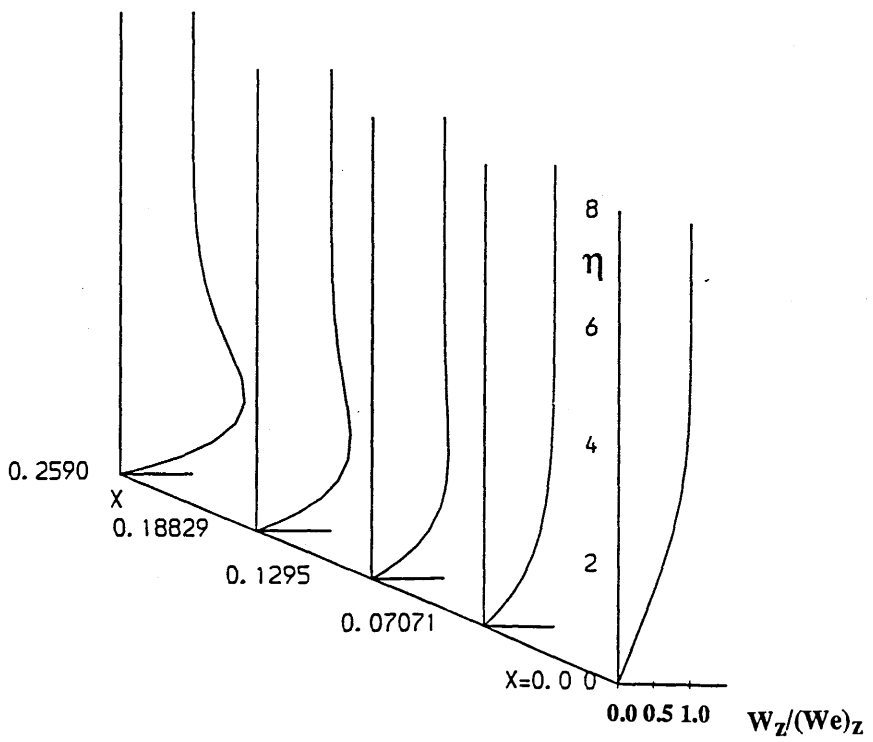


Fig. 6.5 G_Z Profiles along the Line of Symmetry

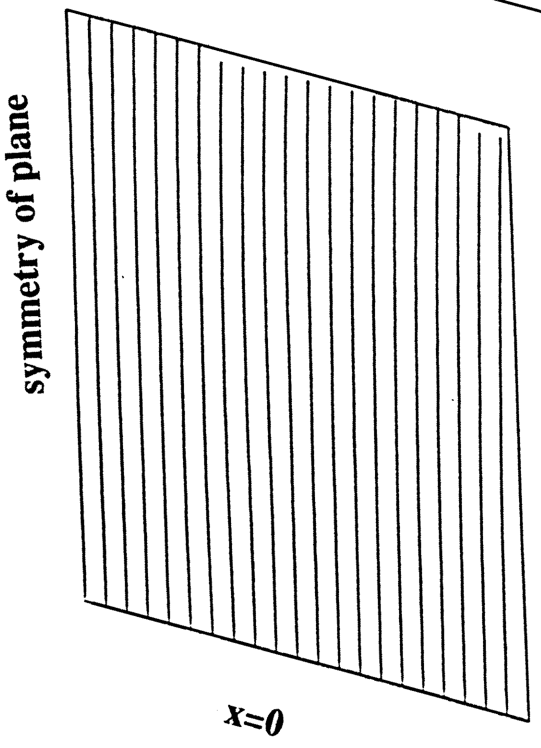


Fig. 6.6 Streamlines of the Inviscid Flow

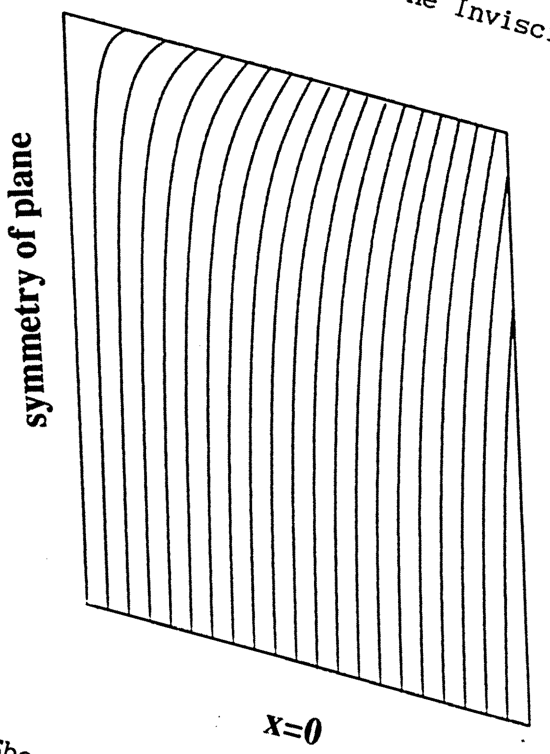


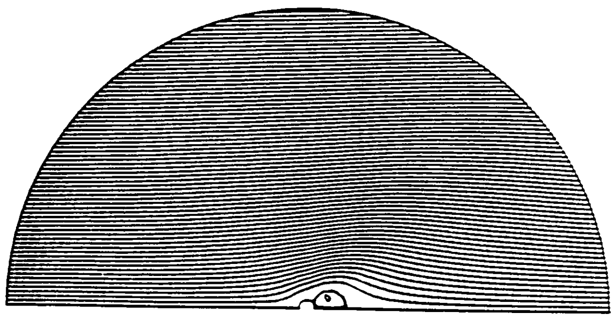
Fig. 6.7 The wall Shear Lines of the Boundary Layer

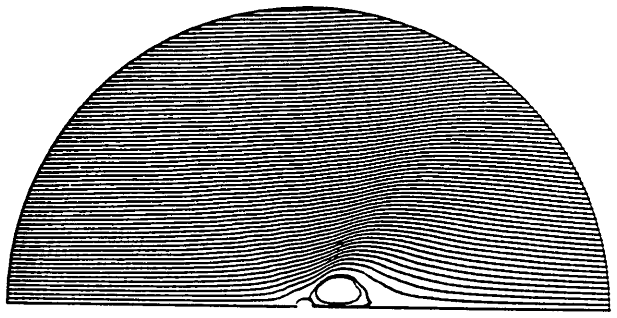
6.4 Unsteady Boundary Layer solutions

The unsteady boundary layer simulation is another area of interest in CFD. On the one hand, this is because many practical problems are unsteady. On the other hand, there is still a controversy in the literature concerning the occurrence of singularities in the solution of unsteady boundary layers. For steady boundary layer problems, it is well known that there may exist a Goldstein singularity in the boundary layer solution. But for unsteady boundary layer problems, there are arguments as to whether there exists a finite time singularity in the solution of the unsteady boundary layer equations. For the unsteady, **2D** laminar flow past a circular cylinder started impulsively from rest, some researchers (e.g. Bodonyi and Stewartson 1977, Van Dommelen and Shen 1982) claimed that there is a finite time singularity in the solution procedure, while others (e.g. Cebeci 1979, 1986) suggested that there is no finite time singularity.

We will choose this flow problem as a test case for the unsteady simulation. The problem has been introduced in section 6.2.2. To begin our study, we choose $y_\infty = 35.0$, and discretize all spatial derivatives by GDQ, and integrals by GIQ. The mesh size used is 21×31. It is found that the reverse flow first starts at $\theta=180^0$ and time, t=0.644 which is in agreement with Cebeci's results. As time increases, the point of zero wall shear moves along the surface of the cylinder towards the steady state value $\theta_G = 104.5^0$ (position of the Goldstein singularity). However, the computation cannot reach the steady state resolution because the numerical instability breaks down the calculation at $t \cong$

3.0. Fig. 6.8 shows the instantaneous streamlines computed by the GDQ-GIQ approach. Clearly, when $t = 2.5$, some wiggles occur in the streamlines. This is because GDQ is a global method, and when the solution develops a singularity at a point, this singularity will spread in the whole computational field. To study this, we use a second order finite difference scheme to discretize the derivatives in the x direction, the derivatives in the y direction being discretized by GDQ. We call this scheme the GDQ-GIQ-FD approach for convenience. Figure 6.9 shows the instantaneous streamlines computed by the GDQ-GIQ-FD approach. The mesh size used is 81×31. Compared with Fig. 6.8, when $t \le 2.0$, the results for both approaches are nearly the same, but when $t > 2.0$, the GDQ-GIQ-FD approach gives a considerable improvement over the GDQ-GIQ approach. Fig. 6.10 displays the wall shear distributions, Fig.6.11 shows the displacement thickness values, and Fig. 6.12 gives the velocity profiles at the rear stagnation point. The solid lines in these figures are the results of the GDQ-GIQ-FD approach, and the symbols are the results of the GDQ-GIQ approach. It is found from these figures that when $t \le 2.0$, both approaches agree well, but when $t > 2.0$, some GDQ-GIQ results are less accurate. All the velocity profiles at the rear stagnation point are nearly the same for both approaches. Fig. 6.13 shows the position of zero wall shear, where the dashed line is the position of Goldstein singularity. It is seen that the unsteady computation cannot reach the position of the Goldstein singularity. Table II lists the present and other researcher's results of the position of the zero wall shear and the time.


(a) t = 1.5


(b) t = 2.0

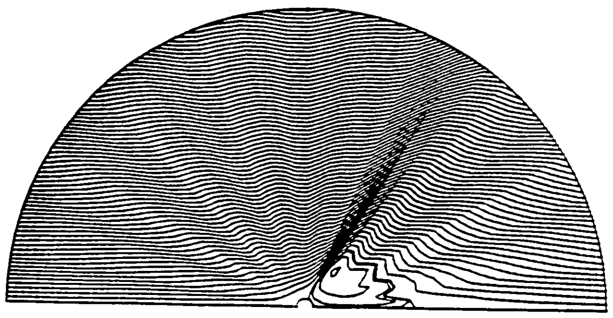
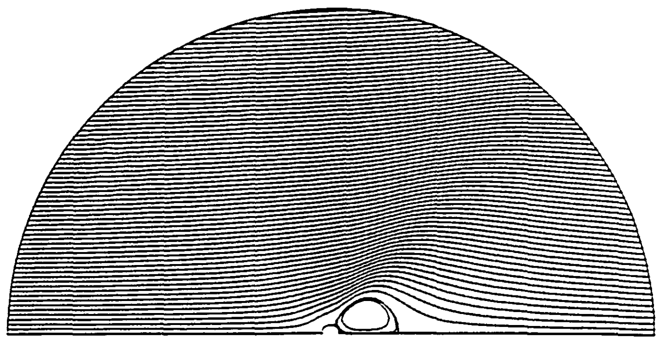
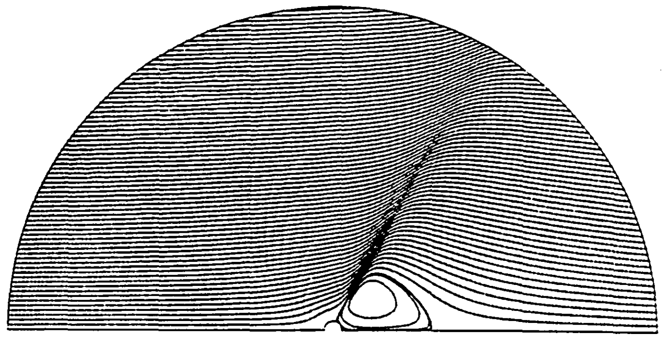

(c) t = 2.5

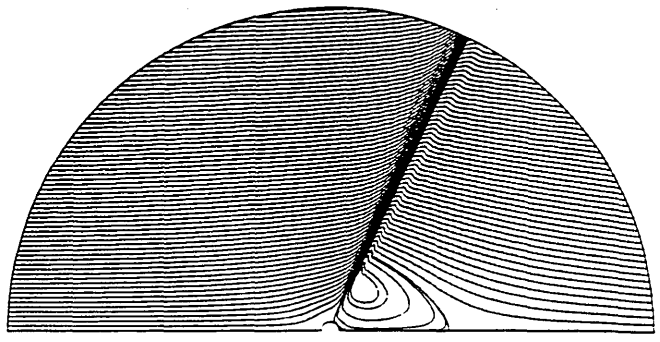
Fig. 6.8 Instantaneous Streamlines of the GDQ-GIQ approach



(a) t = 2.0



(b) t = 2.5



(c) t = 2.8

Fig. 6.9 Instantaneous Streamlines of the GDQ-GIQ-FD approach

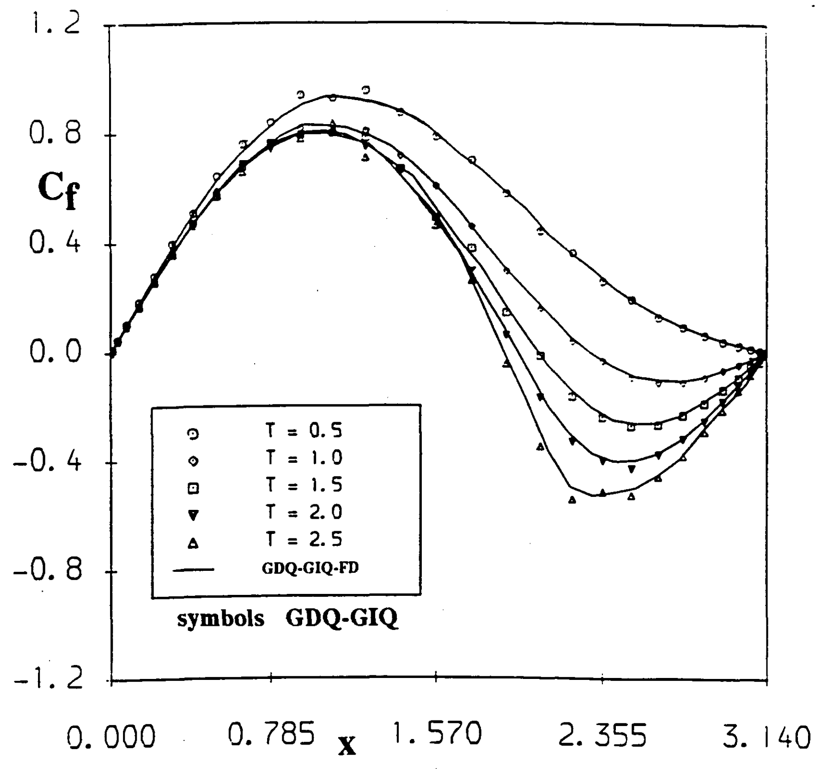


Fig. 6.10 Wall Shear Distributions

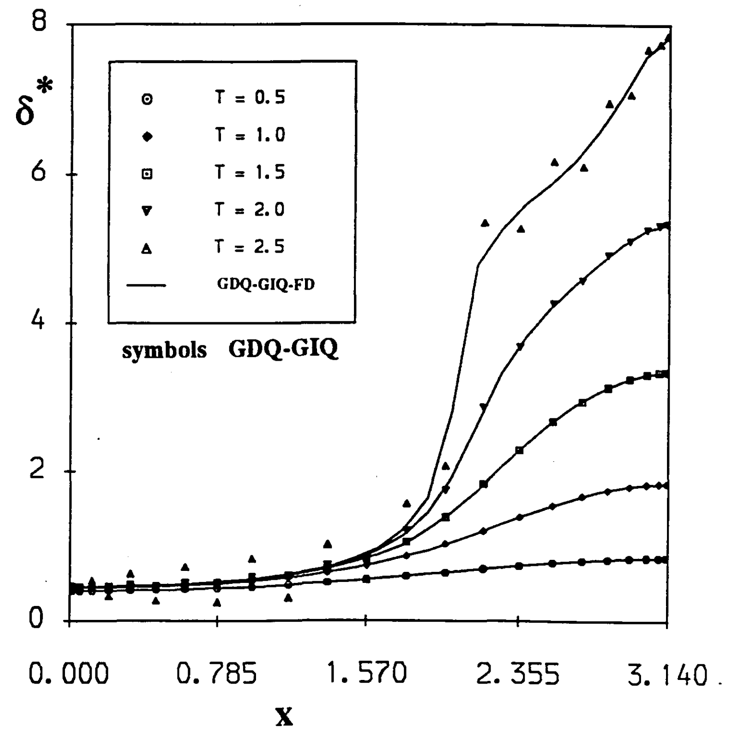


Fig. 6.11 Displacement Thickness Values

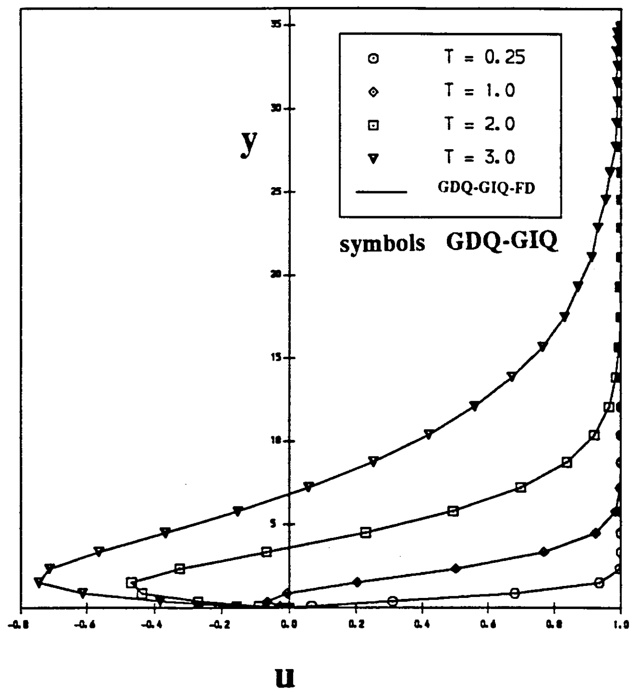


Fig. 6.12 Velocity Profiles at the Rear Stagnation Point

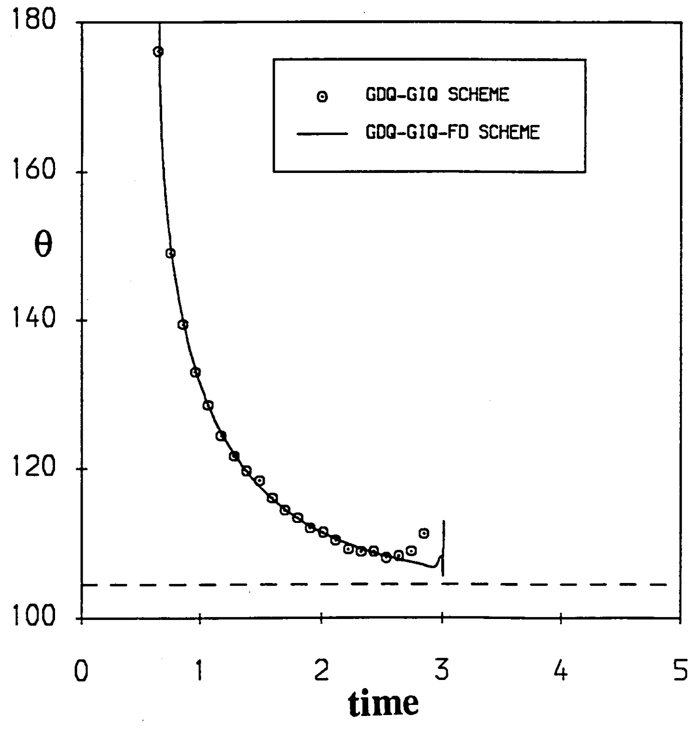


Fig. 6.13 Position of Zero wall Shear

Table II Comparison of the Time and the Zero Wall Shear Stress

Reference	180^0	166^0	146^0	138^0	124^0	110^0
Bar-Lev and Yang	0.644	0.660	0.778	0.876	1.204	2.188
Cebeci	0.640	0.660	0.780	0.872	1.192	2.200
Present (GDQ-GIQ)	0.644	0.664	0.790	0.874	1.193	2.098
Present (GDQ-GIQ-FD)	0.644	0.668	0.791	0.878	1.196	2.204

6.5 Concluding Remarks

The GDQ-GIQ approach for the solution of the boundary layer equations has been introduced in this chapter. It has been demonstrated, using test problems, that this approach can achieve accurate results using just a few grid points. Thus, both the number of degrees of freedom and the computation time can be greatly reduced. For the dependent variable, we recommend the use of the stream function as a dependent variable because in this case, the three boundary conditions can be easily implemented in the solution procedure which then yields more accurate results and needs less computation time. It is also found that when the computational field has a singularity at some point, the GDQ-GIQ approach is less efficient.

CHAPTER SEVEN

Parallel Simulations of Incompressible Navier-Stokes Flows

7.1 Introduction

As stated in Chapter One, the simulation of a general **3D**, unsteady viscous flow requires considerable computational power, which may not be achieved with single processor computers due to physical limitations such as the speed of light and gate switching limits. In contrast, parallel processor computers offer the possibility of achieving the throughput needed in CFD. The research on the effectiveness of using parallel computers for the solution of CFD problems is now becoming an active area. Although problems using explicit methods have been successfully implemented on parallel computers (Gropp and Smith 1990), it is still necessary to demonstrate the same level of performance for problems using an implicit method. In practice, when an implicit problem is considered, the domain decomposition (multi-domain) technique is usually used in parallel simulation. The major problem in the multi-domain technique is how to deal with the interface accurately for an implicit problem. The efficiency of the multi-domain approach for implicit problems is still under study since treatment of interfaces can greatly reduce accuracy and convergence rate. Incompressible viscous flow provides an implicit problem when the N-S equations with vorticity-stream function formulation or the vorticity-velocity formulation are used.

As shown in Chapter Four, as the number of grid points increases, the time step size for the ordinary differential equations resulting from GDQ becomes extremely small. This unfavourable feature can be relaxed by the GDQ element approach or the GDQ multi-domain approach. In Chapter Five, we have successfully applied the GDQ multi-domain approach to solve two complex problems on a single computer. In this chapter, these two problems are simulated on a transputer-based distributed Meiko Computing Surface using the same scheme as shown in Chapter Five. The program on each transputer was written in FORTRAN and run from an Occam harness which can control the placement and the communication between transputers. To study the influence of the interface, and compare the efficiency of the N-S formulations, we have chosen the driven cavity flow as a test problem, where three types of interface topology and two formulations of the incompressible N-S equations were investigated. Many results have been obtained. Since the parallel computer has a different architecture to the computer with a single processor, the data, tasks and communication need to be distributed among the transputers. Thus different flow problems may require different transputer architectures. This may require the modification of the parallel program when the same problem is run on different arrays of transputers, leading to an inconvenience in practice. In this chapter, the idea to develop a general code which can run on any array of transputers without any modification to the program has been developed, and successfully applied to the driven cavity flow problem.

7.2 Parallel Architectures

A number of different types of parallel architecture computers are available today. Generally, they can be classified as multiple vector processors and MIMD concurrent processors.

7.2.1 Multiple Vector Processors

Multiple vector processors consist of several vector computers operating in parallel. In these computers, mathematical operations are pipelined and performed on data which flows through the processor in a stream. Usually, in the computers the data stream is a single vector stream, so the computers in this category are also called vector computers. A great advantage of these machines is that old serial code can be run without modification, users then being able to improve performance by "vectorizing" the costly sections of code. To run in parallel, exchanges of information and synchronization between the processors is facilitated by special high speed communication hardware. In general, each single processor is very powerful, and the degree of parallelism available is not very high, usually two to six processors being available. Examples are the Cray X-MP, Cray 2 and IBM 3090 600 VF.

7.2.2 MIMD Concurrent Processors

The MIMD (multiple instruction, multiple data-stream) parallel computers work on essentially independent data in parallel. This kind of computer has become a feasible proposition over the last few years due to the cheapness of microprocessor systems. Numerous microprocessors can be linked together in a loosely-bound network in which they all have their own independent memory, or they can be combined as a tightly-bound network in which each processor can access any memory.

Tightly coupled MIMD machines are collections of independent processors which are closely connected, usually by shared memory. A parallel program running on this type of machine consists of independent threads of control which may access each other's data and can tightly control each other's operation by manipulating this shared data. The advantage of this type of machine is that there is no single thread of control, and hence no a priori serial execution. One example of this type is the BBN Butterfly machine, which consists of a large number of microprocessors, each with 1 to 4 Mbytes of memory, interconnected via a butterfly switch.

Loosely coupled MIMD (distributed memory) machines are collections of independent processors which communicate through some reliable mechanism, but do not share directly any data. Instead, all interprocessor sharing of data is done by I/O operations, typically the sending and receiving of message packets. This provides a measure of programming safety and reproducibility of results often absent in tightly coupled MIMD machines, since all modifications to "shared" data structures are done explicitly by the programmer, rather than implicitly through a shared memory access. This kind of machine can easily be developed to a massively parallel computer. As the number of processors increases, both the memory and floating point operation per second increase. Examples of this type include the transputer-based systems and the LCAP system of IBM.

7.2.3 Transputer-Based System

7.2.3.1 Transputer Hardware

The MIMD concurrent computers can be constructed out of (a) large processors each of which could be the basis of a powerful computer; (b) processors which occupy a board and are equivalent to a full mini-computer; and (c) single-chip processors which can perform the full range of requisite functions. The transputer-based systems are in category (c), where the single chip is the transputer.

Transputers are built by INMOS Ltd. It is a programmable building block for concurrent systems, spanning a wide range of system sizes from microcomputer to supercomputer. The transputer is designed to implement the process model of concurrency, expressed through the Occam programming language, which was developed in parallel with the hardware. The transputer architecture is wordlength independent so that transputers of different wordlengths may be interconnected and programmed as a single system. Since all memory is local, the memory bandwidth grows in proportion to the number of transputers. Each transputer has an external memory interface which extends the address space into off-chip memory. Transputers use point to point communication links. Every member of the transputer family has one or more standard links which may be connected to links on other transputers to build networks of various sizes and topologies. Hence, the communication bandwidth does not saturate as more transputers are added. Each link provides synchronous bi-directional communication corresponding to two Occam channels, one in each direction. Communication via any link may occur concurrently with communication on all other links and with program execution. The transputer can be programmed in other high level languages such as FORTRAN, C, PASCAL, but that if concurrency is to be exploited , Occam should be used as a harness to link modules written in the selected language. Figure 7.1 is a schematic diagram of a transputer.

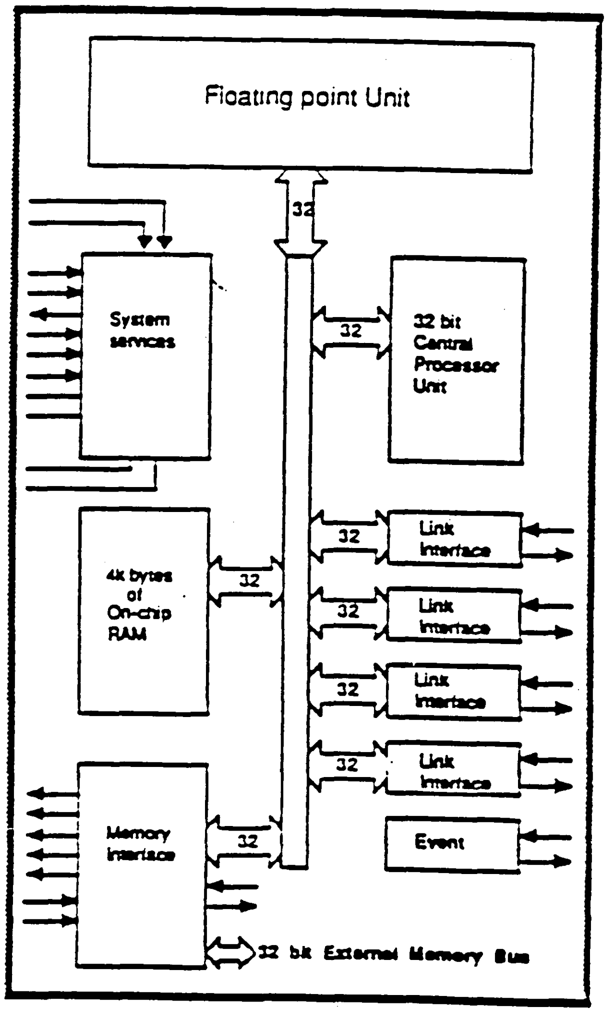


Fig. 7.1 Schematic diagram of a transputer

Among the family of transputers, **T414** and **T800** are extensively used.

T414 is a **32** bit, **10** MIPS processor with **2**Kbyte of memory and sustainable **32** bit floating-point performance of around **80** Kflops;

T800 is a **32** bit, **10** MIPS processor with **4**Kbyte of memory, a **64** bit floating-point unit which is capable of sustaining **1.5** Mflops which is around **5** times faster than a **MC68020** processor and about the same speed as a **VAX 8600**.

All transputers are single 1.5μ CMOS chips with a reduced instruction set architecture, supporting 4 INMOS standard, full duplex, serial links.

7.2.3.2 The Meiko Computing Surface

There are several computing systems available which use the transputers as the basic processors. Amongst them, the Meiko Computing Surface is widely used. The Meiko Computing Surface is a computer system designed by Meiko Ltd. to exploit the power of the transputer on compute-intensive applications. Compute, graphics and I/O elements are available, together with interface and intermodule boards. The links from each element are electronically switched enabling the user to create the topology required by their program. The Meiko Computing Surface is a modular, reconfigurable transputer array, which consists of a number of modules each containing up to 40 boards housed in two 19-inch racks. All inter-board links within a module are routed via the system backplane, and links between modules are provided by special

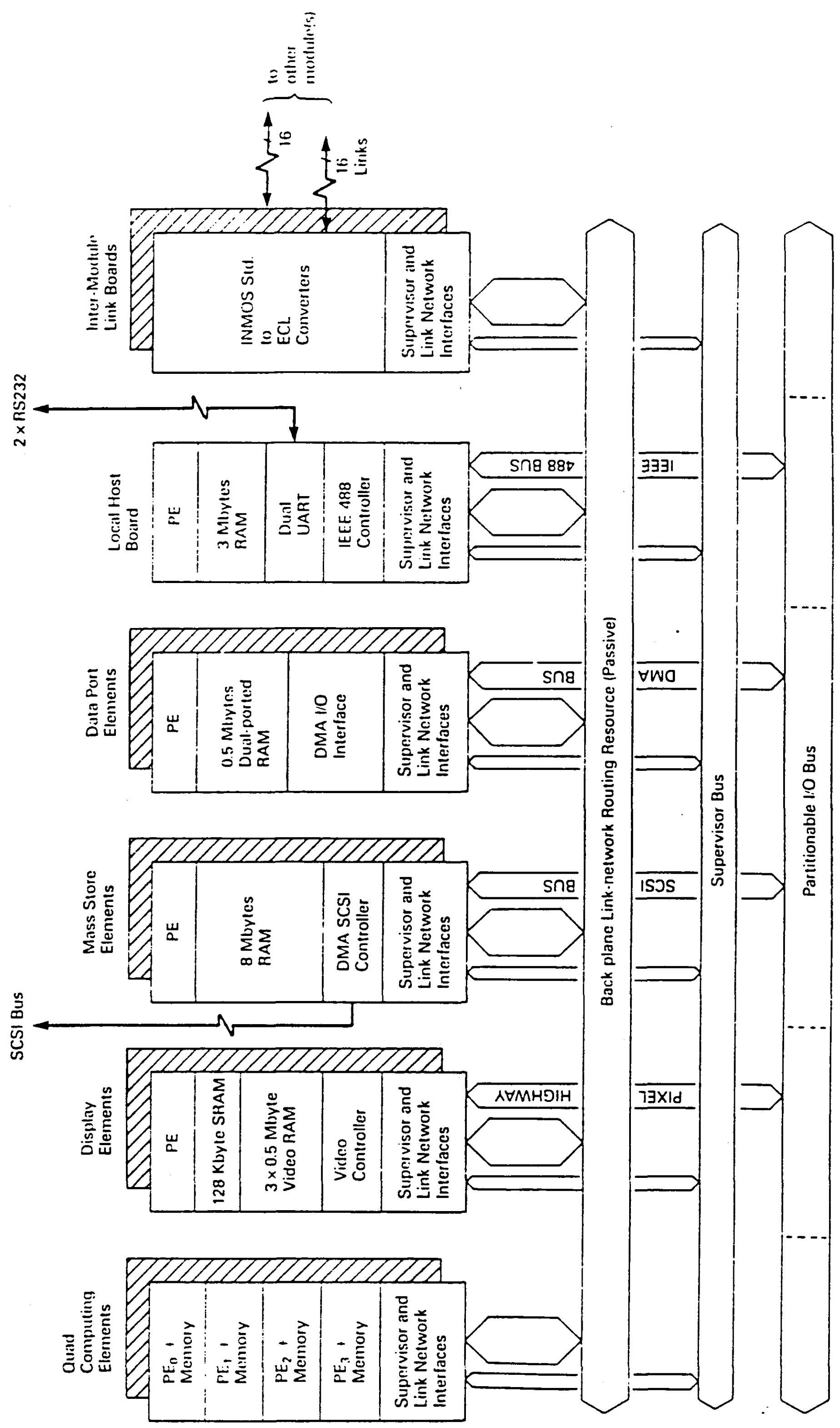


Fig. 7.2 Structure of the Meiko Computing Surface

inter-module link boards. The structure of the Computing Surface is illustrated in Fig. 7.2. The system backplane in each module supports a supervisor bus as well as the link connectivity, and this provides for low bandwidth communication between all computing elements in the system. Meiko provides system software to control a computing surface in single or multiple mode, and are developing MEIKOS, a UNIX-like system. In the system, there are compilers for Occam, C, Fortran and Pascal, each generates code for T414's and T800's. Communication over Occam-style channels has been added to the conventional languages. Multiprocessor programs are built by linking sequential single programs (in C and Fortran) with an Occam harness.

The Meiko Computing Surface is still in development. Since the work in this thesis was completed, CS tools have been made available. These can run the pure conventional languages in parallel without establishing an Occam harness. This makes it easy to use for a new user, but may reduce the efficiency since the configuration may not be optimal to some physical problems.

7.3 Parallel Algorithms

Ideally, a program runs N times faster on N processors than on a single processor, although the actual speed-up may be much less. The design of algorithms to achieve this sort of speed-up is an active area of research. Since the algorithm, programming language and hardware are intimately connected, the major effort should be to match the parallelism of the algorithm to the parallelism of the computer in such a way as to minimise the execution time of the program. At any stage within an algorithm, the parallelism of the algorithm is the number of operations that are independent and can therefore be performed concurrently. This may vary from stage to stage. The natural hardware parallelism is the number of processors that may run concurrently, including both arithmetic and link processors. The current parallel algorithms can be roughly categorised into three classes: event parallelism; geometric parallelism; and algorithmic parallelism.

7.3.1 Event Parallelism

Event parallelism is suitable for independent tasks where each processor executes a program in isolation from all the other processors. One of the simplest, and often the most efficient, ways of exploiting parallel processing is to distribute independent tasks to each of the processors. Such a configuration of the system is also called a task farm. Another variant on the task farm approach, is when a single computation can be divided into many independent sub-tasks which can be farmed out amongst the slave processors. The event parallelism is an ideal way to make parallel computation, but unfortunately, most engineering problems are beyond this type.

7.3.2 Geometric Parallelism

The geometric parallelism is also called domain decomposition or multi-domain parallelism, in which each processor executes the same program on data corresponding to a subdomain of the system being simulated and communicates boundary data to neighbouring processors handling neighbouring subdomains. Since, at the present level of technology, communication is expensive, we should minimise the data transferred from one processor to another processor. In addition, the communication time is also proportional to the distance between processors. So, the processor array should have, as far as possible, the same geometric configuration as the system being simulated. Geometric parallelism has been achieved in some useful applications of CFD problems (Lin 1989, Gropp and Smith 1990).

For transputer-based systems, **2D** problems can be treated in a straight forward manner because each subdomain has at most 4 neighbouring subdomains. This requires 4 channel pairs for communication, and the transputer provides a promising way to do that. But for **3D** problems, the application is not straight forward. Since some subdomains may require 6 links, we need to build a block using 2 transputers for each subdomain. This introduces the extra complication of distributing the data in one subdomain over the transputers, and the handling of internal communications.

7.3.3 Algorithmic Parallelism

This approach is to construct a network of processors, each with its own special role to play, through which all the data flows, as in a factory production line. There are a number of difficulties with algorithmic parallelism. One is that at different stages during the computation, different algorithms may apply, and a configuration of transputers optimised for implementing one algorithm is unlikely to be appropriate for another. Another difficulty to be solved is how to get control data to each of the slave processors, for example, to initialise them at the start of the computation. Finally, it may happen that one process dominates the execution time. If this process cannot be divided up amongst more than one processor, it alone will determine the throughput and constitute a bottleneck. Because of these difficulties, many published papers referring to algorithmic parallelism are beginning to appear in mathematical periodicals and books. Obviously, algorithmic parallelism is a natural way to simulate a machine, a production process, or even a whole factory. The major obstacle to efficiency is load blancing: the number of processors assigned to simulating each component must be carefully tuned.

7.4 Domain Decomposition and Topology of the Interface

In the domain decomposition technique, the overall computational fluid field is divided into several subdomains, equal in number to the number of available slave processors. With the boundary conditions at the physical boundary or at the interface, the evolution of the fluid in the interior of each subdomain is determined entirely by data which is present in that processor's local memory. But since the boundary conditions at the interfaces are usually given the initial values at the beginning of the solution procedure which are not the solutions of a problem, they should be corrected as iteration progresses. Generally for simplicity, the functional values at the interfaces are determined by an explicit formulation for an implicit problem, which is related to the neighbouring subdomains. Thus communication between processors is required for exchanging the data to give the new functional values at the interface. The formulation to determine those values at the interface is related to the topology of the interface. In the following, three basic cases are considered.

7.4.1 Patched with Continuity Condition (Interface I)

This case has been considered in Chapter Five. Along the interface, the function is C^1 continuity, and may not satisfy the N-S equations. According to formulation (5.35), the functional values at the interface are related to all the interior values of both adjoining subdomains. Since each subdomain is assigned to a processor, that means all the interior values in a subdomain should be transferred into neighbouring subdomains. This may cost considerable communication time. To reduce that, we can combine all the interior functional values using (5.35) to give a stream of data, which is, in number, equal to the number of grid points at the interface, and is transferred into neighbouring processors to determine the functional values at the interface.

7.4.2 Patched with Interpolation (Interface II)

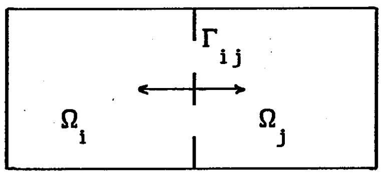


Fig. 7.3 Topology of a Patched Interface

The topology of the interface II is the same as the above case. But the functional values at the interface are obtained using a high order Lagrange interpolated polynomial. For simplicity, it is assumed that subdomains Ω_i and Ω_j have the same structure of grid and the same local coordinate system, and globally, the coordinates in both subdomains are symmetric to the interface Γ_{ij}. Thus, we can establish a frame whose origin is on the interface for subdomains Ω_i and Ω_j, as shown in Fig. 7.4. Using the Lagrange interpolation polynomial, the functional values at the interface can be obtained by the extrapolation from both sides of the interface, which is given as

$$\bar{f} = \sum_{k=2}^{L} L_k^r \cdot f(x_k^j) + \sum_{k=2}^{L} L_k^l \cdot f(x_{N-k+1}^i) \qquad (7.1)$$

where x_k^i, x_k^j are the local coordinates in Ω_i and Ω_j, $\bar{f}$ is the functional value at the interface, (L-1) is the number of the functional values in a subdomain being used for the interpolation, and

$$L_i^r = L_i^l = \frac{1}{2} \cdot \prod_{j=2, j\neq i}^{L} \frac{x_j^2}{x_j^2 - x_i^2}$$

Here $x_k = x_k^j$. Similar to the above case, the interior functional values in a subdomain can be combined, using (7.1), to give a minimum stream of data before transferred into neighbouring subdomains. In this case, the functional values at the interface may not satisfy the governing equations.

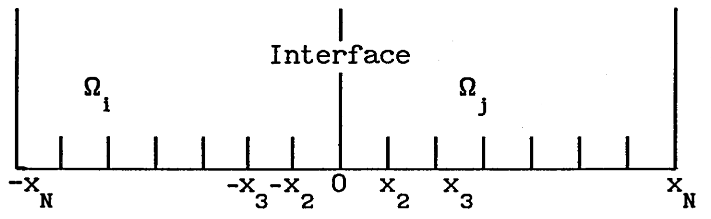


Fig. 7.4 Global Coordinates for subdomains Ω_i and Ω_j

7.4.3 Overlapped (Interface III)

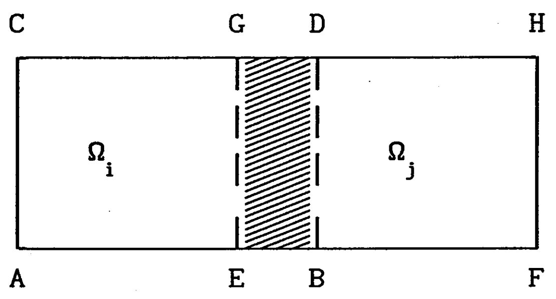


Fig. 7.5 Topology of Overlapped Interface

The overlapped topology of the interface, as shown in Fig. 7.5, is an easy way to implement parallel computation using the domain decomposition approach. In Fig. 7.5, subdomain ABCD is overlapped with subdomain EFGH (shaded area). It is noted that the right boundary of subdomain Ω_i, BD, is in the interior of subdomain Ω_j, and the left boundary of subdomain Ω_j, EG, is in the interior of subdomain Ω_i. Thus, if the solution in the interior of the subdomains is known at a time step, then the functional values along the lines of BD and EG are known. These values are then transferred into neighbouring subdomains as new boundary conditions to get the solution in the interior of subdomains at next time step, i.e. the values along EG are transferred into subdomain Ω_j, and the values along BD are transferred into subdomain Ω_i. This process continues until converged solutions in all subdomains are obtained. For the overlapped topology, the functional values at the interface are given from the solution of the governing equations, but the functional values within the overlapped region may not be unique.

They can be determined from the solution of a subdomain, and can also be given from the solution of another neighbouring subdomain. The overlapped region may include 2 or more grid points overlapped.

7.5 Development of A General Computer Code

A parallel program, which can run on any number of processors without any change of the program, is very attractive in engineering. To develop this, we firstly consider the particular features of parallel computation. Usually, these new considerations include: (1) how the data is to be distributed in the memory; (2) how computations are distributed among the processors; (3) inter-processor communications; and (4) inter-processor connections (configurations). If a domain decomposition technique and the distributed Meiko Computing Surface are used, the first two items are easy to implement since each processor is assigned to be responsible for the computation of a subdomain and the data used in the computation is stored in the local memory of that processor. The problem lies in determining how the master processor divides the whole computational field into required subdomains, each with its local properties such as the generation of local coordinates, assignment of the initial field, and the way that the local slave processors ascertain the location of the physical boundary and the interface in a subdomain. It is very important to know the position of the interface in the solution of a differential equation. This is because, on the one hand, the interior solutions are greatly affected by the boundary conditions, not only in their values and types, but also in their positions. On the other hand, since the data communication is required across the

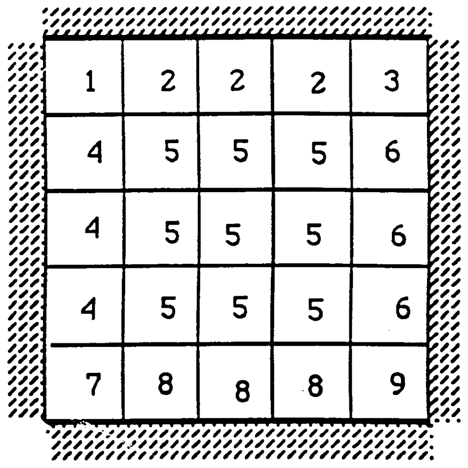


Fig. 7.6 Basic Cases for A Two-Dimensional Problem

interface, the inter-processor communications and connections are related to the position of interfaces. In other words, depending on the position of the interfaces in a subdomain, the local slave processor needs to know where the communications and connections are required to the neighbouring subdomains. Thus, in terms of the topology of subdomains, we may generate a program which can automatically distribute the data and computation to each slave processor, and produce a configuration for the communications and connections. To demonstrate this, we consider a rectangular computational field for simplicity. For this case, if it is assumed that in both the x and y directions, there are 2 or more than 2 subdomains, then all the topology of subdomains can be described within a set of 9 basic cases, as shown in Fig. 7.6, where i indicates the case i, $i=1,2,\cdots,9$. If the program includes all these 9 basic cases, it can then be run on any array of processors (again only if the number of subdomains in both the x and y directions are 2 or more than 2).

7.6 Numerical Results

To test the efficiency of a parallel computation, the two channel flow problems, which have been studied using a multi-domain GDQ approach in Chapter Five, are chosen to extend to parallel schemes. Some preliminary results for a comparative study of the driven cavity flow problem to demonstrate the application of the general computer code in section 7.5 are also presented. The efficiency of a parallel computation is defined as

$$\eta = \frac{t_1}{N \cdot t_N} \tag{7.2}$$

where

t_1 = time taken by the program on a single processor

$N \cdot t_N$ = time taken by the program on N slave processors

Using the Meiko Computing Surface, the computational program is written in Fortran operating under the MeikOS system, and the Occam harness is edited by the Occam 2 language operating under the OPS (Occam Programming System). The network of transputers consists of one host transputer, one master transputer, and an array of slave transputers. In the present research, the master transputer is also a slave transputer. The host transputer is one which runs OPS. The master transputer directly connects to the host transputer which handles I/O with the outside world (terminal, keyboard, files on disk, etc.), and which may control a number of slave transputers. In particular, for the solution of incompressible N-S equations, the master processor firstly reads input data from a file on the disk, then broadcasts them to all the slave processors, and sends a message telling them to begin work. After marching a time step, all the slave processors send the maximum residuals in their local subdomains to the master processor, and the master processor then compares them with a given tolerance. If they are within the tolerance, the master processor then sends a message telling the slave processors to cease their calculations and send the results to the master processor for output to a file on the disk, if not, then sends a message telling the slave processors to continue their tasks. The slave transputers are the working processors.

7.6.1 The Flow past A Backward Facing Step

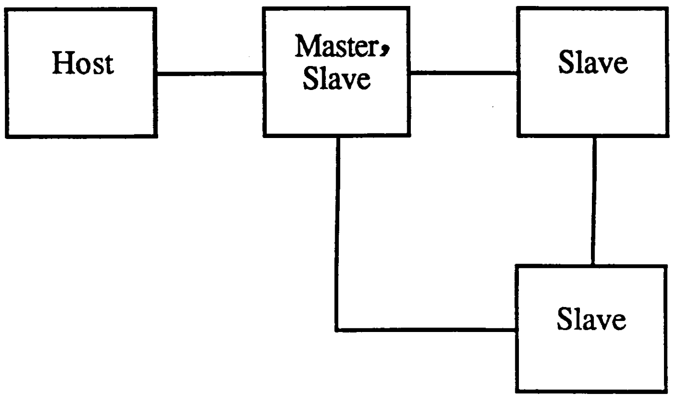


Fig. 7.7 Configuration of A Backward Facing Step Problem

The network of transputers for this case is shown in Fig. 7.7, which has the same configuration as the physical system. Three slave transputers (one is also a master transputer) are used for parallel simulation. Almost the same program (without communications and connections) has also been run on a single transputer. It was found that numerical results obtained by a single transputer and multiple transputers are

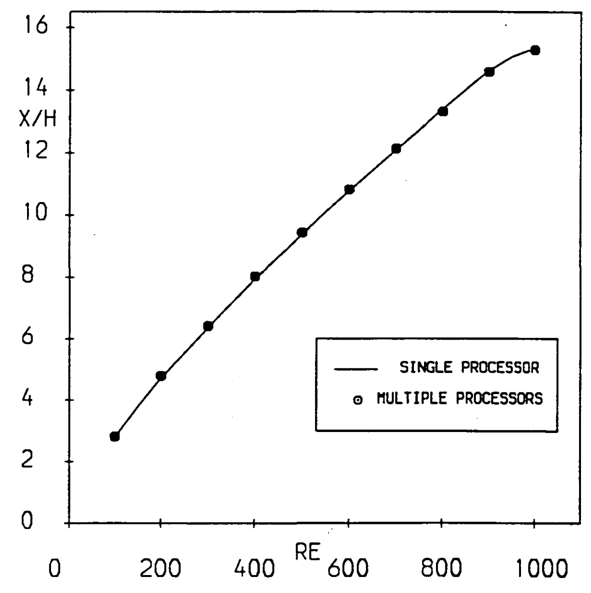


Fig. 7.8 Lengths of Recirculation Zone vs Reynolds Numbers

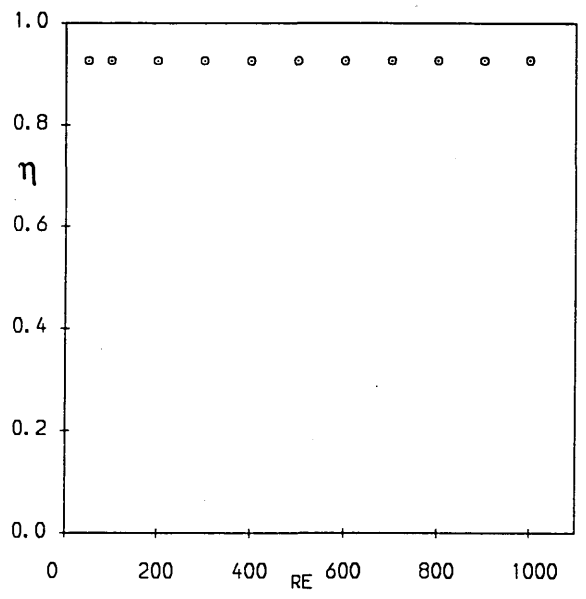


Fig. 7.9 Efficiency of the Parallel Simulation

nearly the same, the efficiency of parallel computation is very high, which is about 92.5% . Fig. 7.8 shows the length of recirculation zone vs Reynolds numbers for both cases. Clearly, both cases give almost the same values. Fig. 7.9 displays the efficiency of the parallel simulation, where $\eta = t_s/t_m$, t_s is the operation time taken by the single processor, t_m is the total operation time taken by 3 slave processors.

7.6.2 The Flow past A Square Step

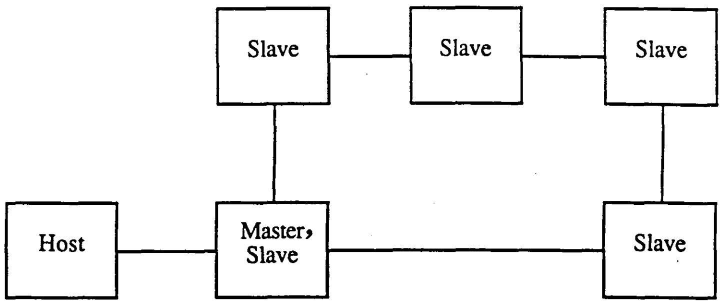


Fig. 7.10 Configuration of A Square Step Problem

The network of transputers for parallel simulation is shown in Fig. 7.10, which, again, has the same configuration as the physical problem. To study the efficiency of the parallel computation, the program (without communications and connections) was also run on a single transputer. Numerical results for both cases are nearly the same. Fig. 7.11 shows the efficiency of parallel simulation for this problem, where $\eta = t_s/t_m$, t_s is the operation time taken by a single processor, t_m is the total operation time taken by 5 slave processors. The efficiency for all the simulated cases is over 90% .

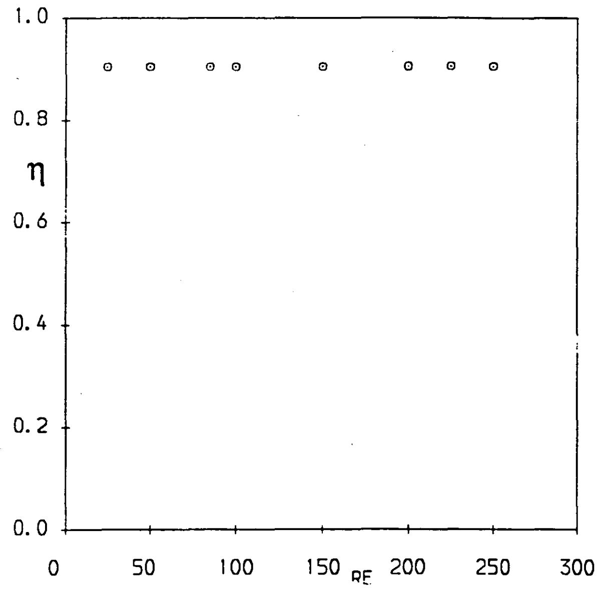


Fig. 7.11 Efficiency of the Parallel Simulation

7.6.3 Comparative Studies of the Driven Cavity Flows

For this test case, we firstly use the vorticity-velocity formulation to study the influence of the order of interpolated polynomials, the number of grid points overlapped, on the accuracy of results and the operational time. The comparison of the accuracy and operational time needed, between three types of interfaces using the vorticity-velocity formulation, and between the vorticity-velocity and vorticity-stream function formulations, are also studied. Finally, the accuracy of results and the computational efficiency are discussed, as the number of subdomains increases. In all the following cases, the program is produced using the general scheme proposed in section 7.5, which can be run on any array of transputers without modification to the program. One type of network of transputers is shown in Fig. 7.12, which uses 5x5 slave transputers, and has the same configuration as the physical case.

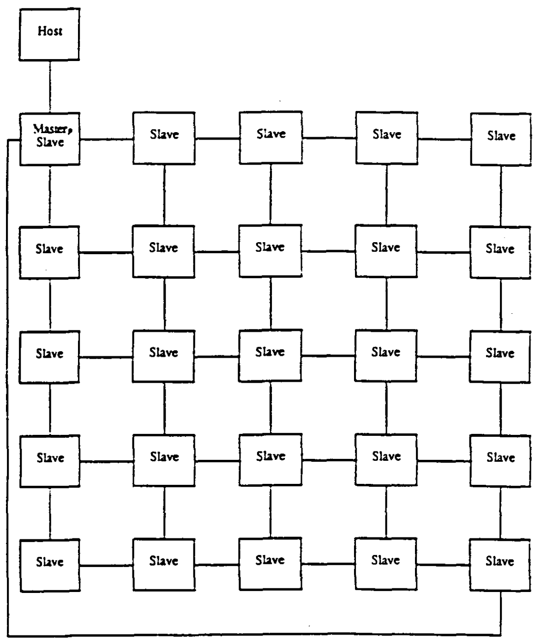
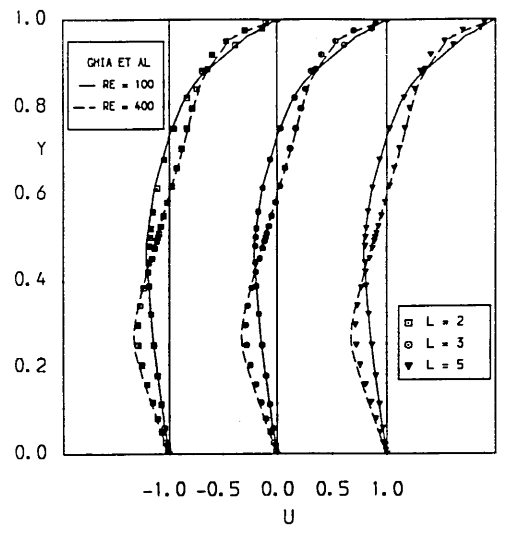


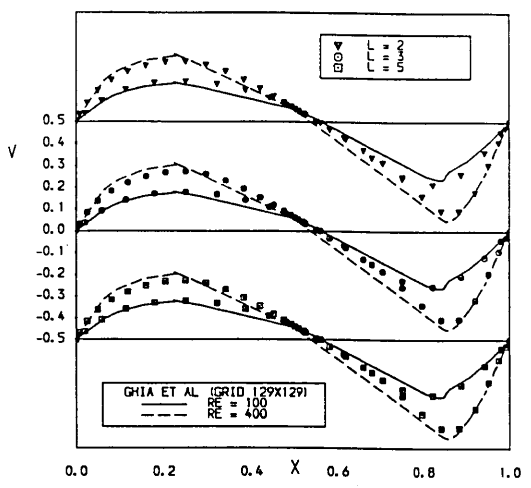
Fig. 7.12 Configuration of A Driven Cavity Flow Problem

7.6.3.1 Different Order of Lagrange Interpolated Polynomials

Using interface type II, it was found that, as L increases from 2 to 3, the accuracy of results is improved, and more operational time is required; and as L increases above 3, the accuracy of results keeps nearly the same, and a little more operational time is needed. Thus in balance, to reduce the operational time and obtain accurate results, the use of L=3 is recommended. Fig. 7.13 displays the velocities through the geometric centre of the cavity, where the array of 2x2 slave transputers (or subdomains), and a local mesh size of 11x11 for Re=100, 17x17 for Re=400 in each subdomain, were used. Fig. 7.14 shows the non-dimensional operation time, where the reference time T_{ref} is the operational time



(a) Horizontal Velocity



(b) Vertical Velocity

Fig. 7.13 Velocities Through the Geometric Centre of the Cavity

(for section 7.6.3.1)

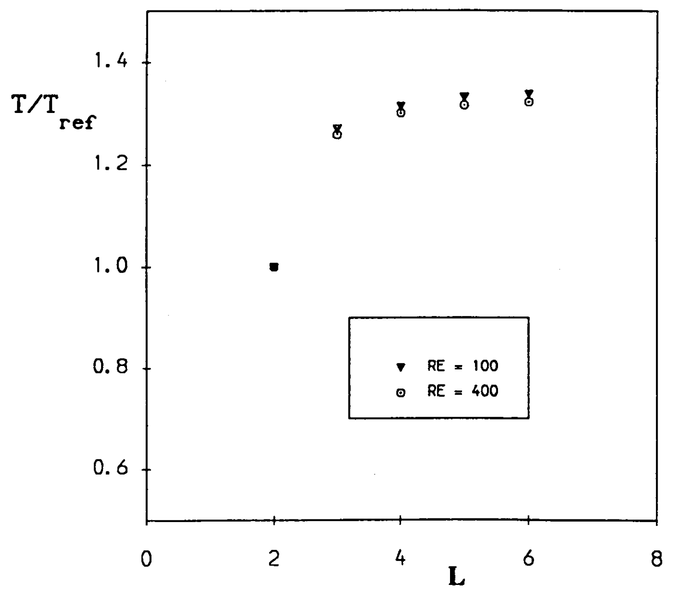


Fig. 7.14 Comparison of the Non-Dimensional Operational Time

(for section 7.6.3.1)

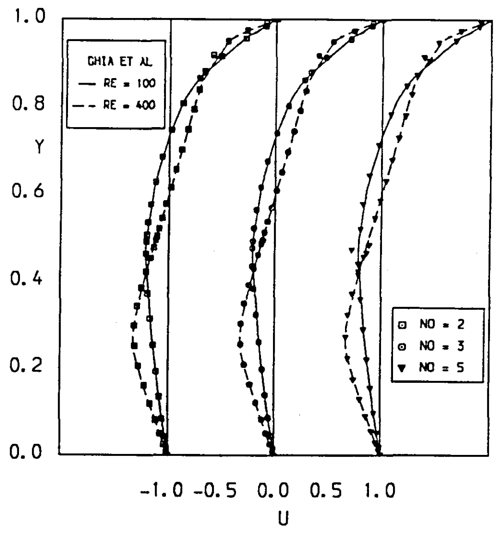
taken by the case of L=2.

7.6.3.2 Different Number of Grid Points Overlapped

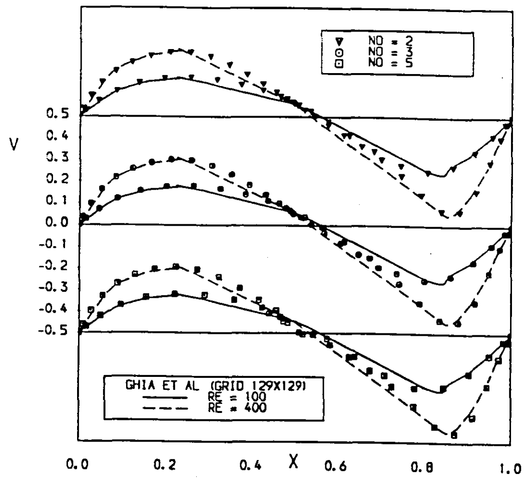
Using interface Type III, it was found that, as NO increases from 2 to 3, where NO is the number of grid points overlapped, the numerical results are nearly the same, but the operational time is greatly reduced; and when NO increases above 3, both the accuracy of results, especially in the overlapped region, and the operational time, are reduced. The reason for reduction of the accuracy in the overlapped region is that solutions in this region may not be unique since they can be obtained in a subdomain and in other neighbouring subdomains. Although the physical positions in the overlapped region are the same, solutions derived from different subdomains may not be consistent with each other. This is particularly true when NO becomes relatively large. Hence, to obtain accurate results with less operational time, the use of NO=3 is recommended. Fig. 7.15 shows the velocities through the geometric centre of the cavity, where the array of 2×2 slave transputers and the local mesh size of 13×13 for Re=100, 17×17 for Re=400 in each subdomain, were used. Fig. 7.16 shows the non-dimensional operational time, where the reference time T_{ref} is the operational time taken by the case of NO=2.

7.6.3.3 Comparison of the Interface Treatment

Using the vorticity-velocity formulation, the performance of the three types of interface treatment introduced in section 7.4, has been studied. Numerical experiment showed that the interface III gives the



(a) Horizontal Velocity



(b) Vertical Velocity

Fig. 7.15 Velocities Through the Geometric Centre of the Cavity

(for section 7.6.3.2)

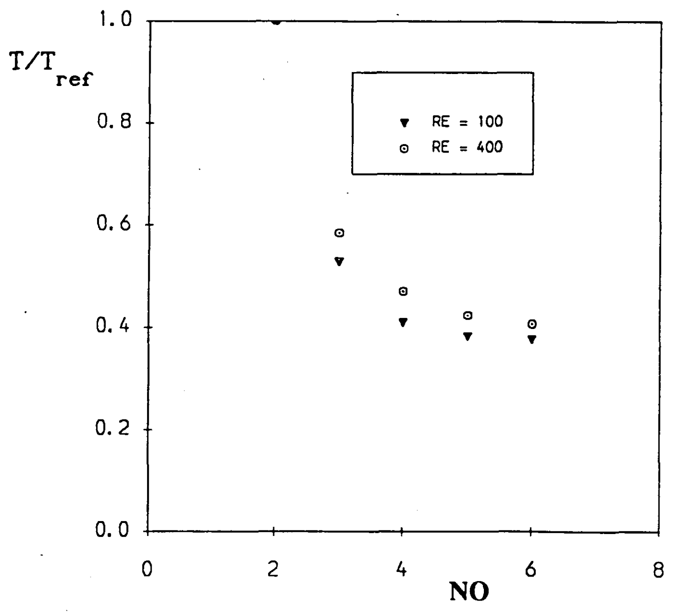
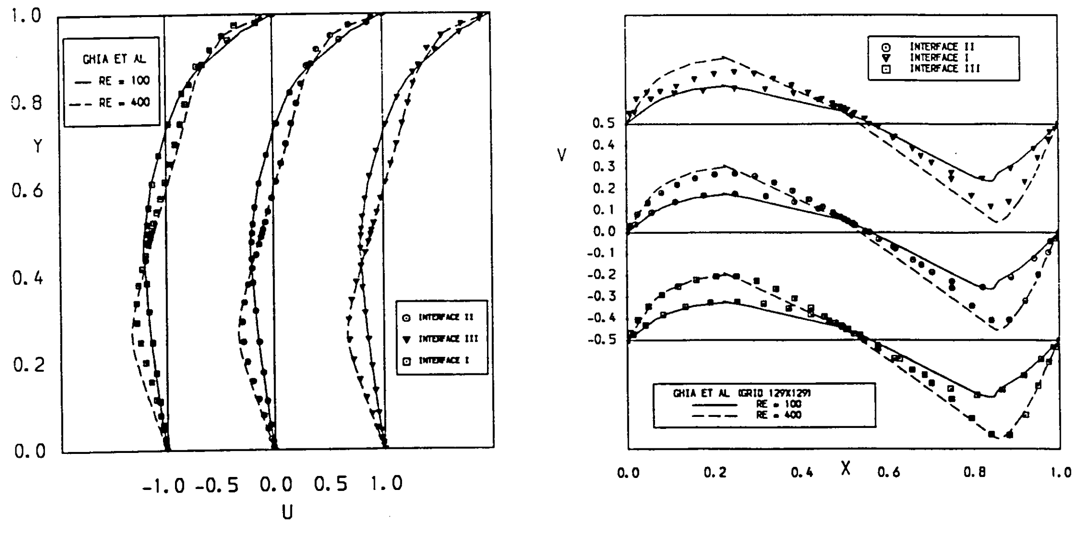


Fig. 7.16 Comparison of the Non-Dimensional Operational Time

(for section 7.6.3.2)

most accurate results and needs the least operational time, and the use of interface II presents more accurate results and requires less operational time than that of interface I. For the reason of this behaviour, it is analysed that the solutions in the interior of each subdomain can be affected by the boundary conditions and by the solutions at the interface, thus any error introduced at the interface may spread into the interior solutions. Hence, although the interior solutions are obtained by GDQ with high order accuracy, the low order solutions at the interface may produce low order solutions in the whole computational field. Since the use of interfaces II and III give solutions at the interface with high order accuracy, which is consistent with the accuracy of interior solutions in subdomains, and the use of interface I gives solutions at the interface with accuracy of order one, the interfaces II and III provide more accurate results than interface I. On the other hand, although solutions at the interface are obtained by high order polynomials for interface II, they may not satisfy the governing equations. This is not the case for interface III. As a result, the use of interface III gives more accurate results than interface II. From numerical experiment, it seems that the higher order accuracy of solutions at the interface may require fewer time steps, and therefore less operational time, to steady state resolution. Fig. 7.17 gives the velocities through the geometric centre of the cavity, where the array of 2×2 slave transputers and the local mesh size of 13×13 for Re=100, 17×17 for Re=400 in each subdomain, were used. Fig. 7.18 shows the non-dimensional operational time, where the reference time T_{ref} is the operational time taken by using interface I.



(a) Horizontal Velocity (b) Vertical Velocity

Fig. 7.17 Velocities Through the Geometric Centre of the Cavity

(for section 7.6.3.3)

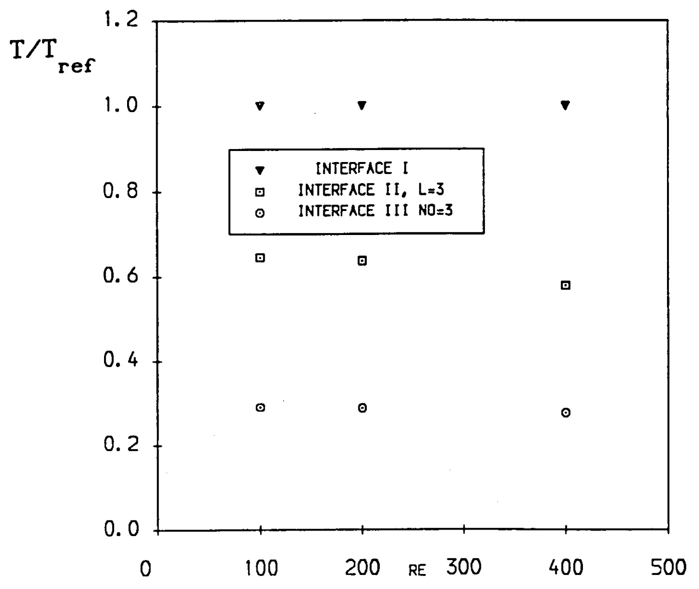
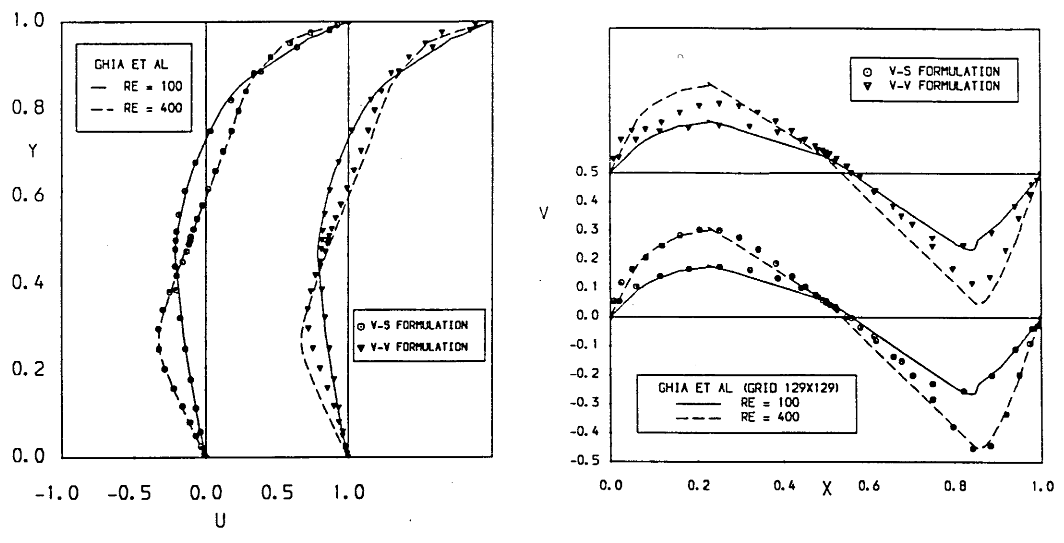


Fig. 7.18 Comparison of the Non-Dimensional Operational Time

(for section 7.6.3.3)

7.6.3.4 Vorticity-Velocity and Vorticity-Stream Function Formulations

We have compared the numerical results and the operational time using two formulations of the N-S equations, and found that the vorticity-stream function (V-S) formulation provides more accurate results and requires less operational time even though more time steps are taken, than the vorticity-velocity (V-V) formulation. To analyse this behaviour, it is noted that in the present research, two boundary conditions on the solid boundary are implemented for the solution of the stream function while only one boundary condition is used for the solution of each velocity component. This may be the reason that the vorticity-stream function gives better results. The vorticity equation for both V-S and V-V formulations are exactly the same, but the V-S formulation needs to solve only one Poisson equation for the stream function while the V-V formulation needs to solve two Poisson equations for two components of velocity. Thus although the V-S formulation needs more time steps to steady state resolution, it requires less operational time than the V-V formulation. Fig. 7.19 displays the velocities through the geometric centre of the cavity, where the array of 2x2 slave transputers and the local mesh size of 11x11 for Re=100, 17x17 for Re=400 in each subdomain, were used. Fig. 7.20 shows the non-dimensional operational time, where the reference time T_{ref} is the operational time taken by the V-V formulation.



(a) Horizontal Velocity (b) Vertical Velocity

Fig. 7.19 Velocities Through the Geometric Centre of the Cavity

(for section 7.6.3.4)

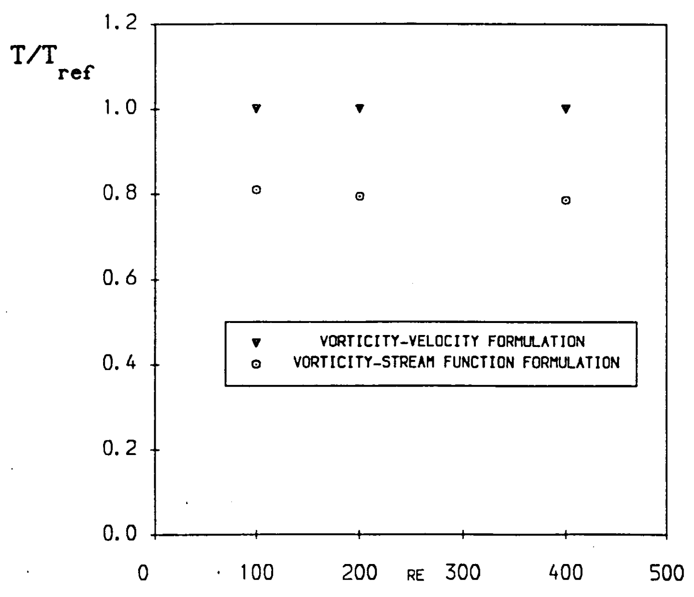


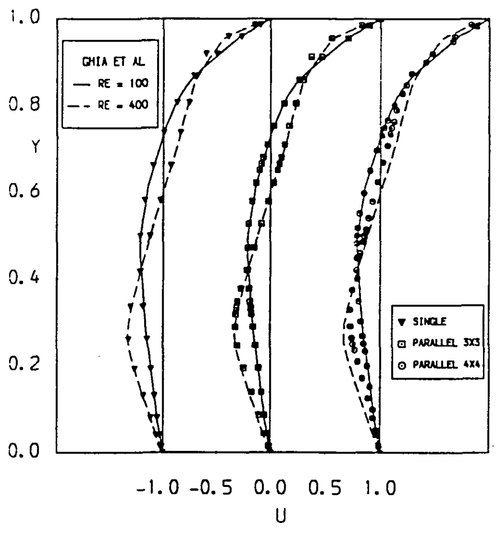
Fig. 7.20 Comparison of the Non-Dimensional Operational Time

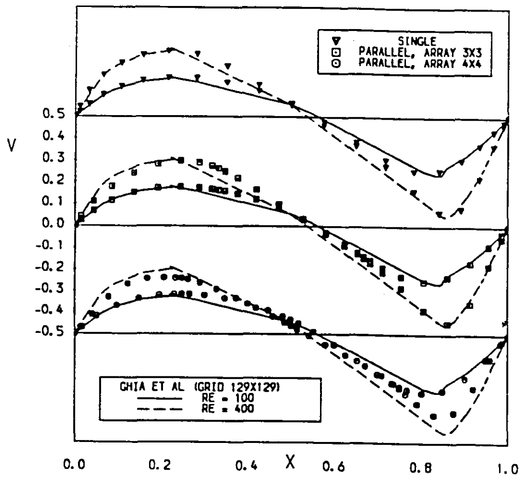
(for section 7.6.3.4)

7.6.3.5 The Global GDQ Approach and the Multi-Domain GDQ Approach

Compared with the global GDQ results, the multi-domain GDQ results, obtained by parallel computation, demonstrated that they are less accurate even though a larger number of grid points was actually used in the whole domain. This is because, on the one hand, although the total number of grid points in the full computational field for the multi-domain GDQ approach is larger than that for the global GDQ approach, the order of the discretization of derivatives in subdomains from the multi-domain GDQ approach is usually less than that from the global GDQ approach. Thus, the accuracy of solutions in subdomains is generally less than the accuracy of solutions obtained globally in the whole domain. On the other hand, the concept of multi-domain introduces the interface. Usually, the solutions at the interface are less accurate than those in the interior of subdomains. In other words, additional errors are introduced at the interface. As analysed in section 7.6.3.3, the error at the interface may spread into the interior of subdomains and reduce the accuracy of solutions in the whole domain. So, the increase of the number of interfaces may decrease the accuracy of results. Keeping the total number of grid points the same, as the number of subdomains increases, the number of interfaces increases and the accuracy of solutions in subdomains is reduced. As a result, the accuracy of solutions in the whole domain is reduced.

Another interesting result is that the total operational time taken by all the slave processors using the multi-domain GDQ approach may be less than that taken by a single processor using the global GDQ approach if the total number of grid points in the whole domain is kept the same. As shown in Chapter Four, the allowable time step sizes are determined by the eigenvalues of the spatial discretization matrix, and the eigenvalues increase as the number of grid points increases. Thus, if the total number of grid points in the overall domain is kept the same, the multi-domain GDQ approach may use much larger allowable time step sizes in subdomains than the global GDQ approach in the whole domain. Therefore, if one subdomain is allocated to one slave processor, each slave processor may take much less operational time to the convergent solutions and the total operational time may be less than that taken by a single processor. Fig. 7.21 displays the velocities through the geometric centre of the cavity, where the "single" indicates the global GDQ results using the mesh size of 21×19, and the "parallel 3x3 or 4x4" means that the array of 3x3 or 4x4 slave processors was used for the multi-domain GDQ results. The numerical solutions are obtained by using the vorticity-stream function formulation and interface I. For the Reynolds numbers of 100, 400, the local mesh size in each subdomain of 10×10 for the array of 3x3 slave processors, and 8x8 for the array of 4x4 slave processors, were used. Obviously, as the number of subdomains increases, although the total number of grid points is increased, the accuracy of solutions is reduced, especially for the high Reynolds number cases. Fig. 7.22 shows the non-dimensional operational time, where the reference time T_{ref} is the operational time taken by a single processor using the global GDQ approach with the mesh size of 21×19. This figure demonstrates that the total operational time taken by all the slave processors may be less than that taken by a single processor for some cases. For a given array of slave processors, if more grid points in each subdomain, e.g.





(a) Horizontal Velocity (b) Vertical Velocity

Fig. 7.21 Velocities Through the Geometric centre of the Cavity

(for section 7.6.3.5)

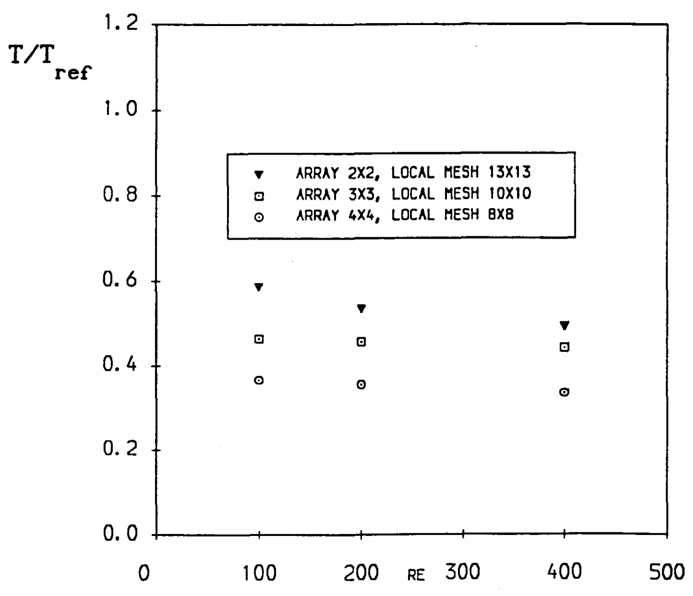


Fig. 7.22 Comparison of the Non-Dimensional Operational Time

(for section 7.6.3.5)

the local mesh size of 15×15 for the array of 2×2 slave processors, are used, the total operational time taken by all the slave processors may be larger than that taken by a single processor.

7.7 Concluding Remarks

The multi-domain GDQ approach has been shown to suit and to benefit from parallel computation. The computational efficiency of this approach on the distributed parallel computer is very high when the number of subdomains is not large. The overlapped interface provides the most accurate results and needs the least operational time, compared with the patched interface using the continuity condition or using the Lagrange interpolation scheme. If high order numerical approaches such as the GDQ approach are used in the local subdomains, it is better to use a high order scheme for the solutions at the interface to keep high order resolutions in the whole computational field. The vorticity-stream function formulation gives more accurate results and requires less operational time than the vorticity-velocity formulation. Keeping the number of grid points in the whole domain the same, as the number of subdomains increases, the accuracy of solutions is reduced, and the total operational time taken by all the slave processors using the multi-domain GDQ approach may be less than that taken by a single processor using the global GDQ approach. The general codes, which can be run on any array of processors without modification to the program, appear to offer· a way forward in engineering applications. The Meiko Computing Surface is a reconfigurable parallel computer, which is easy to use, and to control the placement of and the communication between processors.

CHAPTER EIGHT

Conclusions and Prospect of Future Research

8.1 Conclusions

The new numerical approaches of generalized differential quadrature (GDQ) and generalized integral quadrature (GIQ) have been developed in this research. Their application to the solutions of incompressible viscous flows without discontinuities involving parallel simulations was also reported. Some conclusions drawn from this research are listed as follows:

(1) The GDQ approach, developed in this research, is a global method, which overcomes the difficulty of differential quadrature (DQ) in obtaining the weighting coefficients for the discretization of the first derivative when the number of grid points is large and the distribution of grid points is arbitrary. The determination of the weighting coefficients for any order of derivatives are very easy. The weighting coefficients of the second or higher order derivatives are obtained from those of the first derivative by a recurrence relationship. Thus, the application of the GDQ approach in engineering may be more widespread than the DQ approach. Although the application of the GDQ approach is demonstrated for CFD in this research, it can also be applied to other engineering problems in continuum media.

(2) Some basic features of the GDQ approach such as arising in stability conditions differ from those of low order numerical schemes. In the GDQ approach, features, such as the allowable time step size, are dependent on the global characteristics of the problem such as the global distribution of grid points, while in low order numerical approaches such as finite differences, the features are dependent on the local characteristics of the problem. In particular, the GDQ approach requires the grid to be stretched near the boundary to ensure stability of solutions. A uniform grid or a grid stretched near the mid-point is not suitable for solving the incompressible N-S equations.

(3) It has been shown that the GDQ approach developed can be considered as the highest order finite difference scheme. It has also been shown that, both the GDQ approach and the spectral collocation method can give the same results if the same grid points are used for both approaches although they are based on different mathematical foundations. This is demonstrated by the fact that when the coordinates of grid points are chosen as the roots of a Chebyshev polynomial, both the GDQ approach and the spectral Chebyshev collocation method give the same weighting coefficients for the discretization of the first derivative.

(4) The GIQ approach developed is also a global method, where not only the integral of a function over the whole domain but also the integral of a function over a part of the whole domain can be approximated by the combination of all the functional values in the overall domain. They have the same order of numerical errors.

(5) The GDQ and GIQ approaches can be used with arbitrary distributions of grid points. Thus, they can be applied in a general coordinate system.

(6) Application of GDQ to the solutions of the incompressible N-S equations showed that the GDQ approach can achieve the same accuracy using just a few grid points and needs much less computational time, as compared with low order numerical approaches using a large number of grid points.

(7) Application of the GDQ-GIQ approach to the solutions of the boundary layer equations demonstrated that this approach is very efficient, and can achieve the same accuracy using just a few grid points as conventional schemes such as the Keller-box finite difference scheme. In the direction normal to the surface, the derivatives and the integral are discretized by the GDQ and GIQ approaches. In other directions, the derivatives can be discretized by the GDQ approach or by the local finite difference scheme. It is better to use the stream function as the dependent variable to achieve more accurate results and less computational time needed. The use of the GDQ approach in the streamwise direction is very sensitive to discontinuities such as a separation point.

(8) The multi-domain GDQ approach combines the ability to deal with geometric complexity along with high order accuracy. It provides a promising way to treat increasingly complex problems.

(9) For parallel simulation of the incompressible N-S equations using a

multi-domain GDQ approach, an overlapped interface gives more accurate results and needs less operational time than a patched interface. The vorticity-stream function formulation presents more accurate solutions and requires less operational time than the vorticity-velocity formulation. The computational efficiency is very high when a small number of subdomains and a multi-domain GDQ method are used. The increase of the number of subdomains may reduce the accuracy of results and the computational efficiency. For any two dimensional problem in engineering and using the multi-domain concept, the achievement of a general code, which can be run on any array of processors without modification to the program, is possible. Extension to three dimension is expected to be possible.

(10) The Meiko Computing Surface is a flexible, reconfigurable parallel computer, on which parallel computation is easily implemented.

8.2 Prospect of Future Research

The research work reported in this thesis is by no means complete. Indeed, both the application and the algorithms themselves are still in their infancy. Some topics for further research are suggested as:

(1) The application of the GDQ approach to other engineering problems such as in solid mechanics is of interest area in the future. In CFD, the application of the GDQ method to compressible flows, where shock waves may exist, is a challenging area. To capture the shock wave, available schemes adopted by other numerical methods, such as artificial

viscosity and filtering, need to be introduced.

(2) When the number of grid points is large, the explicit allowable time step size becomes extremely small, and hence, a large number of time steps is required for the convergent results. To speed up the convergence, an implicit scheme can be used, which allows the use of a bigger time step size, and therefore needs less time steps for convergence. To further accelerate the convergence, the multi-grid approach and some preconditioning schemes can be used. It is expected that the application of a multi-grid GDQ approach will be easy and straightforward since in the GDQ approach, the interpolation from fine to coarse or from coarse to fine grids is very easy and accurate.

(3) The application of the GDQ approach to turbulent flows, especially in the incompressible case, is another interest area where an appropriate turbulence model need be introduced.

(4) If more complex problems are tackled, the physical space will need to be transformed into computational space. Thus, grid generation schemes will be required, especially for the three-dimensional case.

(5) The development of the GDQ element method appears to be an attractive area in the future. This approach can also combine the geometric complexity with high order accuracy. The solutions over all the computational domain including the interface between elements are obtained globally through solutions of the governing equations.

(6) In mathematics, it has been proved that any smooth function in a domain can be approximated by an optimal polynomial. Since in the GDQ approach, the approximated polynomial is related to the distribution of grid points, there should exist a optimal distribution of grid points for a smooth problem. This may be achieved by using an adaptive grid approach.

(7) The application of a marching scheme of the GDQ-GIQ approach to the solution of the boundary layer equations could be fruitfully studied in the future. Two schemes could be tested. One scheme would apply the GDQ and GIQ methods only in the normal direction to the surface. In the streamwise or the crossflow direction, conventional local finite difference schemes, which make it possible to march the solution along the streamwise or the crossflow direction, would be used. Another scheme would divide the computational domain in the streamwise and crossflow directions into several subdomains (blocks). Thus, the surface would be composed of several blocks. In one block, the GDQ and GIQ approaches would be applied in all directions, that is, solutions in one block would be obtained globally. After getting all the solutions in a block, the solutions on the boundary of this block would be taken as the initial values for the solution of another neighbouring block. This procedure would continue by marching the solution block by block until all the solutions in the computational field are obtained.

(8) Some other functions such as triangular polynomials could be used as the base functions for the approximation of the problem. Triangular polynomials are well suited to the approximation of periodic problems.

If other functions are used, the formulation of the weighting coefficients for the discretization of derivatives could be obtained using a similar approach as in this thesis.

(9) The efficiency of parallel computation using the multi-domain GDQ approach, especially the study of the treatment of the interface, needs to be investigated further. Parallel simulation using the GDQ element method is another area of interest in parallel computation. In addition, parallel simulation using conventional finite differences, finite elements and finite volumes methods is also a future area of activity.

REFERENCES

Bar-Lev, M. and Yang, H.T., (1975), "Initial Flow Field over an Impulsively Started Circular Cylinder", *J. Fluid Mech.*, Vol. **72**, 625-647

Beam, R.M. and Warming, R.F., (1978), "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations", *AIAA J*, Vol. **16**, 393-402

Bellman, R., Kashef, B.G. and Casti, J., (1972), "Differential Quadrature : A Technique for the Rapid Solution of Nonlinear Partial Differential Equations", *J. Comput. Phys.* Vol.**10**, 40-52

Bellman, R., Naadimuthu, G., Wang, K.M. and Lee, E.S., (1984), "Differential Quadrature and Partial Differential Equations: Some Numerical Results", *J. Math. Anal. Appl.* Vol.**98**, 220-235

Bellman, R.E. and Roth, R.S., (1986), "Methods in Approximation", *D. Reidel Publishing Company*

Bodonyi, R.J. and Stewartson, K., (1977), "The Unsteady Laminar Boundary Layer on a Rotating Disk in a Counter-Rotating Fluid", *J. Fluid Mech.*, Vol. **79**, 669-688

Bredif, M., (1984), "Calculation of Laminar Flow over A Step by A Finite Element Method Based on the Stream Function-Vorticity Formulation", *Notes on Numerical Fluid Mechanics*, Vol. **9**, Vieweg, Braunschweig

Burggaf, O.R., (1966), "Analytical and Numerical Studies of the Structure of Steady Separated Flows", *J. Fluid Mech.* Vol.**24**, 113-151

Canuto, C., Hussaini, M.Y., Quarteroni, A., and Zang, T.A., (1987), "Spectral Methods in Fluid Dynamics", *Springer, New York, Berlin, Heidelberg, London, Paris, Tokyo*

Cebeci, T., (1975), "Calculation of Three-Dimensional Boundary Layers, II. Three-Dimensional Flows in Cartesian Coordinates", *AIAA J.* Vol. **13**, No. **8**, 1056-1064

Cebeci, T., (1979), "The Laminar Boundary Layer on a Circular Cylinder Started Impulsively from Rest", *J. Comput. Phys.* Vol. **31**, 153-172

Cebeci, T., Hirsh, R.S., Keller, H.B., and Williams, P.G., (1981), "Studies of Numerical Methods for the Plane Navier-Stokes Equations", *Computer Methods in Applied Mechanics and Engineering*, Vol. **27**, 13-44

Cebeci, T. (1986), "Unsteady Boundary Layers with an Intelligent Numerical Scheme", *J. Fluid Mech.*, Vol. **163**, 129

Chakravarthy, S.R. and Osher, S., (1985), "A New Class of High Accuracy TVD Schemes for Hyperbolic Conservation Laws", *AIAA Paper* 85-0360

Chan, D.C., Sindir, M.M., Gosman, A.D., (1987), "Numerical Flow Simulation of Passages with Strong Curvature and Rotation Using A Three-Dimensional Navier-Stokes Solver", *AIAA Paper* 87-1354

Chang, J.L. and Kwak, D., (1984), "On the Method of Pseudo Compressibility for Numerically Solving Incompressible Flows", *AIAA Paper* 84-0252

Chorin, A.J., (1967), "A Numerical Method for Solving Incompressible Viscous Flow Problems", *J. Comput. Phys.*, Vol.**2**, 12-26

Chung, T.J., (1978), "Finite Element Analysis in Fluid Dynamics", *New York, McGraw*

Civan, F. and Sliepcevich, C.M., (1983), "Application of Differential Quadrature to Transport Processes", *J. Math. Anal. Appl.*, Vol. **93**, 206-221

Civan, F. and Sliepcevich, C.M., (1984), "Differential Quadrature for Multi-Dimensional Problems", *J. Math. Anal. Appl.*, Vol. **101**, 423-443

Coutanceau, M. and Bouard, R., (1977), "Experimental Determination of the Main Features of the Viscous Flow in the Wake of A Circular Cylinder in Uniform Translation. Part I: steady Flow", *J. Fluid Mech.*, Vol. **79**, 231-256

Davis, P.J. and Rabinowitz, P., (1975), "Methods of Numerical Integration", *Academic Press, Inc. (London) Ltd.*

De Vahl Davis, G. and Jones, I.P, (1983), "Natural Convection in A Square Cavity: A Comparison Exercise", *Int. J. Numer. Methods Fluids*, Vol. **3**, 227-248

De Vahl Davis, G., (1983), "Natural Convection of Air in A Square Cavity: A Benchmark Numerical Solution", *Int. J. Numer. Methods Fluids*, Vol. **3**, 249-264

Dennis, S.C.R. and Chang, G., (1970), "Numerical Solutions for Steady Flow past A Circular Cylinder at Reynolds Numbers up to 100", *J. Fluid Mech.*, Vol. **42**, 471-489

Donea, J., Selmin, V. and Quartapelle, L., (1988), "Recent Developments of the Taylor-Galerkin Method for the Numerical Solution of Hyperbolic Problems", *In Numerical Methods for Fluid Dynamics,* (Morton and Baines ed.), Academic Press

Dwyer, H.A., (1968), "Solution of a Three-Dimensional Boundary Layer Flow with Separation", *AIAA J,* Vol. **6**, No.7, 1336-1342

Ehrenstein, U. and Peyret, R., (1989), "A Chebyshev Collocation Method for the Navier-Stokes Equations with Application to Double-Diffusive Convection", *Inter. J. Numer. Methods Fluids*, Vol. **9**, 427-452

Eiseman, P.R., (1982), "Coordinate Generation with Precise Controls Over Mesh Properties", *J. Comput. Phys.,* Vol. **47**, 331-351

Farouk and, B. and Fusegi, T., (1985), "A Coupled Solution of the Vorticity-Velocity Formulation of the Incompressible Navier-stokes Equations", *Inter. J. Numer. Methods. Fluids,* Vol. **5**, 1017-1034

Fasel, H. and Booz, O., (1984), "Numerical Investigation of Supercritical Taylor-Vortex Flow for a wide Gap", *J. Fluid Mech.,* Vol. **138**, 21-52

Ghia, U., Ghia, K.N., Rubin, S.G., and Khosla, P.K., (1981), "Study of Incompressible Flow Separation Using Primitive Variables", *Comput. Fluids*, Vol. **9**, 123-142

Ghia, U., Ghia, K.N. and Shin, C.T., (1982), "High-Re solutions for Incompressible Flow Using the Navier-Stokes Equations and A Multi-Grid Method", *J. Comput. Phys.*, Vol. **48**, 387-411

Gottlieb, D. and Orszag, S.A., (1977), "Numerical Analysis of Spectral

Methods: Theory and Applications", *SIAM-CBMS, Philadelphia*

Gresho, P.M., Chan, S.T., Lee, R.L. and Upson, C.D., (1984), "A Modified Finite Element Method for Solving the Time-Dependent, Incompressible Navier-Stokes Equations, Part II: Applications", *Inter. J. Numer. Methods Fluids*, Vol. **4**, 619-640

Gropp, W.D. and Smith, E.B., (1990), "Computational Fluid Dynamics on Parallel Processors", *Computers and Fluids*, Vol. **18**, No. **3**, 289-304

Harten, A., (1983), "High Resolution Schemes for hyperbolic Conservation Laws", *J. Comput. Phys.*, Vol. **49**, 357

Hassan, O., Morgan, K. and Peraire, J., (1989), "An Adaptive Implicit /Explicit Finite Element Scheme for Compressible Viscous High Speed Flows", *AIAA Paper* 89-0363

Hughes, T.J.R., Liu, W.K. and Brooks,A., (1979), "Finite Element Analysis of Incompressible Viscous Flows by the Penalty Function Formulation", *J. Comput. Phys.*, Vol. **30**, 1-60

Hughes, T.J.R., Tezduyar, T.E. and Brooks, A.N., (1982), "A Petrov-Galerkin Finite Element Formulation for Systems of Conservation Laws with Special Reference to Compressible Euler Equations", *In Numerical Methods for Fluid Dynamics,* (Morton and Baines ed.), Academic Press

Hussaini, M.Y. and Zang, T.A., (1987), "Spectral Methods in Fluid Dynamics", *Ann. Rev. Fluid Mech.*, Vol. **19**, 339-367

Jameson, A., Schmidt, W. and Turkel, E., (1981), "Numerical Solution of the Euler Equations by the Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes", *AIAA Paper* 81-1259

Jang, S.K., Bert, C.M. and Striz, A.G., (1989), "Application of Differential Quadrature to Static Analysis of Structural Components", *Inter. J. Numer. Methods Eng.*, Vol. **28**, 561-577

Johnson, G.M., (1983), "Multiple-Grid Convergence Acceleration of Viscous and Inviscid Flow Computations", *Appl. Math. Comp.*, Vol. **13**, 375-398

Keller, H.K. and Cebeci, T., (1970), "Accurate Numerical Methods for Boundary layer flows, I. Two Dimensional Laminar Flows", *Lecture Notes in Physics*, Vol. **8**, 92-100

Keller, H.B., (1974), "Accurate difference methods for Two-Point Boundary Value Problems", *SIAM J. Numer. Anal.* Vol. **11**, 305-320

Kim, J. and Moin, P., (1985), "Application of A Fractional-Step Method to Incompressible Navier-Stokes Equations", *J. Comput. Phys.*, Vol. **59**, 308-323

Korczak, K.Z. and Patera, A.T., (1986), "Isoparametric Spectral element Method for Solution of the Navier-Stokes Equations in Complex Geometry", *J. Comput. Phys.*, Vol. **62**, 361-382

Ku, H.C. and Hatziavramidis, D., (1985), "Solutions of the Two-Dimensional Navier-Stokes Equations by Chebyshev Expansion Methods", *Comput. Fluids*, Vol. **13**, 99-113

Kueny, J.L. and Binder, G., (1984), "Viscous Flow over Backward Facing Steps: An Experimental Investigation", *Notes on Numerical Fluid Mechanics*, Vol. **9**, Vieweg, Braunschweig

Leone, Jr., J.M. and Gresho, P.M., (1981), "Finite Element Simulations of Steady, Two-Dimensional, Viscous Incompressible Flow over A Step", *J. Comput. Phys.*, Vol. **41**, 167-191

Liakopulos, A., (1988), "Pseudospectral solutions of separated Flows", *AIAA Paper* 88-3643-CP

Lin, A., (1989), "Fast Parallel Algorithms for Computational Fluid Dynamics", *in Numerical Methods in Laminar and Turbulent Flows*, Vol. **6**

MacCormack, R.W., (1969), "The Effect of Viscosity in Hyper-velocity Impact Cratering", *AIAA Paper* 69-354

MacCormack, R.W. and Paullay, A.J., (1972), "Computational Efficiency Achieved by Time Splitting of Finite Difference operators", *AIAA Paper* 72-154

MacCormack, R.W., (1982), "A Numerical Method for Solving the Equations of Compressible Viscous Flows", *AIAA J.*, Vol. **20**, 1275-1281

McDonald, P.W., (1971), "The Computation of Transonic Flow Through Two-Dimensional Gas Turbine Cascades", *ASME Paper* 71-GT-89

Mingle, J.O., (1977), "The Method of Differential Quadrature for Transient Nonlinear Diffusion", *J. Math. Anal. Appl.* Vol.**60**, 559-569

Morgan, K., Periaux, J. and Thomasset, F., (Eds), (1984), "Analysis of Laminar Flow over A Backward Facing Step", *Notes on Numerical Fluid Mechanics*, Vol. **9**, Vieweg, Braunschweig

Morgan, K. and Peraire, J., (1987), "Finite Element Methods for Compressible Flows", *Von Karman Institute for Fluid Dynamics, Lecture Series*, 1987-04, Belgium

Morrison, J.H. and Napolitano, M., (1988), "Efficient Solutions of Two Dimensional Incompressible Steady Viscous Flows", *Comput. Fluids*, Vol. **16**, 119-132

Morton, K.W., (1982), "Generalised Galerkin Methods for Steady and Unsteady Problems", *In Numerical Methods for Fluid Dynamics*, (K.W. Morton and M.J. Baines ed.), Academic Press

Morton, K.W. and Paisley, M.F., (1989), "A Finite Volume Scheme with Shock Fitting for the Steady Euler Equations", *J. Comput. Phys.* Vol. **80**, 168

Ni, R.H., (1982), "A Multiple Grid Scheme for Solving the Euler Equations", *AIAA J*, Vol. **20**, 1565-1571

Orszag, S.A., (1980), "Spectral Methods for Problems in Complex Geometries", *J. Comput. Phys.*, Vol. **37**, 70-92

Osswald, G.A. and Ghia, K.N., (1985), "An Implicit Time-Marching Method for Studying Unsteady Flow with Massive Separation", *AIAA Paper* 85-1489

Patankar, S.V. and Spalding, D.B., (1972), "A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows", *Int. J. Heat Transfer*, Vol. **15**, 1787-1806

Patera, A.T., (1984), "A Spectral Element Method for Fluid Dynamics: Laminar Flow in A Channel Expansion", *J. Comput. Phys.*, Vol. **54**, 468-488

Phillips, T.N., (1984), "Natural Convection in an Enclosed Cavity", *J. Comput. Phys.*, Vol. **54**, 365-381

Phillips, T.N. and Karageorghis, A., (1988), "Chebyshev Collocation Methods for the Solution of the Incompressible Navier-Stokes Equations in Complex Geometries", *In Numerical Methods for Fluid Dynamics III* (K.W. Morton and M.J. Baines ed.), Clarendon Press, Oxford

Pike, J. and Roe, P.L., (1985), "Accelerated Convergence of Jameson's Finite Volume Euler Scheme Using Van Der Houwen Integrators", *Comput. Fluids*, Vol. **13**, 223-236

Pulliam, T.H., (1986), "Artificial Dissipation Models for the Euler Equations", *AIAA J.* Vol. **24**, 1931-1940

Richtmeyer, R.D. and Morton, K.W., (1967), "Difference Methods for Initial Value Problems", *John Wiley, New York*

Rizzi, A.W., and Inouye, M., (1973), "Time split Finite Volume Method for Three-Dimensional Blunt-Body Flows", *AIAA J.*, Vol. **11**, 1478-1485

Rizzi, A.W. and Eriksson, L.E., (1983), "Explicit Multistage Finite Volume Procedure to Solve the Euler Equations for Transonic Flow", *Lecture Series on Computational Fluid Dynamics*, Von Karman Institute for Fluid Dynamics, Belgium

Rizzi, A.W. and Eriksson, L.E., (1985), "Computation of Inviscid Incompressible Flow with Rotation", *J. Fluid Mech.*, Vol. **153**, 275-312

Roe, P.L., (1981), "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", *J. Comput. Phys.*, Vol. **43**, 357-372

Schreiber, R. and Keller, H.K., (1983), "Driven Cavity Flows by Efficient Numerical Techniques", *J. Comput. Phys.*, Vol. **49**, 310-333

Shu, C. and Richards, B.E., (1990), "High Resolution of Natural Convection in A Square Cavity by Generalized Differential Quadrature", *in Proceedings of 3rd Int. Conf. on Advances in Numer. Methods in Engineering: Theory and Applications*, Swansea, U.K., Jan. 1990, (eds. by Pande and Middleton), Vol. **II**, 978-985

Steger, J.L. and Kutler, P., (1976), "Implicit Finite Difference Procedures for the Computation of Vortex Wakes", *AIAA Paper* 76-385

Steger, J.L. and Sorenson, R.C., (1980), "Use of Hyperbolic Partial Differential Equations to Generate Body Fitted Coordinates", *Numerical Grid Generation Techniques, Proceedings of A Workshop*, NASA Langley Research Centre

Steger, J.L. and Warming, R.F., (1981), "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods", *J. Comput. Phys.*, Vol. **40**, 263-293

Stone, H.L., (1968), "Iterative Solutions of Implicit Approximations of Multi-Dimensional Partial Differential Equations", *SIAA Journal of Numerical Analysis*, Vol. **5**, No.3

Streett, C.L., Zang, T.A. and Hussaini, M.Y., (1984), "Spectral Methods for Solution of the Boundary Layer Equations", *AIAA Paper* 84-0170

Streett, C.L., (1987), "Spectral Methods and Their Implementation to Solution of Aerodynamic and Fluid Mechanic Problems", *Int. J. Numer. Methods Fluids*, Vol. **7**, 1159-1189

Stuben, K. and Trottenberg, U., (1982), "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Application", *Lect. Notes in Math.*, Vol. **960**, 1-176

Swanson, R.C.,and Turkel, E., (1985), "A Multistage Time-Stepping Scheme for the Navier-Stokes Equations", *AIAA Paper* 85-035

Thomas, C.E., Morgan, K. and Taylor, C., (1981), "A Finite Element Analysis of Flow over A Backward Facing Step", *Comput. Fluids*, Vol. **9**, 265-278

Thompson, J.F., (1978), "Numerical Solution of Flow Problems Using

Body-Fitted Coordinate Syetems", *Lecture Series in Computational Fluid Dynamics*, Von Karman Institute for Fluid Dynamics, Belgium

Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W., (1982), "Boundary Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review", *J. Comput. Phys.*, Vol. **47**, 1-108

Tritton, D.J., (1959), "Experiments on the Flow past A Circular Cylinder at Low Reynolds Numbers", *J. Fluid Mech.*, Vol. **16**, No. **4**

Van Dommelen, L.L. and Shen, S.F., (1982), "The Genesis of Separation", in Numerical and Physical Aspects of Aerodynamic Flows, *Springer-Verlag*

Van Leer, B., (1982), "Flux-Vector Splitting for the Euler Equations", *Lect. Notes in Phys.*, Vol. **170**, 507-511

Werle, M.J., and Bertke, S.D., (1972), "A Finite Difference Method for Boundary Layers with Reverse Flow", *AIAA J.* Vol. **10**, 1250-1252

Wu, J.C., Sankar, N.L. and Sugavanam, A., (1978), "A Numerical Study of Unsteady Viscous Flows around Airfoils", *Proc. Symp. on Unsteady Aerodynamics,* AGARD CP-227, Paper 24

Wu, J.C., (1984), "Fundamental Solutions and Numerical Methods for Flow Problems", *Int. J. Numer. Methods in Fluids*, Vol. **4**, 185-201

Yang, C. and Atluri, S.N., (1984), "An 'Assumed Deviatoric Stress-Pressure-Velocity' Mixed Finite Element Method for Unsteady, Convective, Incompressible Viscous Flow, Part II: Computational Studies", *Inter. J. Numer. Methods Fluids*, Vol. **4**, 43-69

Yee, H.C., (1987), "Construction of Explicit and Implicit Symmetric TVD Schemes and Their Applications", *J. Comput. Phys.*, Vol. **68**, 151

BIBLIOGRAPHY

Anderson, D.A., Tannehill, J.C. and Pletcher, R.H., (1984), "Computational Fluid Mechanics and Heat Transfer", *McGraw-Hill, New York*

Bercovier, M. and Engelman, M., (1979), "A Finite Element for the Numerical Solution of Viscous Incompressible Flows", *J. Comput. Phys.*, Vol. **30**, 181-201

Cebeci, T. and Smith, A.M.O., (1977), "Analysis of Turbulent Boundary Layers", *McGraw-Hill-Hemisphere, Washington, D.C.*

Jiang, D.C., (1986), "Prediction of Shock/Turbulent Boundary Layer Separated Flows Using the Navier-Stokes Equations", Ph. D. Thesis, Dept. of Aerospace Eng., University of Glasgow

Kopal, Z., (1955), "Numerical Analysis", *London, Chapman and Hall Ltd.*

Pulliam, T.H. and Steger, J.L., (1985), "Recent Improvements in Efficiency, Accuracy, and Convergence for Implicit Approximate Factorization Algorithms", *AIAA Paper* 85-0360

Qin, N., (1987), "Towards Numerical Simulation of Hypersonic Flow around Space-Plane Shapes", Ph. D. Thesis, Dept. of Aerospace Engineering, University of Glasgow

Riley, N., (1975), "Unsteady Boundary Layers", *SIAM Rev.* Vol. **17**, 274

Rinaldo, A. and Giorgini, A., (1984), "A Mixed Algorithm for the Calculation of Rapidly Varying Fluid Flows: the Impulsively Started Circular Cylinder", *Inter. J. Numer. Methods Fluids*, Vol. **4**, 949-969

Streett, C.L., Zang, T.A. and Hussaini, M.Y., (1985), "Spectral Multigrid Methods with Applications to Transonic Potential Flow", *J. Comput. Phys.*, Vol. **57**, 43-76

Wang, Z., (1990), "Numerical Simulation Of 3D Hypersonic Flow Using High Resolution Schemes", Ph. D. Thesis, Dept. of Aerospace Eng., University of Glasgow

APPENDIX A

Weierstrass Theorem and Definition of A Vector Space

Weierstrass Theorem : *Let $f(x)$ be a real valued continuous function defined in the closed interval* [0, 1], *there exist a sequence of polynomials $P_n(x)$ which converge to $f(x)$ uniformly as n goes to infinity.*

The proof of this theorem is shown in many text books. For details, see the book of Bellman and Roth (1986).

A vector space V over a field F is defined as follows: it is a set of elements called vectors such that any two vectors α and β of V determine a (unique) vector $\alpha+\beta$ as sum and that any vector α from V and c from F determine a vector $c\alpha$ in V with the properties:

(1) Associativity : $\alpha + (\beta + \psi) = (\alpha + \beta) + \psi$

(2) Identity : there is a vector 0 in V such that $\alpha + 0 = \alpha$ for all α

(3) Inverse Law: there is a solution in V to the equation $\alpha + x = 0$, that is in V

(4) Commutative Law : for all α and β in V, $\alpha + \beta = \beta + \alpha$

(5) Distributive Law : $c(\alpha + \beta) = c\alpha + c\beta$

(6) $(ab)\alpha = a(b\alpha)$, where a, b are in F

(7) Unity : $1\alpha = \alpha$

APPENDIX B

Application of The GDQ Approach in the Domain $[x_{i-1}, x_{i+1}]$

As an example, we will show that the discretization of the first order derivative by the GDQ approach in the domain $[x_{i-1}, x_{i+1}]$ is the same as that given from the second order finite difference scheme. Clearly, the domain $[x_{i-1}, x_{i+1}]$ includes three grid points x_{i-1}, x_i, x_{i+1}, and it is known that any smooth function in this domain can be approximated by a polynomial of degree 2, which constitutes a 3-dimensional polynomial vector space. Thus the weighting coefficients of the first derivative for this specific case can be determined as follows according to formulation (3.7), (3.9)

$$M(x) = (x-x_{i-1})\cdot(x-x_i)\cdot(x-x_{i+1}) \tag{B.1}$$

$$M^{(1)}(x_{i-1}) = (x_{i-1}-x_i)\cdot(x_{i-1}-x_{i+1}) \tag{B.2}$$

$$M^{(1)}(x_i) = (x_i-x_{i-1})\cdot(x_i-x_{i+1}) \tag{B.3}$$

$$M^{(1)}(x_{i+1}) = (x_{i+1}-x_i)\cdot(x_{i+1}-x_{i-1}) \tag{B.4}$$

and

$$a_{i,i-1} = -\Delta_2/[(\Delta_1+\Delta_2)\Delta_1] \tag{B.5}$$

$$a_{i,i+1} = \Delta_1/[(\Delta_1+\Delta_2)\Delta_2] \tag{B.6}$$

$$a_{i,i} = (\Delta_2-\Delta_1)/(\Delta_1\cdot\Delta_2) \tag{B.7}$$

where

$$\Delta_1 = x_i - x_{i-1}$$

$$\Delta_2 = x_{i+1} - x_i$$

Hence, the first derivative of a function f can be approximated as

$$f_x(x_i) = \sum_{j=-1}^{1} a_{i,i+j}\cdot f(x_j) \tag{B.8}$$

It is easy to show that (B.8) is exactly the same as that from the second order finite difference scheme and if the grid is uniform, (B.8) can be reduced to

$$f_x(x_i) = 0.5[f(x_{i+1}) - f(x_{i-1})]/\Delta \qquad \text{(B.9)}$$

where $\Delta = \Delta_1 = \Delta_2$

which is the same as used in the finite difference scheme. In the same manner, the discretization of the first derivative at x_{i-1} and x_{i+1} can be written as

$$f_x(x_{i-1}) = -\frac{2\Delta_1+\Delta_2}{(\Delta_1+\Delta_2)\Delta_2} f(x_{i-1}) + \frac{\Delta_1+\Delta_2}{\Delta_1\Delta_2} f(x_i) - \frac{\Delta_1}{(\Delta_1+\Delta_2)\Delta_2} f(x_{i+1}) \qquad \text{(B.10)}$$

$$f_x(x_{i+1}) = \frac{\Delta_2}{(\Delta_1+\Delta_2)\Delta_1} f(x_{i-1}) - \frac{\Delta_1+\Delta_2}{\Delta_1\Delta_2} f(x_i) + \frac{2\Delta_2+\Delta_1}{(\Delta_1+\Delta_2)\Delta_1} f(x_{i+1}) \qquad \text{(B.11)}$$

which are exactly the same as those from the second order finite difference scheme. For the overall domain case, it is suggested that such a domain can be divided into N-1 elements with grid points, x_1, $\cdots$, x_N. At location x_i, i = 2, 3, $\cdots$, N-1, the first order derivative of a function can be discretized by (B.8) in the element $[x_{i-1}, x_{i+1}]$. It is noted that in the case here, the two neighbouring elements $[x_{i-1}, x_{i+1}]$ and $[x_i, x_{i+2}]$, used for the discretization of the first derivative at collocation points x_i and x_{i+1}, are overlapped with the region of $[x_i, x_{i+1}]$. This behaviour is different from the standard finite element approach where the neighbouring elements are patched. Similarly, at x_1 and x_N, the discretization of the first order derivative of a function can be obtained by (B.10) in the element $[x_1, x_3]$ and by (B.11) in the element $[x_{N-2}, x_N]$. It can be concluded that any higher order finite difference scheme can be designed using this technique in a straight forward way.

APPENDIX C

Time-Split MacCormack Finite Difference Scheme

For demonstration, we consider a two-dimensional Burger equation as

$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \qquad \text{(C.1)}$$

The time-split MacCormack scheme "splits" the original MacCormack scheme into a sequence of one-dimensional operations thereby achieving a less restrictive stability condition. The one-dimensional difference operators $L_x(\Delta t_x)$ and $L_y(\Delta t_y)$ are defined as

$$u^*_{i,j} = L_x(\Delta t_x)u^n_{i,j} \qquad \text{(C.2)}$$

which is equivalent to the two-step formula

$$\bar{u}^*_{i,j} = u^n_{i,j} - \frac{\Delta t_x}{\Delta x}(F^n_{i+1,j} - F^n_{i,j}) + \mu\cdot\Delta t_x\cdot\delta^2_x u^n_{i,j} \qquad \text{(C.3)}$$

$$u^*_{i,j} = \frac{1}{2}\left[u^n_{i,j} + \bar{u}^*_{i,j} - \frac{\Delta t_x}{\Delta x}(\bar{F}^*_{i,j} - \bar{F}^*_{i-1,j}) + \mu\cdot\Delta t_x\cdot\delta^2_x \bar{u}^*_{i,j}\right] \qquad \text{(C.4)}$$

where $\delta^2_x u_{i,j} = \dfrac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2}$

and $u^*_{i,j} = L_y(\Delta t_y)u^n_{i,j}$ (C.5)

which is equivalent to the following two-step formula

$$\bar{u}^*_{i,j} = u^n_{i,j} - \frac{\Delta t_y}{\Delta y}(G^n_{i,j+1} - G^n_{i,j}) + \mu\cdot\Delta t_y\cdot\delta^2_y u^n_{i,j} \qquad \text{(C.6)}$$

$$u^*_{i,j} = \frac{1}{2}\left[u^n_{i,j} + \bar{u}^*_{i,j} - \frac{\Delta t_y}{\Delta y}(\bar{G}^*_{i,j} - \bar{G}^*_{i,j-1}) + \mu\cdot\Delta t_y\cdot\delta^2_y \bar{u}^*_{i,j}\right] \qquad \text{(C.7)}$$

where $\delta^2_y u_{i,j} = \dfrac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2}$

A second order time-split MacCormack scheme can be constructed by applying the L_x and L_y operators to $u^n_{i,j}$ in the following manner

$$u^{n+1}_{i,j} = L_y\left(\frac{\Delta t}{2}\right)L_x(\Delta t)L_y\left(\frac{\Delta t}{2}\right)u^n_{i,j} \qquad \text{(C.8)}$$

This scheme has a truncation error of $O[(\Delta t)^2, (\Delta x)^2, (\Delta y)^2]$. Generally, a scheme constructed by a sequence of one-dimensional operators has the following features: (1) stable, if the time step of each operator does not exceed the allowable time step size for that operator; (2) consistent, if the sum of the time steps for each of the operators are equal; (3) second order accurate, if the sequence is symmetric. Accordingly, other sequences can be formed, for example,

$$u_{i,j}^{n+1} = [L_y(\frac{\Delta t}{2})L_x(\frac{\Delta t}{2})L_x(\frac{\Delta t}{2})L_y(\frac{\Delta t}{2})]^m u_{i,j}^n \qquad (C.9)$$

where m is an integer.

APPENDIX D

The SIP (Strongly Implicit Procedure) and Modified SIP

The strongly implicit procedure is presented by Stone (1968) for solving the algebraic equations arising from the numerical solution of second order partial differential equations by a finite difference scheme. To illustrate this approach, we consider a set of algebraic equations arising from the use of the second order finite difference scheme for Poisson's equations as

$$B_{i,j}u_{i,j-1}+D_{i,j}u_{i-1,j}+E_{i,j}u_{i,j}+F_{i,j}u_{i+1,j}+H_{i,j}u_{i,j+1}=R_{i,j} \qquad (D.1)$$

which can be written in matrix form as

$$[A]\mathbf{u} = \mathbf{r} \qquad (D.2)$$

where [A] is a matrix with five nonzero diagonals, **u** is a vector of unknowns, and **r** is a vector of known quantities. The SIP scheme is to replace the matrix [A] by a modified form [A+P] such that the modified matrix can be decomposed into upper and lower triangular sparse matrices denoted by [U] and [L]. An iterative procedure is defined by writing (D.2) as

$$[A+P]\mathbf{u}^{n+1} = \mathbf{r} + [P]\mathbf{u}^{n} \qquad (D.3)$$

Decomposing [B] = [A+P] into the upper and lower triangular matrices [U] and [L], (D.3) can be written as

$$[L][U]\mathbf{u}^{n+1} = \mathbf{r} + [P]\mathbf{u}^{n} \qquad (D.4)$$

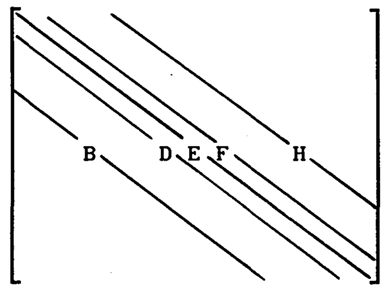
Defining an intermediate vector as $\mathbf{V}^{n+1} = [U]\mathbf{u}^{n+1}$, the following two-step algorithm can be formed

step 1: $[L]\mathbf{V}^{n+1} = \mathbf{r} + [P]\mathbf{u}^{n}$ (D.5)

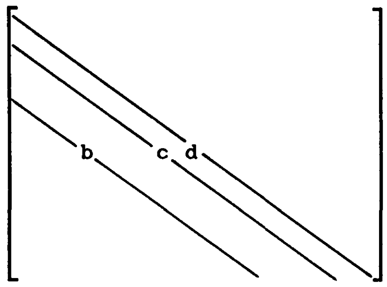
step 2: $[U]\mathbf{u}^{n+1} = \mathbf{V}^{n+1}$ (D.6)

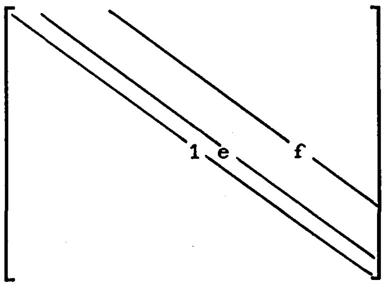
This procedure is repeated iteratively. Stone (1968) selected [P] so that [L] and [U] have only three nonzero diagonals with the principal diagonal of [U] being the unity diagonal. Furthermore, the elements of [L] and [U] are determined such that the coefficients in the matrix [B] in the locations of the nonzero entries of matrix [A] are identical with those in [A]. Two additional nonzero diagonals appear in [B]. The procedure is implicit in both the x and y directions. The details of this procedure are given by Stone (1968). Here some results are listed. With the form of matrix [A] as

[A] =



the matrix [L] and [U] can be written as

[L] = 

, [U] = 

The coefficients are determined by the following formula

$$b_{i,j} = \frac{B_{i,j}}{1+\alpha\cdot e_{i,j-1}} \qquad \text{(D.7)}$$

$$c_{i,j} = \frac{D_{i,j}}{1+\alpha\cdot f_{i-1,j}} \qquad \text{(D.8)}$$

$$d_{i,j}=E_{i,j}+\alpha(b_{i,j}\cdot e_{i,j-1}+c_{i,j}\cdot f_{i-1,j}) - b_{i,j}\cdot f_{i,j-1} - c_{i,j}\cdot e_{i-1,j} \qquad \text{(D.9)}$$

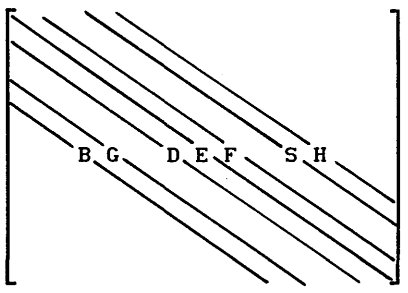
$$e_{i,j} = \frac{F_{i,j} - \alpha\cdot b_{i,j} e_{i,j-1}}{d_{i,j}} \qquad \text{(D.10)}$$

$$f_{i,j} = \frac{H_{i,j} - \alpha\cdot c_{i,j} f_{i-1,j}}{d_{i,j}} \qquad \text{(D.11)}$$

Here α is a parameter. When the program operates in the relaxation mode, α can be varied according to the following relationship

$$\alpha = 1 - (\mathrm{Der})^{n/N} \quad , \qquad n = 0,\ 1,\ \cdots,\ N \qquad \text{(D.12)}$$

where N is the maximum number of elements in the α sequence, Der = $1-\alpha_{max}$
We have developed the modified strongly implicit procedure when the matrix [A] has a form

[A] = 

For this case, the matrices [L] and [U] have the same form as shown above, but the formula for the determination of the coefficients are different, which are given as follows

$$b_{i,j} = \frac{B_{i,j}+\alpha\cdot G_{i,j}}{1+\alpha\cdot e_{i,j-1}} \tag{D.13}$$

$$c_{i,j} = \frac{D_{i,j}+\alpha\cdot S_{i,j}}{1+\alpha\cdot f_{i-1,j}} \tag{D.14}$$

$$d_{i,j}=E_{i,j}+\alpha(b_{i,j}\cdot e_{i,j-1}+c_{i,j}\cdot f_{i-1,j}- G_{i,j} - S_{i,j}) - b_{i,j}\cdot f_{i,j-1}- c_{i,j}\cdot e_{i-1,j} \tag{D.15}$$

$$e_{i,j} = \frac{F_{i,j} - \alpha\cdot b_{i,j}e_{i,j-1}+\alpha\cdot G_{i,j}}{d_{i,j}} \tag{D.16}$$

$$f_{i,j} = \frac{H_{i,j} - \alpha\cdot c_{i,j}f_{i-1,j}+\alpha\cdot S_{i,j}}{d_{i,j}} \tag{D.17}$$

APPENDIX E

LU Decomposition Approach

A set of algebraic equations can be written as a form of matrix

$$[A]\mathbf{X} = \mathbf{b} \tag{E.1}$$

The matrix [A] can be decomposed into two upper and lower triangular matrices [U] and [L], which is written as

$$[A] = [L]\ [U] \tag{E.2}$$

Introducing an intermediate vector **Y**, and setting

$$[U]\mathbf{X} = \mathbf{Y} \tag{E.3}$$

(E.1) can be reduced to

$$[L]\mathbf{Y} = \mathbf{b} \tag{E.4}$$

The solutions of (E.4) and (E.3) can be written as

$$Y_i = b_i - \sum_{j=1}^{i-1} L_{ij}Y_j \quad , \qquad i = 1,\ 2,\ \cdots,\ N \tag{E.5}$$

$$X_i = (Y_i - \sum_{j=i+1}^{N} U_{ij}X_j)/U_{ii} \quad , \qquad i = N, N-1, \cdots, 1 \qquad \text{(E.6)}$$

The components of L_{ij} and U_{ij} can be calculated as

$$U_{ij} = A_{ij} - \sum_{m=1}^{i-1} L_{im}U_{mj} \quad , \qquad (j \ge i) \qquad \text{(E.7)}$$

$$L_{ij} = (A_{ij} - \sum_{m=1}^{j-1} L_{im}U_{mj})/U_{jj} \quad , \qquad (i > j) \qquad \text{(E.8)}$$

APPENDIX F

Drag Coefficient for the Flow past A Circular Cylinder

To calculate the total drag coefficient on the surface of a cylinder, we need to calculate the surface pressure distribution firstly which can be obtained by integrating the tangential component of the N-S equations, for the case here, that is

$$p = \frac{1}{\mathrm{Re}} \int_0^{\xi} \omega_{\eta}\, d\xi \quad , \qquad \text{with } p(0) = 0 \qquad \text{(F.1)}$$

The pressure coefficient c_p is equal to 2p on the surface, the tangential stress is given by $\sigma = \omega/\mathrm{Re}$. Thus the total drag coefficient can be written as

$$C_D = C_{DP} + C_{DF} \qquad \text{(F.2)}$$

with

$$C_{DP} = 2 \int_0^{2\pi} p \cdot y_{\xi} \cdot d\xi \qquad \text{(F.3)}$$

$$C_{DF} = - \frac{2}{\mathrm{Re}} \int_0^{2\pi} \omega \cdot x_{\xi} \cdot d\xi \qquad \text{(F.4)}$$

GLASGOW UNIVERSITY LIBRARY