Temple University
Doctoral Dissertation
Submitted to the Graduate Board

Title of Dissertation:
(Please type) Studies of Some High Order Finite/Spectral Element Methods for Viscous Incompressible Flow

Author:
(Please type) Jianjun Xu

Date of Defense:
(Please type) April 20, 2001

Dissertation Examining Committee: (please type)	Read and Approved By: (Signatures)
Dr. Jian-Guo Liu Dissertation Advisory Committee Chairperson	[illegible]
Dr. Daniel B. Szyld	[illegible]
Dr. Wei-Shih Yang	[illegible]
Dr. Yuan Shi	[illegible]
Dr. Eric Grinberg Examining Committee Chairperson	[illegible] If Member of the Dissertation Examining Committee

Date Submitted to Graduate Board: 5-11-01

Accepted by the Graduate Board of Temple University in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Date 5/31/01 [illegible]
(Dean of the Graduate School)

STUDIES OF SOME HIGH ORDER FINITE/SPECTRAL ELEMENT METHODS FOR VISCOUS INCOMPRESSIBLE FLOW

A Dissertation
Submitted to
the Temple University Graduate Board

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

by
Jian-Jun Xu
August, 2001

ABSTRACT

STUDIES OF SOME HIGH ORDER FINITE/SPECTRAL ELEMENT METHODS FOR VISCOUS INCOMPRESSIBLE FLOW

Jian-Jun Xu
DOCTOR OF PHILOSOPHY
Temple University, August, 2001
Professor Jian-Guo Liu, Chair

The topic of this thesis focuses on high order finite/spectral element methods for solving unsteady incompressible viscous Navier-Stokes equations. The gauge formulation of the Navier-Stokes equations was proposed by Weinan E and Jian-Guo Liu recently [10]. The main advantage is that the gauge variable is nonphysical, so we have the freedom to assign boundary conditions.

In many cases, the low order methods are not able to catch up the complicated flow structures, to achieve good accuracy for the pressure term, or the divergence free condition. In this thesis, we focus on developing high order finite/spectral element methods for solving Navier-Stokes equations.

Based on the gauge formulation, several high order finite/spectral element methods based on the gauge formulation are developed. For the time stepping procedures, backward Euler and Crank-Nicholson methods are used. The numerical experiments show clean high order accuracy. Some high order time-stepping procedures such as the backward differentiation methods or the explicit forth-order Runge-Kutta method have also been tried. In all these methods, the computations of the gauge variable and the auxiliary field are decoupled, with the nonlinear term treated explicitly. All the computations are reduced to solving heat equations and Poisson equations, so they are very efficient.

Based on the vorticity-stream function formulation of the Navier-Stokes equations, Jian-Guo Liu and Weinan E [22] proposed an efficient time-stepping procedure recently so that the computations of the vorticity function and the stream function are decoupled, with the nonlinear term treated explicitly. The second part of the thesis is the implementation of the procedure by using high order finite/spectral element methods for space discretization. The numerical experiments are presented including clean high order accuracy and the simulations of the canonical driven cavity flows.

Preconditioned conjugated gradient (CG) methods are used to solving the resulting linear systems. We have studied the convergence property of the CG method when applied to symmetric positive semi-definite systems.

To exploit the sparsity structures of the stiffness matrix and mass matrix, the local assembly technologies are used to evaluate the nonlinear terms, various right hand sides, the gradients and the matrix-vector product.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Professor Jian-Guo Liu for his guidance and support.

I also would like to thank Professors Daniel Szyld, Eric Grinberg, John J. Schiller, Wei-Shih Yang, Yuan Shi, Shifraw Berhanu and Hans Johnston for their support and guidance in the past few years.

Dedicated to my mother Yun-Lan Wang

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENT v

DEDICATION vi

LIST OF FIGURES ix

LIST OF TABLES xii

1 INTRODUCTION 1

2 HIGH ORDER GAUGE FINITE/SPECTRAL ELEMENT METHODS 6
- 2.1 Semi-Discrete Methods 8
- 2.2 Full Discrete Methods 10
- 2.3 Is There Any Higher-Order Time Stepping Procedure? 14
  - 2.3.1 Stability Regions of the Mixed ABAM Methods 15
  - 2.3.2 Stability Regions of Backward Differentiation Methods 18
  - 2.3.3 Is There Any Higher Order Time-Stepping Procedure? 21

3 SIMPLE FINITE/SPECTRAL ELEMENT METHODS 23
- 3.1 Semi-Discrete Method 26
- 3.2 Full Discrete Scheme 27

4 FAST POISSON SOLVERS 30
- 4.1 The Conjugate Gradient (CG) Method 31
- 4.2 Preconditioned Conjugate Gradient Methods 34
- 4.3 Discrete Trigonometric Transforms as Preconditioners 36
- 4.4 A Curved Boundary Problem 41

5 COMPUTATION OF INTEGRALS AND ASSEMBLY TECHNIQUES 43
5.1 Assembly of the Nonlinear Terms 44
5.2 Unassembled Stiffness Matrix 48
5.3 Unassembled Mass Matrix 50
5.4 Assembly of Matrix-Vector Product 51
5.5 Evaluation of the Gradient Terms in Gauge Method 51

6 NUMERICAL EXPERIMENTS 55
6.1 Accuracy Check for the Gauge Finite/Spectral Element Methods 55
6.2 Accuracy Check for the Simple Finite/Spectral Element Methods 57
6.3 Simulations of the Driven-Cavity Flows 62

REFERENCES 68

A MAPLE CODES FOR THE COMPUTATION OF SOME INTEGRALS ON THE REFERENCE TRIANGLE 72

B MAPLE CODES FOR THE COMPUTATION OF SOME INTEGRALS OF HIGH ORDER SPECTRAL ELEMENT METHODS ON THE REFERENCE RECTANGLE 76

LIST OF FIGURES

2.1 The stability region of the ABAM method of first-order, method is stable in the left side of the curve 16
2.2 The stability region of the ABAM method of second-order, method is stable in the left side of the curve 17
2.3 The stability region of the ABAM method of third-order, method is stable inside the curve 17
2.4 The stability region of the ABAM method of forth-order, method is stable inside the curve 17
2.5 The stability region of the ABAM method of fifth-order, method is stable inside the curve 19
2.6 The stability region of the backward-differentiation method of first-order, method is stable in the left side of the curve 19
2.7 The stability region of the backward-differentiation method of second-order, method is stable in the left side of the curve . . 19
2.8 The stability region of the backward-differentiation method of third-order, method is stable in the left side of the curve . . . 20
2.9 The stability region of the backward-differentiation method of forth-order, method is stable in the left side of the curve . . . 20
2.10 The stability region of the backward-differentiation method of fifth-order, method is stable in the left side of the curve 20

4.1 A curved boundary domain 41

6.1 Error of the first component of velocity for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order, Re=1000, t=1, cell number=5^2 58
6.2 Error of the second component of velocity for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order, Re=1000, t=1 , cell number=5^2 58

6.3 Error of pressure for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order, Re=1000, t=1 , cell number=5^2 59
6.4 Error of divergence-free property for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order, Re=1000, t=1 , cell number=5^2 59
6.5 Error of the stream function in 5th-order simple spectral element method, Re=1000, t=1 , cell number=10^2 61
6.6 Error of the vorticity function in 5th-order simple spectral element method, Re=1000, t=1, cell number=10^2 61
6.7 stream function, time=5, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 62
6.8 stream function, time=10, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 62
6.9 stream function, time=15, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 63
6.10 vorticity function, time=5, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 63
6.11 vorticity function, time=10, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 63
6.12 vorticity function, time=15, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 64
6.13 stream function, time=1, Re=100000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=25^2 64
6.14 vorticity function, time=1, Re=100000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=25^2 64
6.15 steady state of stream function,Re=1000, discontinuous boundary condition, 5th order simple spectral rectangle element method, cell number=15^2 65
6.16 steady state of stream function, Re=7500, discontinuous boundary condition, 5th order simple spectral rectangle element method, cell number=25^2 65

6.17 steady state of stream function, Re=10000, discontinuous boundary condition, 5th order simple spectral rectangle element method. cell number=25^2 66

6.18 steady state of vorticity function, Re=1000, discontinuous boundary condition, 5th order simple spectral rectangle element method. cell number=15^2 66

6.19 steady state of vorticity function, Re=7500, discontinuous boundary condition, 5th order simple spectral rectangle element method. cell number=25^2 67

6.20 steady state of vorticity function, Re=10000, discontinuous boundary condition, 5th order simple spectral rectangle element method. cell number=25^2 67

LIST OF TABLES

2.1 Coefficients for ABAM methods 15
2.2 Coefficients for Backward-Differentiation methods 18

4.1 Accuracy when DST as preconditioner for Poisson equation on a curved domain (third-order triangle elements) 42

6.1 Errors for Backward-Euler Gauge Finite Element Methods (mixed 3rd/4th order triangle elements), Re=1000, t=1 56
6.2 Errors for Crank-Nicholson Gauge Finite Element Methods (mixed third/forth-order triangle finite elements), Re=1000, t=1 . . . 56
6.3 Errors for Crank-Nicholson Gauge Spectral Element Methods (mixed 5th/6th order rectangle elements), Re=1000, t=1 . . . 57
6.4 Errors for Simple Finite Element Methods (third-order triangle finite element method), Re=1000, t=1 60
6.5 Errors for Simple Spectral Element Methods (fifth-order rectangle elements), Re=1000, t=1 61

CHAPTER 1

INTRODUCTION

The motion of fluid flow is governed by the principles of classical mechanics and thermodynamics for the conservation of mass. momentum. and energy. Applications of these principals lead to the conservation equations in integral form for mass. momentum. and energy. The properties of the flow need not be continuous functions of space and time: If the physics properties of the flow are continuous and sufficiently differentiable in some domain of space and time. then the conservation integral equations can be transformed into an equivalent set of partial differential equations-the Navier-Stokes equations [6].

The topic of my dissertation focuses on finding the numerical solutions of the Navier-Stokes equations for incompressible viscous flow. of which the primitive-variable formulation in two-dimensional case reads as follows:

$$(1.1) \qquad \begin{cases} \frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} + \nabla p = \frac{1}{Re}\Delta\vec{u} + \vec{f} & \text{in} \quad \Omega \\ \nabla\cdot\vec{u} = 0 & \text{in} \quad \Omega \end{cases}$$

where $\vec{u} = (u_1, u_2)$ is the velocity field and p is the pressure. with the simplest physical boundary condition:

$$(1.2) \qquad \vec{u}|_\Gamma = 0$$

where $\Gamma = \partial\Omega$. The difficulties in the numerical computation of (1.1) are the lack of an evolution equation for pressure and the implementation of the incompressibility condition($div(\vec{u}) = 0$). Taking divergence on the first equation leads to a pressure Poisson equation,

$$\Delta p = \nabla \cdot (\vec{u} \cdot \nabla \vec{u}) + \nabla \cdot \vec{f} \tag{1.3}$$

The task of solving (1.1) would become much easier if we could attach a simple boundary condition to this equation. Unfortunately the most natural candidate obtained from extending (1.1) to the boundary:

$$\frac{\partial p}{\partial \vec{n}} = \frac{1}{Re} \vec{n} \cdot \Delta \vec{u} + \vec{f} \cdot \vec{n} \tag{1.4}$$

involves evaluating the viscous term at the boundary, where $\vec{n}$ is the unit outward normal direction. Not only it is difficult to enforce this condition accurately, maintaining consistency between (1.3) and (1.4) (since this is Neumann problem) in a discrete setting is also very difficult. The projection method, which was invented by Chorin [5] and Teman [33] independently in late sixties, by-passes the issue of the pressure boundary condition, see also [16]. The price has been paid is that some special discretization schemes have to be used to discretize the pressure equation. This seriously limits the simplicity and flexibility of the projection method.

Recently E & Liu [10] proposed a new formulation–gauge formulation– for the Navier-Stokes equations. The key point of the gauge formulation is to replace the pressure by a gauge variable ϕ. The main advantage of the gauge formulation is that the gauge variable is non-physical, so we have the freedom to assign boundary condition for it.

There are already many existing low order methods for solving the Navier-Stokes equations. In many cases, however, the low order methods are not able to catch up the complicated flow structures, or to achieve good accuracy for the pressure term or the divergence free condition.

High order methods are necessary to overcome these drawbacks of low order methods. High order finite/spectral element methods are more promising than high order finite difference methods, because the boundary conditions are very difficult to be satisfied for high order finite difference methods, while the variational formulations of finite/spectral element methods have the natural enforcement of boundary conditions. In this thesis, several high order finite/spectral element methods based on the gauge formulation are presented. For the time stepping procedures, backward Euler and Crank-Nicholson methods are used. The numerical experiment shows clean high order accuracy.

It is natural to ask if we can have any higher order time stepping procedure which matches the high order finite element space discretization. We have considered the mixed Adams-Bashforth-Adams-Molton methods and the backward differentiation methods, see e.g. [12, 19]. We have tried the third-order backward differentiation gauge finite element method, unfortunately our numerical experiments show that these schemes are not stable in the gauge formulation. We have also tried the explicit forth-order Runge-Kutta method for time marching, not stable either. So it is an open problem what are high order time stepping procedures for the high order gauge finite element methods.

The two-dimensional incompressible Navier-Stokes equations in vorticity-stream function formulation read follows:

$$(1.5) \qquad \begin{cases} \frac{\partial \omega}{\partial t} + (\vec{u}\cdot\nabla)\omega = \frac{1}{Re}\Delta\omega + f_1 & \text{in} \quad \Omega \\ \Delta\psi = \omega & \text{in} \quad \Omega \end{cases}$$

with the boundary conditions $\psi|_\Gamma = f_2, \frac{\partial\psi}{\partial\vec{n}}|_\Gamma = f_3$, where $\vec{u} = \nabla^\perp\psi \equiv (-\frac{\partial\psi}{\partial y}, \frac{\partial\psi}{\partial x})$, and $\vec{n}$ is the unit outward normal direction.

Note that in (1.5) the vorticity function ω and the stream function ψ are coupled together, much confusion and complexity would come if their computation still coupled together in a numerical method for solving (1.5).

Also note that there is no explicit boundary conditions for the vorticity function, this could cause difficulty when one constructs finite difference methods , especially for curved boundary problems.

Finite element methods for solving (1.5) are more promising, since the variational formulation of (1.5) has natural enforcement of boundary conditions. There have been some finite element methods for the steady Navier-Stokes equations, i.e. the time independent problem, see [1, 2, 4, 18, 30]. But in all the schemes, the convection term is treated implicitly. so the computation of the vorticity and stream functions are fully coupled.

To avoid this coupling, it is natural to think to treat the convection term explicitly. Then the issue of stability becomes crucial in choosing time stepping procedure.

Recently E and Liu ([8, 9, 22]) discovered the efficient time stepping procedures–the explicit high order Runge-Kutta methods, which allow the convection term and diffusion term are treated explicitly, and they argued that stability is obtained under standard Courant-Fridrich-Lewy(CFL) condition in both finite difference and finite element setting.

In this thesis, we implement the high order simple finite/spectral element methods, i.e., we are going to use the explicit forth-order Runge-Kutta method for the time marching and high order finite/spectral element methods for the space discretization. Numerical experiments including clean high order accuracy and the simulations of the canonical Driven-Cavity flow are presented. The thesis is organized as follows.

The gauge finite/spectral element methods for the Navier-Stokes equations are derived in chapter 2. In chapter 3 we derive the simple finite/spectral element methods for the vorticity-stream function formulation of the Navier-Stokes equations.

In chapter 4 we study the preconditioned conjugate gradient methods for solving the resulting large sparse linear systems. In the gauge finite/spectral element methods, the linear systems arise in the discretization of a heat-like equation, a Poisson-Neumann problem. and the evaluation of the gradient of the gauge variable. The linear system corresponding to the Poisson-Neumann problem is symmetric, but only positive semi-definite. We will show that the convergence properties of conjugate gradient method when applied to the symmetric positive semi-definite system. In the simple finite/spectral element methods, in each stage of the Runge-Kutta time marching procedure, a Poisson-Dirichlet problem needs to be solved. To accelerate the convergence rate of the conjugate gradient method, we use the preconditioning techniques by using the discrete trigonometric transforms(see e.g. [34, 31]) as preconditioners. We want to emphasize that as preconditioners the discrete trigonometric transforms are still good choices even if the computational domain is not rectangular. The numerical computation of the integrals and the assembly techniques for both triangle and rectangle elements are described in chapter 5. Finally we present the results of the numerical experiments in chapter 6, including accuracy checking and simulations of driven cavity flows.

CHAPTER 2

HIGH ORDER GAUGE FINITE/SPECTRAL ELEMENT METHODS

We start with the velocity-pressure formulation of the incompressible Navier-Stokes equations

$$\begin{cases} \frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} + \nabla p = \frac{1}{Re}\Delta \vec{u} + \vec{f} & \text{in} \quad \Omega \\ \nabla \cdot \vec{u} = 0 & \text{in} \quad \Omega \end{cases} \tag{2.1}$$

where $\vec{u} = (u, v)$ is the velocity field and p is the pressure, with the simplest physical boundary condition:

$$\vec{u}|_{\Gamma} = 0 \tag{2.2}$$

where $\Gamma = \partial\Omega$.

A new approach-Gauge method- was introduced by E & Liu [10] in 1996. The key point of the gauge method is to replace pressure by a gauge variable ϕ. They write the velocity field as a sum of two parts: $\vec{u} = \vec{a} + \nabla\phi$, and use $\vec{a}$ to take care of the convective term and $\nabla\phi$ to take care of the boundary condition and incompressibility constraint.

Then we have a new formulation of (2.1)

$$\tag{2.3} \begin{cases} \frac{\partial \vec{a}}{\partial t} - \frac{1}{Re}\Delta\vec{a} + (\vec{u}\cdot\Delta)\vec{u} = \vec{f} \\ -\Delta\phi = \nabla\cdot\vec{a} \\ \vec{u} = \vec{a} + \nabla\phi \end{cases}$$

And the pressure can be recovered as following:

$$\tag{2.4} p = \frac{\partial\phi}{\partial t} - \frac{1}{Re}\Delta\phi$$

The idea of gauge formulation dated back to 1989, when Oseledets [24] first introduced an impulse variable to reformulate Euler equations in a Hamiltonian system. It has received considerable attention since then. Buttke [3] used an impulse variable to design a numerical method for incompressible fluid flow in 1993. E and Liu [11] found that the velocity impulse formulation of Buttke is marginally ill-posed for the inviscid flow and they presented numerical evidence of this instability. Maddocks and Pego [23] used an impulsive variable to formulate an unconstrained Hamiltonian for the Euler equation in 1995. Russo and Smereka [28] studied the connection of different impulse/gauge formulation in 1999.

The main advantage of the gauge method (2.3) is that we have the freedom to assign boundary condition for ϕ, since it is a non-physical variable. Corresponding to (2.2), for example, one can either prescribe the Neumann boundary condition

$$\tag{2.5} \frac{\partial\phi}{\partial\vec{n}} = 0, \quad \vec{a}\cdot\vec{n} = 0, \quad \vec{a}\cdot\vec{\tau} = \frac{\partial\phi}{\partial\vec{\tau}}$$

or the Dirichlet boundary condition

$$\tag{2.6} \phi = 0, \quad \vec{a}\cdot\vec{n} = \frac{\partial\phi}{\partial\vec{n}}, \quad \vec{a}\cdot\vec{\tau} = 0$$

on Γ. Here $\vec{\tau}$ is the unit vector in the tangential direction, and $\vec{n}$ is the unit vector in the outward normal direction.

2.1 Semi-Discrete Methods

We first discretize time, keep space continuous. We consider two schemes: backward Euler method and Crank-Nicholson method.

Given the values $\vec{a}^n, \vec{u}^n, \vec{u}^{n-1}, \phi^n, \phi^{n-1}$, we have the the following methods to produce values of $\vec{a}, \vec{u}, \phi$ at the $(n+1)$th time step by using either the Dirichlet boundary condition or the Neumann boundary condition :

Method 2.1. *Backward Euler gauge method with Neumann boundary condition*

$$\text{Step. 1}\begin{cases} \frac{\vec{a}^{n+1}-\vec{a}^n}{\Delta t} + (\vec{u}^n\cdot\nabla)\vec{u}^n = \frac{1}{Re}\Delta\vec{a}^{n+1} + \vec{f}^{n+1} & \text{in } \Omega \\ \vec{a}^{n+1}\cdot\vec{n} = 0, \quad \text{on } \Gamma \\ \vec{a}^{n+1}\cdot\vec{\tau} = -\frac{\partial\phi^n}{\partial\vec{\tau}} \quad \text{on } \Gamma \end{cases}$$

$$\text{Step 2.}\begin{cases} -\Delta\phi^{n+1} = \nabla\cdot\vec{a}^{n+1}, & \text{in } \Omega \\ \frac{\partial\phi^{n+1}}{\partial\vec{n}} = 0, \quad \text{on } \Gamma \end{cases}$$

$$\text{Step 3.} \quad \vec{u}^{n+1} = \vec{a}^{n+1} + \nabla\phi^{n+1}$$

$$\text{Step 4.} \quad p^{n+1} = \frac{\phi^{n+1}-\phi^n}{\Delta t} - \frac{1}{Re}\Delta\phi^{n+1}.$$

Method 2.2. *Backward Euler gauge method with Dirichlet boundary condition*

$$\text{Step1.}\begin{cases} \frac{\vec{a}^{n+1}-\vec{a}^n}{\Delta t} + (\vec{u}^n\cdot\nabla)\vec{u}^n = \frac{1}{Re}\Delta\vec{a}^{n+1} + \vec{f}^{n+1} & \text{in } \Omega \\ \vec{a}^{n+1}\cdot\vec{\tau} = 0, \quad \text{on } \Gamma \\ \vec{a}^{n+1}\cdot\vec{n} = -\frac{\partial\phi^n}{\partial\vec{n}} \quad \text{on } \Gamma \end{cases}$$

$$\text{Step2.}\begin{cases} -\Delta\phi^{n+1} = \nabla\cdot\vec{a}^{n+1}, & \text{in } \Omega \\ \phi^{n+1} = 0, \quad \text{on } \Gamma \end{cases}$$

$$\text{Step3.} \quad \vec{u}^{n+1} = \vec{a}^{n+1} + \nabla\phi^{n+1}$$

$$\text{Step 4.} \quad p^{n+1} = \frac{\phi^{n+1}-\phi^n}{\Delta t} - \frac{1}{Re}\Delta\phi^{n+1}.$$

From step 2 and step 3 in Method 2.2, we have by divergence theorem

$$
(2.7)\qquad
\begin{aligned}
-\int_\Omega \Delta\phi^{n+1}d\Omega &= -\int_\Gamma \frac{\partial\phi^{n+1}}{\partial\vec{n}}ds\\
=\int_\Gamma \vec{a}^{n+1}\cdot\vec{n}ds &= -\int_\Gamma \frac{\partial\phi^{n}}{\partial\vec{n}}ds\\
= -\int_\Omega \Delta\phi^{n}d\Omega &
\end{aligned}
$$

This means that $\int_\Omega \Delta\phi^{n+1}d\Omega$ does not change for all the time steps. In general this is not true for the exact solution of the Navier-Stokes equations. So the numerical solution of Method 2.2 will not converges to the exact solution. The numerical solution of p will not either, since it is given by (2.4). But it is claimed that the velocity $\vec{u}^n$ converges to the exact solution, see [26]. Also see [35] for the setting of finite difference method.

It is an open problem how to fix this incompatibility problem for the Dirichlet boundary condition. And for this reason, we will use the Neumann boundary condition. Now let us use the second-order Crank-Nicholson method for the time discretization.

Given the values $\vec{a}^n, \vec{u}^n, \vec{u}^{n-1}, \phi^n, \phi^{n-1}$, we have the the following method to produce values of $\vec{a}, \vec{u}, \phi$ at the $(n+1)$th time step

Method 2.3. *Crank-Nicholson gauge method with Neumann boundary condition*

$$
\text{Step1}\begin{cases}
\frac{\vec{a}^{n+1}-\vec{a}^n}{\Delta t} + [\frac{3}{2}(\vec{u}^n\cdot\nabla)\vec{u}^n - \frac{1}{2}(\vec{u}^{n-1}\cdot\nabla)\vec{u}^{n-1}]\\
= \frac{1}{Re}\Delta(\frac{\vec{a}^n+\vec{a}^{n+1}}{2}) + \vec{f}^{n+\frac{1}{2}} \quad \text{in } \Omega\\
\vec{a}^{n+1}\cdot\vec{n} = 0, \quad \text{on } \Gamma\\
\vec{a}^{n+1}\cdot\vec{\tau} = -(2\frac{\partial\phi^n}{\partial\vec{\tau}} - \frac{\partial\phi^{n-1}}{\partial\vec{\tau}}) \quad \text{on } \Gamma
\end{cases}
$$

$$
\text{Step2.}\begin{cases}
-\Delta\phi^{n+1} = \nabla\cdot\vec{a}^{n+1}, \quad \text{in } \Omega\\
\frac{\partial\phi^{n+1}}{\partial\vec{n}} = 0, \quad \text{on } \Gamma
\end{cases}
$$

$$
\text{Step3.}\quad \vec{u}^{n+1} = \vec{a}^{n+1} + \nabla\phi^{n+1}
$$

$$
\text{Step 4.}\quad p^{n+1} = \frac{(1.5\phi^{n+1} - 2\phi^n + 0.5\phi^{n-1})}{\Delta t} - \frac{1}{Re}\Delta\phi^{n+1}
$$

The equation in Step1 can be rewritten as

$$(1-\frac{\Delta t}{2Re}\Delta)\vec{a}^{n+1} = (1+\frac{\Delta t}{2Re}\Delta)\vec{a}^{n} - [\frac{3}{2}(\vec{u}^{n}\cdot\nabla)\vec{u}^{n} - \frac{1}{2}(\vec{u}^{n-1}\cdot\nabla)\vec{u}^{n-1}]\Delta t + \vec{f}^{n+\frac{1}{2}}\Delta t$$

Remark. In all of the above methods, the boundary conditions for the auxiliary field $\vec{a}$ are implemented explicitly via extrapolation. The resulting momentum equation is decoupled from the kinematic equation, and the computational cost in each time step is reduced to solving a standard heat equation and a Poisson equation.

2.2 Full Discrete Methods

There are already many existing low order methods for the space discretization. In many cases, however, these low order methods always lead to large error in the divergence of the velocity, especially when there is not enough resolution. They are not able to catch up the complicated flow structures, to achieve good accuracy for the pressure term, or the divergence free condition.

High order methods are necessary to overcome these drawbacks of low order methods. High order finite/spectral element methods are more promising than high order finite difference methods, because the boundary conditions are very difficult to be satisfied for high order finite difference methods, while the variational formulations of finite/spectral element methods have natural enforcement of boundary conditions.

Let H^1, H_0^1 be the standard Sobolev spaces. Then we now give variational formulations of method 2.1 and method 2.3.

Given the values $\vec{a}^n, \vec{u}^n, \phi^n, \phi^{n-1}$, corresponding to Method 2.1 we have

Variational formulation for backward Euler gauge method with Neumann *boundary condition*

$$\text{Step 1.}\begin{cases}\text{Find } \vec{a}^{n+1} \in H^1 \times H^1, \text{ such that} \\ (\vec{a}^{n+1}, \psi) + \frac{\Delta t}{Re}(\nabla \vec{a}^{n+1}, \nabla \psi) = \\ (\vec{a}^n, \psi) - ((\vec{u}^n \cdot \nabla)\vec{u}^n, \psi)\Delta t + (\vec{f}^{n+1}, \psi)\Delta t \\ \qquad\qquad\qquad\qquad \forall \psi \in H_0^1 \\ \vec{a}^{n+1} \cdot \vec{n} = 0, \quad \text{on} \quad \Gamma \\ \vec{a}^{n+1} \cdot \vec{\tau} = -\frac{\partial \phi^n}{\partial \vec{\tau}} \quad \text{on} \quad \Gamma \end{cases}$$

$$\text{Step 2.}\begin{cases}\text{Find } \phi \in H^1 \text{ such that} \\ (\nabla \phi^{n+1}, \nabla \psi) = -(\nabla \cdot \vec{a}^{n+1}, \psi), \forall \psi \in H^1 \end{cases}$$

$$\text{Step 3.}\begin{cases}\text{Find } \vec{u} \in H^1 \times H^1 \text{ such that} \\ (\vec{u}^{n+1}, \psi) = (\vec{a}^{n+1}, \psi) + (\nabla \phi^{n+1}, \psi), \\ \qquad\qquad\qquad\qquad \forall \psi \in H^1 \end{cases}$$

$$\text{Step 4.}\begin{cases}\text{Find } p^{n+1} \in H^1 \text{ such that} \\ (p^{n+1}, \psi) = (\frac{\phi^{n+1}-\phi^n}{\Delta t}, \psi) + \frac{1}{Re}(\nabla \phi^{n+1}, \nabla \psi), \\ \qquad\qquad\qquad\qquad \forall \psi \in H^1. \end{cases}$$

Corresponding to method 2.3, we have

Variational formulation for Crank-Nicholson gauge method with Neumann boundary condition

$$\text{Step1}\begin{cases}\text{Find } \vec{a}^{n+1} \in H^1 \times H^1, \text{ such that} \\ (\vec{a}^{n+1}, \psi) + \frac{\Delta t}{2Re}(\nabla \vec{a}^{n+1}, \nabla \psi) = \\ (\vec{a}^n, \psi) - \frac{\Delta t}{2Re}(\nabla \vec{a}^n, \nabla \psi) - \\ -(\frac{3}{2}(\vec{u}^n \cdot \nabla)\vec{u}^n - \frac{1}{2}(\vec{u}^{n-1} \cdot \nabla)\vec{u}^{n-1}, \psi)\Delta t + \frac{1}{2}(\vec{f}^{n+1} + \vec{f}^n, \psi)\Delta t \\ \qquad\qquad\qquad\qquad \forall \psi \in H_0^1 \\ \vec{a}^{n+1} \cdot \vec{n} = 0, \quad \text{on} \quad \Gamma \\ \vec{a}^{n+1} \cdot \vec{\tau} = -(2\frac{\partial \phi^n}{\partial \vec{\tau}} - \frac{\partial \phi^{n-1}}{\partial \vec{\tau}}) \quad \text{on} \quad \Gamma \end{cases}$$

$$\text{Step2.}\begin{cases}\text{Find } \phi \in H^1 \text{ such that} \\ (\nabla \phi^{n+1}, \nabla \psi) = -(\nabla \cdot \vec{a}^{n+1}, \psi), \forall \psi \in H^1 \end{cases}$$

$$\text{Step3.}\begin{cases} \text{Find } \vec{u} \in H^1 \times H^1 \text{ such that} \\ (\vec{u}^{n+1}, \psi) = (\vec{a}^{n+1}, \psi) + (\nabla\phi^{n+1}, \psi), \\ \hfill \forall\psi \in H^1 \end{cases}$$

$$\text{Step 4.}\begin{cases} \text{Find } p^{n+1} \in H^1 \text{ such that} \\ (p^{n+1}, \psi) = (\frac{1.5\phi^{n+1}-2\phi^n+0.5\phi^{n-1}}{\Delta t}, \psi) + (\nabla\phi^{n+1}, \nabla\psi). \\ \hfill \forall\psi \in H^1. \end{cases}$$

Suppose that Ω is a polygon, and we have a triangulation or rectangulation T_h of Ω. Let $G_h = \{x_i\}$ be the set of all grid points.

Let $X^m_{0,h} \subset H^1_0$ and $X^m_h \subset H^1$ be the standard finite element spaces that consist of piecewise mth order polynomials on Ω. Let Π_h be the standard interpolation operator from C^1 to X^m_h:

$$\Pi_h(f)(x_i) = f(x_i), \forall x_i \in G_h. \tag{2.8}$$

Given the values $\vec{a}^n_h, \vec{u}^n_h, \phi^n_h$, corresponding to method 2.1, we have the values of $\vec{a}_h, \vec{u}_h, \phi_h$ at the $(n+1)$th time step as follows:

Method 2.4. *Backward Euler gauge finite element method with Neumann boundary condition*

$$\text{Step1}\begin{cases} \text{Find } \vec{a}^{n+1}_h \in X^k_h \times X^k_h, \text{ such that} \\ (\vec{a}^{n+1}_h, \psi) + \frac{\Delta t}{Re}(\nabla\vec{a}^{n+1}_h, \nabla\psi) = \\ (\vec{a}^n_h, \psi) - ((\vec{u}^n_h \cdot \nabla)\vec{u}^n, \psi)\Delta t + (\Pi_h\vec{f}^{n+1}, \psi)\Delta t \\ \hfill \forall\psi \in X^k_{0,h} \\ \vec{a}^{n+1}_h \cdot \vec{n} = 0, \quad \text{on } \Gamma \\ \vec{a}^{n+1}_h \cdot \vec{\tau} = -\Pi_h(\frac{\partial\phi^n_h}{\partial\vec{\tau}}) \quad \text{on } \Gamma \end{cases}$$

$$\text{Step2.}\begin{cases} \text{Find } \phi^{n+1}_h \in X^{k+1}_h \text{ such that} \\ (\nabla\phi^{n+1}_h, \nabla\psi) = -(\nabla \cdot \vec{a}^{n+1}_h, \psi), \forall\psi \in X^{k+1}_h \end{cases}$$

$$\text{Step3.}\begin{cases} \text{Find } \vec{u}^{n+1}_h \in X^k_h \times X^k_h \text{ such that} \\ (\vec{u}^{n+1}_h, \psi) = (\vec{a}^{n+1}_h, \psi) + (\nabla\phi^{n+1}_h, \psi), \\ \hfill \forall\psi \in X^k_h \end{cases}$$

$$\text{Step 4.}\begin{cases}\text{Find } p_h^{n+1} \in X_h^k \text{ such that} \\ (p_h^{n+1}, \psi) = (\frac{\phi_h^{n+1}-\phi_h^n}{\Delta t}, \psi) + \frac{1}{Re}(\nabla\phi_h^{n+1}, \nabla\psi), \\ \qquad\qquad \forall\psi \in X_h^k.\end{cases}$$

Note that we do not need to evaluate the divergence of $\vec{a}^{n+1}$ for the right hand side of step 2. Indeed,

$$\begin{aligned}&\int_\Omega \nabla\cdot\vec{a}_h^{n+1}\psi d\Omega = \\ &\int_\Omega \nabla\cdot(\psi\vec{a}_h^{n+1})d\Omega - \int_\Omega \nabla\psi\cdot\vec{a}_h^{n+1}d\Omega = \\ &= \int_\Gamma \psi\vec{a}_h^{n+1}\cdot\vec{n}ds - \int_\Omega \nabla\psi\cdot\vec{a}_h^{n+1}d\Omega \\ &= -\int_\Omega \nabla\psi\cdot\vec{a}_h^{n+1}d\Omega\end{aligned} \tag{2.9}$$

Given the values $\vec{a}_h^n, \vec{u}_h^n, \phi_h^n, \vec{u}_h^{n-1}, \psi_h^{n-1}$, corresponding to method 2.3, we have the values of $\vec{a}_h, \vec{u}_h, \phi_h$ at the $(n+1)$th time step as follows:

Method 2.5. *Crank-Nicholson gauge finite element method with Neumann boundary condition*

$$\text{Step1}\begin{cases}\text{Find } \vec{a}_h^{n+1} \in X_h^k \times X_h^k, \text{ such that} \\ (\vec{a}_h^{n+1}, \psi) + \frac{\Delta t}{2Re}(\nabla\vec{a}_h^{n+1}, \nabla\psi) = \\ (\vec{a}_h^n, \psi) - \frac{\Delta t}{2Re}(\nabla\vec{a}_h^n, \nabla\psi) - \\ -([\frac{3}{2}(\vec{u}_h^n\cdot\nabla)\vec{u}_h^n - \frac{1}{2}(\vec{u}_h^{n-1}\cdot\nabla)\vec{u}_h^{n-1}], \psi)\Delta t + \frac{1}{2}(\Pi_h\vec{f}^{n+1} + \Pi_h\vec{f}^n, \psi)\Delta t \\ \qquad\qquad \forall\psi \in X_{0,h}^k \\ \vec{a}_h^{n+1}\cdot\vec{n} = 0, \quad \text{on } \Gamma \\ \vec{a}_h^{n+1}\cdot\vec{\tau} = -\Pi_h(2\frac{\partial\phi_h^n}{\partial\vec{\tau}} - \frac{\partial\phi_h^{n-1}}{\partial\vec{\tau}}) \quad \text{on } \Gamma\end{cases}$$

$$\text{Step2.}\begin{cases}\text{Find } \phi_h^{n+1} \in X_h^{k+1} \text{ such that} \\ (\nabla\phi_h^{n+1}, \nabla\psi) = -(\nabla\cdot\vec{a}_h^{n+1}, \psi), \forall\psi \in X_h^{k+1}\end{cases}$$

$$\text{Step 3.}\begin{cases}\text{Find } \vec{u}_h^{n+1} \in X_h^k \times X_h^k \text{ such that} \\ (\vec{u}_h^{n+1}, \psi) = (\vec{a}_h^{n+1}, \psi) + (\nabla\phi_h^{n+1}, \psi), \\ \qquad\qquad \forall\psi \in X_h^k\end{cases}$$

$$\text{Step 4.}\begin{cases}\text{Find } p_h^{n+1} \in X_h^k \text{ such that} \\ (p_h^{n+1}, \psi) = (\frac{1.5\phi_h^{n+1} - 2\phi_h^n + 0.5\phi_h^{n-1}}{\Delta t}, \psi) + (\nabla\cdot\vec{a}_h^n, \psi), \\ \qquad\qquad \forall\psi \in X_h^k.\end{cases}$$

Again the right hand side in step 2 is given by

$$-\int_\Omega \nabla \cdot \vec{a}_h^{n+1} \psi d\Omega = \int_\Omega \nabla \psi \cdot \vec{a}_h^{n+1} d\Omega$$

We make the following remarks for both method 2.4 and method 2.5.

Remark 1. In both Method 4 and 5. the integral $(\vec{f}, \psi)$ is approximated by $(\Pi_h \vec{f}, \psi)$. By Sobolev interpolation theory [7] , this approximation does not decrease the order of accuracy.

Remark 2. The reason for that the order of the elements for ϕ is 1 higher than the other terms is that the boundary condition for $\vec{a}$ is given by the derivative of ϕ in step 1.

Remark 3. The gradient of ϕ computed in step 3 can be used to compute the tangent derivative of ϕ along the boundary.

Remark 4. The step 2 is solving a Poisson-Neumann problem. It's solution is unique up to a constant. In gauge method we need only one of this kind of solutions. And we will prove that the conjugate gradient method converges to a solution for any initial gauss, see chapter 4.

2.3 Is There Any Higher-Order Time Stepping Procedure?

Since we are going to use high order finite/spectral element method for space discretization, it would be ideal if we can have high order time stepping procedures to match them. The stability issue is crucial. The stability regions of two important classes of multistep methods–mixed Adams-Bashforth-Adams-Molton(ABAM) methods and backward differentiation methods– for ordinary differential equations are analyzed in this section. We have tried the third-order backward differentiation gauge finite element method and the forth-order Runge-Kutta gauge finite element method. Our preliminary computational results show that they are not stable. It is an open problem to find a high order time stepping procedure for gauge finite element methods.

2.3.1 Stability Regions of the Mixed ABAM Methods

Consider an ordinary differential equation as follows

$$\frac{\partial u}{\partial t} = f + g \tag{2.10}$$

Integrate the equation with respect time from t^n to t^{n+1}. we get

$$u_{n+1} - u_n = \int_{t_n}^{t_{n+1}} f dt + \int_{t_n}^{t_{n+1}} g dt \tag{2.11}$$

The mixed ABAM method is obtained by using the explicit Adams-Bashforth method [12] to treat the first integral and the implicit Adams-Molton method [12] to treat the second integral. The kth-order ABAM method is as following:

$$u_{n+1} = u_n + \Delta t \Sigma_{j=1}^{k} \beta_{kj} f_{n+1-j} + \Delta t \Sigma_{j=1}^{k} \beta^*_{kj} g_{n+2-j} \tag{2.12}$$

The coefficients β_{kj}, β^*_{kj} are given in table 2.1.

Table 2.1: Coefficients for ABAM methods

i	1	2	3	4	5
β_{1i}	1				
$2\beta_{2i}$	3	-1			
$12\beta_{3i}$	23	-16	5		
$24\beta_{4i}$	55	-59	37	-9	
$720\beta_{5i}$	1901	-2774	2616	-1274	251
β^*_{1i}	1				
$2\beta^*_{2i}$	1	1			
$12\beta^*_{3i}$	5	8	-1		
$24\beta^*_{4i}$	9	19	-5	1	
$720\beta^*_{5i}$	251	646	-264	106	-19

To find the stability regions of the mixed ABAM methods. consider the following special ordinary differential equation

$$\frac{\partial u}{\partial t} = \lambda u \tag{2.13}$$

where $\lambda = \bar{b}+i\bar{c}$, $\bar{b} \leq 0$, $\bar{c}$ is also a real number, and i is the imaginary unit. Apply the mixed ABAM method to this equation, treating the imaginary term explicitly and real term implicitly. Then we have

$$u_{n+1} = u_n + ic\Sigma_{j=1}^{k}\beta_{kj}u_{n+1-j} + b\Sigma_{j=1}^{k}\beta_{kj}^{*}u_{n+2-j} \tag{2.14}$$

where $b = \Delta t\bar{b}, c = \Delta t\bar{c}$.

The stability region of a method is defined as the set of all the $\lambda\Delta t$ which makes the method stable. We use the Fourier mode analysis to find the stability region. Let $u_n = e^{in\theta}, 0 \leq \theta \leq 2\pi$, then

$$e^{ik\theta} = e^{i(k-1)\theta} + ic\Sigma_{j=1}^{k}\beta_{k,j}e^{i(k-j)\theta} + \Sigma_{j=1}^{k}\beta_{kj}^{*}e^{i(k+1-j)\theta} \tag{2.15}$$

Solve for b, c, and plot (b, c), we get the following pictures of the stability regions of the mixed ABAM methods.

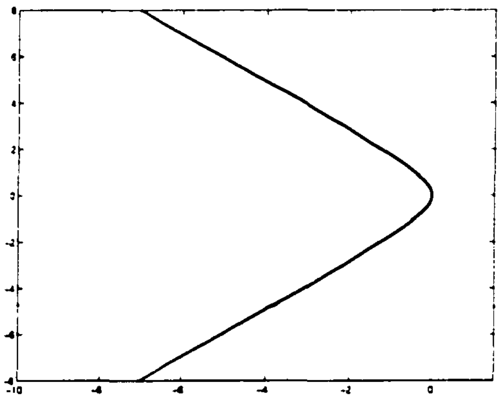

Figure 2.1: The stability region of the ABAM method of first-order. method is stable in the left side of the curve

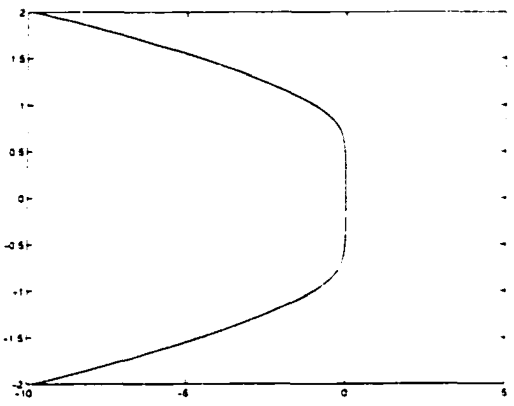

Figure 2.2: The stability region of the ABAM method of second-order. method is stable in the left side of the curve

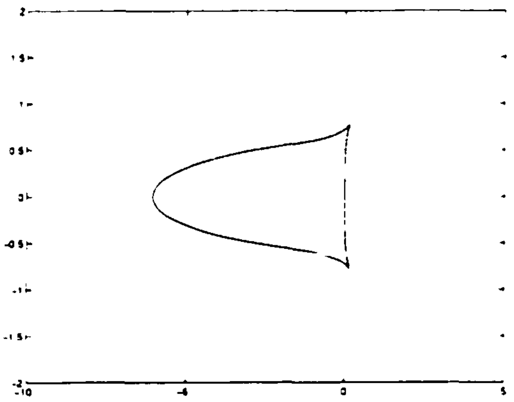

Figure 2.3: The stability region of the ABAM method of third-order. method is stable inside the curve

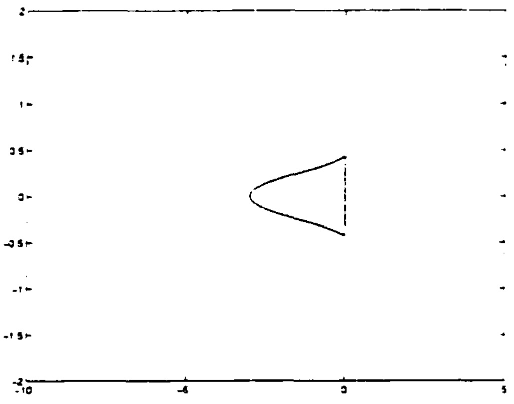

Figure 2.4: The stability region of the ABAM method of forth-order. method is stable inside the curve

2.3.2 Stability Regions of Backward Differentiation Methods

The k-th order backward differentiation method [19] for solving (2.10) is defined as follows

$$\frac{\sum_{j=0}^{k} \beta_{kj} u_{n+1-j}}{\Delta t} = \sum_{j=0}^{k-1} \beta^*_{kj} f_{n-j} + g_{n+1} \tag{2.17}$$

where the coefficients $\beta_{k,j}$, β^*_{kj} for $k = 1, \ldots, 5$, are given in table 2.2.

Table 2.2: Coefficients for Backward-Differentiation methods

i	0	1	2	3	4	5
β_{1i}	1					
β_{2i}	3/2	-2	1/2			
β_{3i}	11/6	-3	3/2	-1/3		
β_{4i}	25/12	-4	3	-4/3	1/4	
β_{5i}	137/66	-5	5	-10/3	5/4	-1/5
β^*_{1i}	1					
β^*_{2i}	2	-1				
β^*_{3i}	3	-3	1			
β^*_{4i}	4	-6	4	-1		
β^*_{5i}	5	-10	10	-5	1	

Similarly to the ABAM methods, we obtain the following figures for the stability regions of the backward-differentiation methods.

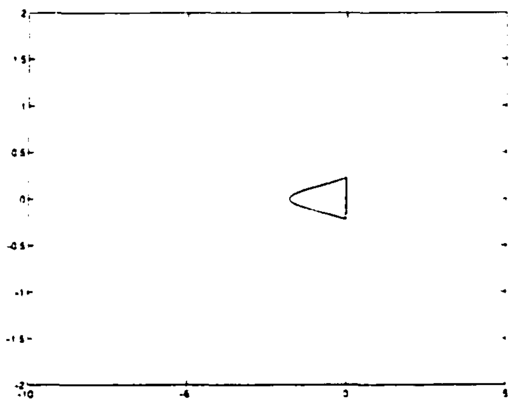

Figure 2.5: The stability region of the ABAM method of fifth-order. method is stable inside the curve

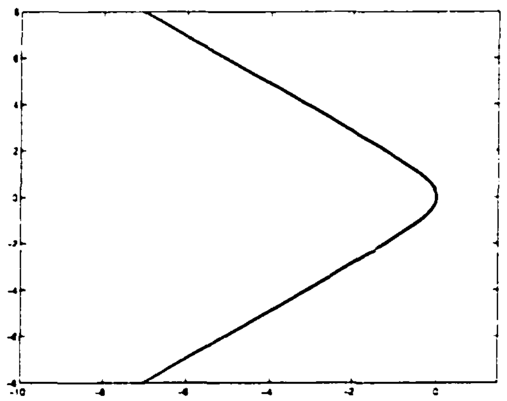

Figure 2.6: The stability region of the backward-differentiation method of first-order. method is stable in the left side of the curve

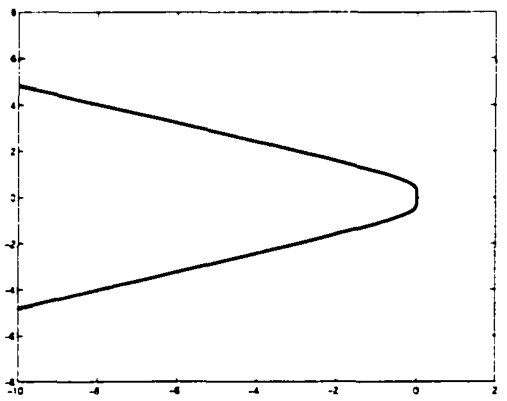

Figure 2.7: The stability region of the backward-differentiation method of second-order. method is stable in the left side of the curve

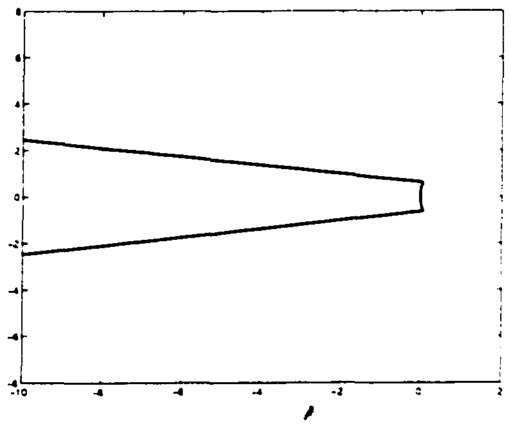

Figure 2.8: The stability region of the backward-differentiation method of third-order, method is stable in the left side of the curve

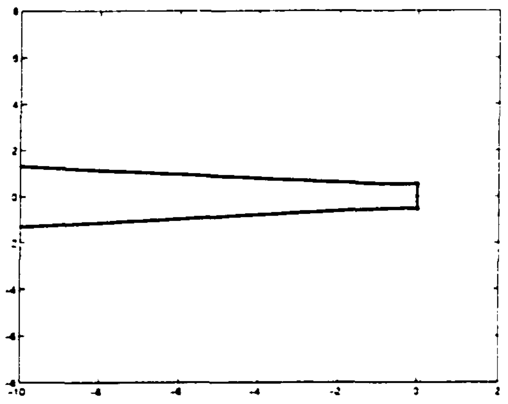

Figure 2.9: The stability region of the backward-differentiation method of forth-order, method is stable in the left side of the curve

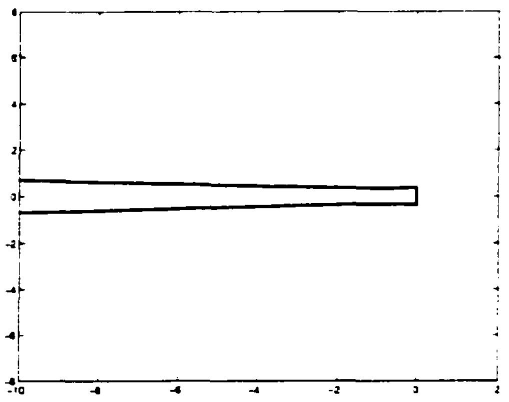

Figure 2.10: The stability region of the backward-differentiation method of fifth-order, method is stable in the left side of the curve

2.3.3 Is There Any Higher Order Time-Stepping Procedure?

We may use the above higher-order methods for temporal discretization and finite element methods for space discretization. For example we may have

Method 2.6. *Third-order backward differentiation gauge method with Neumann boundary condition*

$$\text{Step 1.}\begin{cases} (\frac{11}{6}\vec{a}^{n+1} - 3\vec{a}^n + \frac{3}{2}\vec{a}^{n-1} - \frac{1}{3}\vec{a}^{n-2})/\Delta t + (3(\vec{u}^n\cdot\nabla)\vec{u}^n - 3(\vec{u}^{n-1}\cdot\nabla)\vec{u}^{n-1} \\ +(\vec{u}^{n-2}\cdot\nabla)\vec{u}^{n-2}) = \Delta\vec{a}^{n+1}/Re + \vec{f}^{n+1} \quad \text{in} \quad \Omega \\ \vec{a}^{n+1}\cdot\vec{n} = 0 \quad \text{on} \quad \Gamma \\ \vec{a}^{n+1}\cdot\vec{\tau} = -(3\frac{\partial\phi^n}{\partial\vec{\tau}} - 3\frac{\partial\phi^{n-1}}{\partial\vec{\tau}} + \frac{\partial\phi^{n-2}}{\partial\vec{\tau}}) \end{cases}$$

$$\text{Step 2.}\begin{cases} -\Delta\phi^{n+1} = \nabla\cdot\vec{a}^{n+1}, \quad \text{in} \quad \Omega \\ \frac{\partial\phi^{n+1}}{\partial\vec{n}} = 0, \quad \text{on} \quad \Gamma \end{cases}$$

$$\text{Step 3.} \quad \vec{u}^{n+1} = \vec{a}^{n+1} + \nabla\phi^{n+1}.$$

Unfortunately our preliminary computational results showed that Method 6 is not stable when a higher-order finite element method is used for space discretization.

Using the forth-order explicit Runge-Kutta method for temporal discretization leads to:

Method 2.7: *Forth-order Runge-Kutta gauge method with Neumann boundary condition*

$$\text{Stage 1:}\begin{cases} \frac{\vec{a}^{(1)}-\vec{a}^n}{\Delta t/2} + (\vec{u}^n\cdot\nabla)\vec{u}^n = \frac{1}{Re}\Delta\vec{a}^n \\ \qquad \vec{a}^{(1)}\cdot\vec{n} = 0, \vec{a}^{(1)}\cdot\vec{\tau} = -\frac{\partial\phi^n}{\partial\vec{\tau}} \\ \Delta\phi^{(1)} = -\nabla\cdot\vec{a}^{(1)}, \ \frac{\partial\phi^{(1)}}{\partial\vec{n}} = 0 \\ \vec{u}^{(1)} = \vec{a}^{(1)} + \nabla\phi^{(1)} \end{cases}$$

$$\text{Stage 2:}\begin{cases} \frac{\vec{a}^{(2)}-\vec{a}^n}{\Delta t/2} + (\vec{u}^{(1)}\cdot\nabla)\vec{u}^{(1)} = \frac{1}{Re}\Delta a^{(1)} \\ \qquad \vec{a}^{(2)}\cdot\vec{n} = 0, \vec{a}^{(2)}\cdot\vec{\tau} = -\frac{\partial\phi^{(1)}}{\partial\vec{\tau}} \\ \Delta\phi^{(2)} = -\nabla\cdot\vec{a}^{(2)}, \ \frac{\partial\phi^{(2)}}{\partial\vec{n}} = 0 \\ \vec{u}^{(2)} = \vec{a}^{(2)} + \nabla\phi^{(2)} \end{cases}$$

$$\text{Stage 3: } \begin{cases} \frac{\vec{a}^{(3)}-\vec{a}^{n}}{\Delta t} + (\vec{u}^{(2)}\cdot\nabla)\vec{u}^{(2)} = \frac{1}{Re}\Delta a^{(2)} \\ \vec{a}^{(3)}\cdot\vec{n} = 0,\ \vec{a}^{(3)}\cdot\vec{\tau} = -\frac{\partial\phi^{(2)}}{\partial\vec{\tau}} \\ \Delta\phi^{(3)} = -\nabla\cdot\vec{a}^{(3)},\ \frac{\partial\phi^{(3)}}{\partial\vec{n}} = 0 \\ \vec{u}^{(3)} = \vec{a}^{(3)} + \nabla\phi^{(3)} \end{cases}$$

$$\text{Stage 4: } \begin{cases} \frac{\vec{a}^{(n+1)}-\frac{1}{3}(-\vec{a}^{n}+\vec{a}^{(1)}+2\vec{a}^{(2)}+\vec{a}^{(3)})}{\Delta t/6} + (\vec{u}^{(3)}\cdot\nabla)\vec{u}^{(3)} = \frac{1}{Re}\Delta a^{(3)} \\ \vec{a}^{(n+1)}\cdot\vec{n} = 0,\ \vec{a}^{(n+1)}\cdot\vec{\tau} = -\frac{\partial\phi^{(3)}}{\partial\vec{\tau}} \\ \Delta\phi^{(n+1)} = -\nabla\cdot\vec{a}^{(n+1)},\ \frac{\partial\phi^{(n+1)}}{\partial\vec{n}} = 0 \\ \vec{u}^{(n+1)} = \vec{a}^{(n+1)} + \nabla\phi^{(n+1)} \end{cases}$$

Unfortunately, our preliminary computational results show that this method when coupled with high order finite element method for space discretization is not stable either.

CHAPTER 3

SIMPLE FINITE/SPECTRAL ELEMENT METHODS

The Navier-Stokes equations for two dimensional incompressible flow in vorticity-stream function formulation are as follows

$$(3.1)\qquad \begin{cases} \frac{\partial\omega}{\partial t} + (\vec{u}\cdot\nabla)\omega = \frac{1}{Re}\Delta\omega + f_1 & \text{in } \Omega \\ \Delta\psi = \omega & \text{in } \Omega \end{cases}$$

with the boundary conditions $\psi|_\Gamma = f_2, \frac{\partial\psi}{\partial\vec{n}}|_\Gamma = f_3$, where $\vec{u} = \nabla^\perp\psi \equiv (-\frac{\partial\psi}{\partial y}, \frac{\partial\psi}{\partial x})$, and $\vec{n}$ is the unit outward normal direction.

Note that in (3.1) the vorticity function ω and the stream function ψ are coupled together, much confusion and complexity would come if their computation still coupled together in a numerical method for solving (3.1).

Also note that there is no explicit boundary conditions for the vorticity function, this could cause difficulty when one constructs finite difference methods, especially for curved boundary problems.

Finite element methods for solving (3.1) seem more promising, since the variational formulation of (3.1) has natural enforcement of boundary conditions. There have been some finite element methods for the steady Navier-Stokes equations, i.e. the time independent problem, see [1, 2, 4, 18, 30]. But in all the schemes, the convection term is treated implicitly. so the computation of the vorticity and stream functions are fully coupled.

To avoid this coupling, it is natural to think to treat the convection term explicitly. The issue of stability becomes crucial in choosing time stepping procedure.

Consider the standard advection-diffusion equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}. \tag{3.2}$$

If we use the forward Euler method for time discretization. the second-order centered difference method for space discretization, and both the convection and diffusion terms are treated explicity, then the scheme is stable only under the constraint

$$\Delta t(\frac{a^2}{2\nu} + \frac{2\nu}{\Delta x^2}) > 1. \tag{3.3}$$

Then we have

$$\nu\frac{\Delta t}{\Delta x^2} < \frac{1}{2}, \Delta t < \frac{2\nu}{a^2}. \tag{3.4}$$

Since $\nu = O(\frac{1}{Re})$. we have

$$\frac{a\Delta t}{\Delta x} < O(\frac{1}{\sqrt{Re}}) \tag{3.5}$$

(3.5) is a severe constraint, since ideally we want $\frac{a\Delta t}{\Delta x} = O(1)$ for $Re \gg 1$. This stability constraint still remains even if we discretize the diffusion term implicitly, keeping the advection term explicit, since at high Reynolds number the diffusion term is of very little help. Although such constraint disappears if the advection term is also treated implicitly, it is too expensive.

Recently Liu and E discovered an efficient time step procedure, which allows the convection term and diffusion term are treated explicitly and the stability is obtained under standard Courant-Fridrich-Lewy(CFL) condition. The instability of the forward Euler method is due to the fact that the stability region of the forward Euler method does not contain any part of imaginary axis. The same is true for the second-order explicit Runge-Kutta methods, but not for third- and fourth-order ones. The Fourier symbol for the centered difference operator $-a\tilde{D}_x^2 + \nu D_x^2$ is $C(\xi) = ia\frac{sin\xi}{\Delta x} - \frac{4\nu}{\Delta x^2}sin^2(\frac{\xi}{2})$. Therefore if we use forth-order Runge-Kutta in time, the stability conditions are

$$(3.6) \qquad \frac{a\Delta t}{\Delta x} \leq C_1, \frac{4\nu\Delta t}{\Delta x^2} \leq C_2.$$

We see for large Reynolds number problem, the size of Δt is controlled only by the CFL condition.

This time stepping procedure has been proved successful for the case of finite difference methods [8, 9]. We know that the linear finite element method is equivalent to second order centered difference scheme. We can imagine that higher order finite element discretization is equivalent to some kind of centered difference scheme.

Recently, E and Liu proposed a simple finite element method, in which the nonlinear convection term and the advection term are treated explicitly. The key idea is to use higher-order Runge-Kutta methods for the time discretization.

Theoretically they proved the stability and convergence for the semi-discrete (i.e. keeping time continuous) scheme. The error estimates, however, may be very crude. There is no proof for the convergence of the fully discrete scheme.

We will report the results of some numerical experiments of the simple finite element method, checking the order of accuracy for the convergence by using artificial exact solution of the Navier-Stokes equations and simulation of the canonical driven cavity flows.

We will first briefly review the simple finite element scheme, and the stability and convergence theorems for semi-discrete case. Then the results of numerical experiments will be reported.

3.1 Semi-Discrete Method

Since $div(\phi\omega\vec{u}) = (\omega\vec{u}\cdot\nabla)\phi + \phi div(\omega\vec{u})$, and also by the incompressibility condition $div(\vec{u}) = 0$, we have $div(\omega\vec{u}) = (\vec{u}\cdot\nabla)\omega + \omega div(\vec{u}) = (\vec{u}\cdot\nabla)\omega$.

Denote by $< \cdot,\cdot >$ the inner product in the Sobolev space H_0^1, H^1, then

$$< \phi,(\vec{u}\cdot\nabla)\omega > = < \phi, div(\omega\vec{u}) > = - < \nabla\phi, \omega\vec{u} >, \forall\phi\in H_0^1. \tag{3.7}$$

By using divergence theorem, we have

Method 3.1 *Variational formulation of vorticity-stream function equations*

$$\begin{cases} \text{Find } \omega\in H^1, \psi\in H^1 \text{ such that} \\ < \phi, \frac{\partial\omega}{\partial t} > - < \nabla\phi, \omega\vec{u} > = -\frac{1}{Re} < \nabla\phi, \nabla\omega > + < \phi, f_1 >, & \forall\phi\in H_0^1(\Omega) \\ < \nabla\phi, \nabla\psi > = - < \phi, \omega > + \int_\Gamma \phi f_3 ds, & \forall\phi\in H^1(\Omega) \end{cases}$$

Since $\omega = \Delta\phi$, Method 1 can be written as the following more symmetric form

Method 3.2 *Symmetric variational formulation of vorticity-stream function equations*

$$\begin{cases} \text{Find } \omega\in H^1, \psi\in H^1 \text{ such that} \\ < \nabla\phi, \nabla\frac{\partial\psi}{\partial t} > + < \nabla\phi, \omega\vec{u} > = \frac{1}{Re} < \nabla\phi, \nabla\omega > - < \phi, f_1 >, & \forall\phi\in H_0^1(\Omega) \\ < \nabla\phi, \nabla\psi > = - < \phi, \omega > + \int_\Gamma \phi f_3 ds, & \forall\phi\in H^1(\Omega) \end{cases}$$

We now give the simple finite element methods based on method 2. Let X_h^k be the standard continuous finite element space with kth degree polynomials on each element of a triangulation T_h, where h is the maximum size of the elements. Denote $X_{0,h}^k$ the subspace of X_h^k with zero boundary values.

Method 3.3 *Semi-discrete finite element method for vorticity-stream function formulation*

$$\begin{cases} \text{Find } \omega_h\in X_h^k, \text{ and} \psi_h\in X_h^k \text{ such that} \\ < \nabla\phi, \nabla\frac{\partial\psi_h}{\partial t} > + < \nabla\phi, \omega\vec{u}_h > = \frac{1}{Re} < \nabla\phi, \nabla\omega_h > - < \phi, f_1 >, & \forall\phi\in X_{0,h}^k \\ < \nabla\phi, \nabla\psi_h > = - < \phi, \omega_h > + \int_\Gamma \phi f_3 ds, & \forall\phi\in X_h^k \end{cases}$$

and the velocity field can be obtained from the stream function via

$$\vec{u}_h = \nabla^{\perp}\phi_h. \tag{3.8}$$

The following stability and convergence theorems for method 3 were proved by Liu and E [22]

Theorem 3.1 (Liu & E) . *Let $\vec{u}_h$ and ω_h be the approximate solution given by Method 3.3. Then*

$$\|\vec{u}_h(\cdot,t)\|_{L_2}^2 + \frac{2}{Re}\int_0^t \|\omega_h(\cdot,s)\|_{L_2}^2 ds = \|\vec{u}_h(\cdot,0)\|_{L_2}^2 \tag{3.9}$$

Theorem 3.2 (Liu & E) *Let $(\psi,\omega,\vec{u})$ be a solution of the Navier-Stokes equation (3.1) and $(\psi_h,\omega_h,\vec{u}_h)$ be the approximate solution given by Method 3.3, then we have*

$$\begin{aligned}&\|\vec{u}-\vec{u}_h\|_{L^\infty((0,T];L^2)} + \|\omega-\omega_h\|_{L^\infty((0,T];L^2)}\\ &\le Ch^{k-1/2}(\|\psi\|_{L^\infty((0,T];W^{k+1,\infty})} + \|\omega\|_{L^\infty((0,T];H^{k+1})}e^{(CT\,Re(\|\omega\|_\infty^2,+\|\vec{u}\|_\infty^2)}.\end{aligned}$$

where C is a constant which does not depend on h or the numerical solution.

Remark. The error bound grows exponentially as a function of Reynolds number Re and time T. This may be very crude, as we will see some accuracy checking later.

3.2 Full Discrete Scheme

To avoid the coupling of the computation of the vorticity and stream functions, the convection term is treated explicitly. As in the case of the finite difference methods , we can expect that high order explicit Runge-Kutta method for time discretization will lead to a stable method, even with large Reynolds number .

Now we describe the full simple finite element method. Given the nth time step solution ψ_h^n, ω_h^n, note that $\vec{u}_h = \nabla^{\perp}\psi_h$, the we can produce $\psi_h^{n+1}, \omega_h^{n+1}$ in four steps:

Method 3.4 *Simple Finite Element Method:*

$$\text{Step 1.}\begin{cases} \text{Find } \psi_h^{(1)}, \omega_h^{(1)} \text{ such that} \\ < \nabla\phi, \nabla\psi_h^{(1)} > = < \nabla\phi, \nabla\psi_h^n > - \\ \frac{\Delta t}{2} < \nabla\phi, \omega_h^n \nabla^\perp \psi_h^n > - \frac{\Delta t}{2Re} < \nabla\phi, \nabla\omega_h^n > - \frac{\Delta t}{2} < \phi, f_1^n >. \\ \qquad\qquad \forall\phi \in X_{0,h}^k \\ < \phi, \omega_h^{(1)} > = - < \nabla\phi, \nabla\psi_h^{(1)} > + \int_\Gamma \phi f_3 ds. \\ \qquad\qquad \forall\phi \in X_h^k \end{cases}$$

$$\text{Step 2.}\begin{cases} \text{Find } \psi_h^{(2)}, \omega_h^{(2)} \text{ such that} \\ < \nabla\phi, \nabla\psi_h^{(2)} > = < \nabla\phi, \nabla\psi_h^n > - \\ \frac{\Delta t}{2} < \nabla\phi, \omega_h^{(1)} \nabla^\perp \psi_h^{(1)} > + \frac{\Delta t}{2Re} < \nabla\phi, \nabla\omega_h^{(1)} > - \frac{\Delta t}{2} < \phi, f_1^{n+\frac{1}{2}} >. \\ \qquad\qquad \forall\phi \in X_{0,h}^k \\ < \phi, \omega_h^{(2)} > = - < \nabla\phi, \nabla\psi_h^{(2)} > + \int_\Gamma \phi f_3 ds. \\ \qquad\qquad \forall\phi \in X_h^k \end{cases}$$

$$\text{Step 3.}\begin{cases} \text{Find } \psi_h^{(3)}, \omega_h^{(3)} \text{ such that} \\ < \nabla\phi, \nabla\psi_h^{(3)} > = < \nabla\phi, \nabla\psi_h^n > - \\ \Delta t < \nabla\phi, \omega_h^{(2)} \nabla^\perp \psi_h^{(2)} > + \frac{\Delta t}{Re} < \nabla\phi, \nabla\omega_h^{(2)} > - \Delta t < \phi, f_1^{n+\frac{1}{2}} >. \\ \qquad\qquad \forall\phi \in X_{0,h}^k \\ < \phi, \omega_h^{(3)} > = - < \nabla\phi, \nabla\psi_h^{(3)} > + \int_\Gamma \phi f_3 ds. \\ \qquad\qquad \forall\phi \in X_h^k \end{cases}$$

$$\text{Step 4.}\begin{cases} \text{Find } \psi_h^{n+1}, \omega_h^{n+1} \text{ such that} \\ < \nabla\phi, \nabla\psi_h^{n+1} > = < \nabla\phi, \frac{1}{3}\nabla(-\psi_h^n + \psi_h^{(1)} + 2\psi_h^{(2)} + \psi_h^{(3)}) > - \\ \frac{\Delta t}{6} < \nabla\phi, \omega_h^{(3)} \nabla^\perp \psi_h^{(3)} > + \frac{\Delta t}{6Re} < \nabla\phi, \nabla\omega_h^{(3)} > - \frac{\Delta t}{6} < \phi, f_1^{n+1} >. \quad \forall\phi \in X_{0,h}^k \\ < \phi, \omega_h^{n+1} > = - < \nabla\phi, \nabla\psi_h^{n+1} > + \int_\Gamma \phi f_3 ds. \\ \qquad\qquad \forall\phi \in X_h^k \end{cases}$$

There are two substeps in each step. The first substep involves solving a linear system of which the coefficient matrix is the standard stiffness matrix.

while the second step is to solve a linear system of equations of which the coefficient matrix is the standard mass matrix. These coefficient matrices are large sparse symmetric positive definite matrices.

we use the preconditioned conjugate gradient methods to solve the linear systems. We choose the trigonometric transforms [31, 34] as the preconditioners for the stiffness matrix, while classic Jacobi method for the mass matrix. The details about the assembly of the matrix-vector product, assembly for the nonlinear term are given in chapter 5.

CHAPTER 4

FAST POISSON SOLVERS

The conjugate gradient(CG) method was developed independently and in different forms by Lanczos [21] and Hestness and Stiefel [17] in fifties. The method was essentially viewed as a direct solution technique and was abandoned early on because it did not compare well with other existing techniques. For example, in inexact arithmetic, the method does not terminate in n steps as is predicted by the theory. This is caused by the severe loss of the orthogonality of vector quantities generated by the algorithm. The paper of Reid [27] in 1971 drew the attention of many researchers to the potential of the algorithm as a iterative method for large sparse linear systems. It was a catalyst for much of the subsequent work in conjugate gradients.

Lack of robustness is a widely recognized weakness of iterative solvers, relative to direct solvers. Both of the efficiency and robustness of iterative techniques can be improved by using preconditioning techniques. Preconditioning is simply a means of transforming the original linear system into one which has the same solution, but which is likely to be easier to solve with an iterative solver.

Now the preconditioned conjugate gradient methods have become the most important iterative methods for solving large sparse linear systems of equations, see e.g. [14, 15, 29]. We will use the discrete trigonometric transforms as preconditioners for solving discrete Poisson equations, and classic Jacobi method for solving discrete heat-like equations.

4.1 The Conjugate Gradient (CG) Method

The convergence of the CG method for symmetric, positive definite linear systems are proved in many textbooks, see e.g. [14, 15, 29]. Since we are going to apply the CG method to symmetric positive semi-definite linear systems, we will show that it still converges. And the convergence rate estimate is similar to the case of symmetric positive definite system.

Consider the following linear system of equations

$$Ax = b \tag{4.1}$$

where $A \in \mathbf{R}^{n\times n}$ is symmetric, positive semi-definite, $b \in \mathbf{R}^n$.

Given an initial gauss x_0, we have the following conjugate gradient (CG) method:

```
Compute  $r_0 := b - Ax_0, p_0 = r_0$.
For  $j = 0, 1, ...$until convergence Do :
  $\alpha_j := (r_j, r_j)/(Ap_j, p_j)$
  $x_{j+1} := x_j + \alpha_j p_j$
  $r_{j+1} := r_j - \alpha_j Ap_j$
  $\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$
  $p_{j+1} := r_{j+1} + \beta_j p_j$
Enddo
```

Now we show the convergence of the CG method for symmetric positive semi-definite system. Suppose the eigenvalues of A are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0, \lambda_{r+1} = \cdots = \lambda_n = 0$, the corresponding orthonormal eigenvectors are $v_1, \cdots, v_r, v_{r+1}, \cdots, v_n$. Then the null space $N(A)$ of A is $span\{v_{r+1}, \cdots, v_n\}$, and the range space $R(A)$ is $span\{v_1, \cdots, v_r\}$.

Lemma 4.1 The sequences r_k, p_k are in $R(A)$.

From lemma 4.1, we have

Lemma 4.2 $p_k = 0$ is equivalent to $Ap_k = 0$ or $(Ap_k, p_k) = 0$.

Note that CG method could breakdown if either some $r_k = 0$ or $p_k = 0$. If $r_k = 0$, then x_k is a solution. If $p_k = 0$, we will show that $r_k = 0$. Thus if CG method break down then we end up with a solution.

Lemma 4.3 Let the sequences $r_j, p_j, j = 0, \cdots, k$ are generated by the CG method, then

(4.2) $span\{r_0, \cdots, r_k\} = span\{p_0, \cdots, p_k\} = span\{r_0, Ar_0, \cdots, A^k r_0\}$.

Proof. First we point out that $r_j, p_j \neq 0, j = 0, \cdots k-1$. Let us prove (4.2) by induction method. Assume $span\{r_0, \cdots, r_m\} = span\{p_0, \cdots, p_m\} = span\{r_0, Ar_0, \cdots, A^m r_0\}$, for some $m < k$. Then $p_{m+1} = r_{m+1} + \beta_m p_m \in span\{r_0, \cdots, r_{m+1}\}$, and $r_{m+1} = r_m - \alpha_m A p_m \in span\{r_0, Ar_0, \cdots, A_{m+1} r_0\}$. Thus, $span\{p_0, \cdots, p_{m+1}\} \subset span\{r_0, \cdots, r_{m+1}\} \subset span\{r_0, Ar_0, \cdots, A^{m+1} r_0\}$. Also, $r_{m+1} = p_{m+1} - \beta_m p_m \in span\{p_0, \cdots, p_{m+1}\}$, thus$span\{r_0, \cdots, r_{m+1}\} = span\{p_0, \cdots, p_{m+1}\}$. By the induction hypothesis $A^m r_0 \in span\{p_0, \cdots, p_m\}$, we have $A^{m+1} r_0 \in span\{Ap_0, \cdots, Ap_m\}$. Since $Ap_j = (r_{j+1} - r_j)/\alpha_j$, we have $Ap_j \in span\{r_0, \cdots, r_{j+1}\}$, for $j = 0, \cdots m$. Thus $Ar_{m+1} \in span\{r_0, \cdots, r_{m+1}\}$, the lemma is proved. ◇

Lemma 4.4 Let the sequences $\{r_j\}_{j=0}^k, \{p_j\}_{j=0}^k$ are generated by the CG method, then

(4.3) $(p_i, Ap_j) = 0, i \neq j, \ i, j \leq k$.

(4.4) $(r_i, r_j) = 0, i \neq j, \ i, j \leq k$.

Proof. We use induction method.

$(r_{j+1}, p_j) = (r_j - \alpha_j A p_j, p_j) = (r_j, p_j) - (r_j, r_j)$, for $j = 0, \cdots, k$. Since $p_0 = r_0$, thus $(r_1, r_0) = (r_1, p_0) = 0$.

$(p_1, Ap_0) = (r_1 + \beta_0 p_0, Ap_0) = (r_1, Ap_0) + (r_1, r_1)(p_0, Ap_0)/(r_0, r_0) = (r_1, -(r_1 - r_0)/\alpha_0) + (r_1, r_1)(p_0, Ap_0)/(r_0, r_0) = 0$.

Suppose that $(p_i, Ap_j) = 0, (r_i, r_j) = 0, i \neq j, \ i, j \leq m$. Then $(r_{m+1}, r_m) = (r_m, r_m) - (r_m, r_m)(Ap_m, r_m)/(Ap_m, p_m)$. But $(Ap_m, r_m) = (Ap_m, p_m - \beta_{m-1} p_{m-1}) = (Ap_m, p_m)$, since $(Ap_m, p_{m-1}) = 0$. Thus $(r_{m+1}, r_m) = 0$.

Note $\beta_m = \frac{(r_{m+1}, r_{m+1})}{(r_m, r_m)} = \frac{(r_{m+1}, r_{m+1} - r_m)}{(r_m, r_m)} = \frac{(r_{m+1}, -\alpha_m A p_m)}{(r_m, r_m)} = -\frac{(r_{m+1}, Ap_m)}{(Ap_m, p_m)}$,

thus

$(p_{m+1}, Ap_m) = (r_{m+1}+\beta_m p_m, Ap_m) = 0$. The left thing is to show $(r_{m+1}, r_j) = 0, (p_{m+1}, Ap_j) = 0$, if $j < m$. $(r_{m+1}, r_j) = (r_m - \alpha_m Ap_m, r_j)$, by induction hypothesis, $(r_m, r_j) = 0$. $(Ap_m, r_j) = (p_m, Ar_j) = 0$, since $r_j \in span\{r_0, \cdots, r_j\} = span\{p_0, \cdots, p_j\}$. Again, when $j < k, (p_{m+1}, Ap_j) = (r_{m+1} + \beta_m p_m, Ap_j) = (r_{m+1}, Ap_j) = 0$, since $p_j \in span\{r_0, \cdots, A^j r_0\}, Ap_j \in span\{r_0, \cdots, A^{j+1} r_0\} = span\{r_0, \cdots, r_{j+1}\}$. $\diamond$

From Lemma 4.3 and Lemma 4.4, we can see that if $p_k \neq 0$, and $p_{k+1} = 0$, then we have $r_{k+1} = 0$, the CG method ends with a solution.

Theorem 4.5 *The CG method converges to a solution after at most $rank(A)$ steps.*

Proof. Since $span\{r_0, \cdots, r_k\} \subset R(A)$, of which the dimension is $rank(A)$, and $r_0, \cdots, r_k$ are orthogonal, $r_{rank(A)} = 0$ if the CG method does not break before this step. $\diamond$

Note that $\mathbf{R}^n = N(A) \oplus R(A)$, then the initial gauss x_0 can be decomposed into the sum of its $N(A)$ part and $R(A)$ part. It is easy to see for CG method, all iterates x_k has the same $N(A)$ part as that of x_0. Let x^* be the solution of the CG method started from initial gauss x_0, then $e_0 = x^* - x_0, x_k - x_0 \in R(A)$. Since A is symmetric positive definite on $R(A)$, we can define an inner product $< \cdot, \cdot >$ on $R(A)$ as $< x, y >= x^T Ay$.

Theorem 4.6 *Suppose that x^* is the solution of the CG method started from initial guess x_0, then $x_k - x_0$ is the orthogonal projection of the initial error $e_0 \equiv x^* - x_0$ with respect to $< \cdot, \cdot >$ on the space $W_k = span\{p_0, \cdots, p_{k-1}\} = span\{r_0, \cdots, A^{k-1} r_0\}$.*

Proof. Since $Ax^* = b$, for $j = 0, ..., k-1$, $< e_0 - (x_k - x_0), p_j >= (r_k, p_j) = 0$, by Lemma 4.4 $\diamond$

Theorem 4.7 *Let $\lambda_1 \geq \cdots \geq \lambda_r$ be all the nonzero eigenvalues of A, then for the CG method,*

$$||x^* - x_k||_A \leq max_{1 \leq j \leq r} |\bar{q}(\lambda_j)| ||x^* - x_0||_A, \forall q_k \in P_k \tag{4.5}$$

where P_k is the set of all polynomials of degree at most k and $q_k(0) = 1$.

Proof: Note $W_k = span\{r_0, ..., A^{k-1} r_0\} = span\{A(x^* - x^0), ..., A^{k-1}(x^* -$

$x^0)\}$. By Theorem 4.6, $x_k - x_0 \in W_k$ is the projection of $x^* - x_0$ into W_k, we have

$$\begin{aligned}&\|x_* - x_k\|_A = \|(x^* - x_0) - (x_k - x_0)\|_A\\ &= \min_{q_k \in P_k, q_k(0)=1} \|q_k(A)(x_* - x_0)\|_A\\ &\le \min_{q_k \in P_k, q_k(0)=1} \max_{\lambda_i \neq 0} |q_k(\lambda_i)| \cdot \|x^* - x_0\|_A \quad \diamond\end{aligned}$$

From the theorem we have

$$\|x^* - x_k\|_A \le \min_{q_k \in P_k, q_k(0)=1} \max_{\lambda \in [\lambda_1, \lambda_r]} |q_k(\lambda)| \cdot \|x^* - x_0\|_A.$$

By Chebyshev best polynomial theorem, we have

$$\|x^* - x_k\|_A \le 2\left(\frac{\sqrt{\mu} - 1}{\sqrt{\mu} + 1}\right)^k \|x^* - x_0\|_A. \tag{4.6}$$

where $\mu = \lambda_1/\lambda_r$.

4.2 Preconditioned Conjugate Gradient Methods

Suppose that A is symmetric and positive semi-definite. A preconditioner M is a matrix which approximate A in some yet-undefined sense. It is assumed that M is symmetric positive definite. This is because that $M^{-1}A$ is self-adjoint for the M-inner product,

$$(x, y)_M \equiv (Mx, y) = (x, My) \tag{4.7}$$

since

$$(M^{-1}Ax, y) = (Ax, y) = (x, Ay) = (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M. \tag{4.8}$$

Therefore, an alterative is to replace the usual Euclidean inner product in the CG method by the M-inner product. So the convergence analysis in previous section can apply the system

$$M^{-1}Ax = M^{-1}b \tag{4.9}$$

If the CG method is rewritten for this new product, denoting by $r_j = b - Ax_j$ the original residual and by $z_j = M^{-1}r_j$ the residual for the preconditioned system, the following sequence of operations is obtained, ignoring the initial step:

1. $\alpha_j := (z_j, z_j)_M / (M^{-1}Ap_j, p_j)_M$
2. $x_{j+1} := x_j + \alpha_j p_j$
3. $r_{j+1} := r_j - \alpha_j Ap_j i$
4. $z_{j+1} := M^{-1}r_{j+1}$
4. $\beta_j := (z_{j+1}, z_{j+1})_M / (z_j, z_j)_M$
5. $p_{j+1} := z_{j+1} + \beta_j p_j$

Since $(z_j, z_j)_M = (r_j, z_j)$ and $(M^{-1}Ap_j, p_j)_M = (Ap_j, p_j)$.the M-inner products do not have to be computed explicitly. With this observation, we obtain the following preconditioned conjugate gradient algorithm:

```
Compute $r_0 := b - Ax_0, z_0 = M^{-1}r_0, p_0 = r_0$.
For $j = 0, 1, ...$until convergence Do :
  $\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
  $x_{j+1} := x_j + \alpha_j p_j$
  $r_{j+1} := r_j - \alpha_j Ap_j$
  $z_{j+1} := M^{-1}r_{j+1}$
  $\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
  $p_{j+1} := z_{j+1} + \beta_j p_j$
Enddo
```

From a practical point of view, the only requirement for M is that it is inexpensive to solve linear systems $Mx = b$. This is because the PCG method will require a linear system solution with the matrix M at each step. In the next section we will study the a class of Poisson solvers- discrete trigonometrical transforms-as the preconditioners.

4.3 Discrete Trigonometric Transforms as Preconditioners

We will show in this section that the discrete trigonometric transforms are exact direct solvers of the linear systems of Poisson equations on rectangles when discretized with standard second-order difference scheme. It is well known that for the Poisson equation on a rectangle, the second-order centered difference scheme is equivalent to the standard linear finite element method. Since we will use high order finite element methods, so to use the discrete trigonometric transforms as preconditioners mean that to use first-order finite element methods as preconditioners for high order finite element method.

Definition 4.1 The Discrete Sine Transform(DST): *Given real $x(1:m-1)$, compute $y(1:m-1)$ such that*

$$y_k = \Sigma_{j=1}^{m-1} sin(\frac{kj\pi}{m})x_j \tag{4.10}$$

Definition 4.2 The Discrete Cosine Transform(DCT): *Given real $x(0:m)$, compute $y(0:m)$ such that*

$$y_k = \frac{x_0}{2} + \Sigma_{j=1}^{m-1} cos(\frac{kj\pi}{m})x_j + \frac{(-1)^k x_m}{2} \tag{4.11}$$

If m is a power of 2 then the FFT-based algorithms for the above transforms only require $2.5mlog_2m$ arithmetic operations, see e.g. [34].

Now let us first look at one-dimensional Poisson problem

$$\frac{d^2u}{dx^2} = f(x), a \leq x \leq b \tag{4.12}$$

for each of the following specific boundary conditions

$$\text{Dirichlet} : u(a) = \alpha, u(b) = \beta. \tag{4.13}$$

$$\text{Neumann} : u'(a) = \alpha, u'(b) = \beta. \tag{4.14}$$

Define a tridiagonal $m \times m$ matrix as follows

$$T_m = \begin{bmatrix} -2 & 1 & \cdots & \cdots & 0 \\ 1 & -2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{bmatrix}. \tag{4.15}$$

Let θ be any real number and for any integer k set $c_k = cos(k\theta)$ and $s_k = sin(k\theta)$. Then we have

$$T_m \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{m-1} \\ s_m \end{bmatrix} = -4sin^2(\theta/2) \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{m-1} \\ s_m \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ s_{m+1} \end{bmatrix}. \tag{4.16}$$

and

$$T_m \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-2} \\ c_{m-1} \end{bmatrix} = -4sin^2(\theta/2) \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-2} \\ c_{m-1} \end{bmatrix} - \begin{bmatrix} c_1 \\ 0 \\ \vdots \\ 0 \\ c_m \end{bmatrix} \tag{4.17}$$

For the 1D Poisson problem let us use a uniform mesh.

$$a = x_0 < x_1 < \cdots < x_n = b, h = (b-a)/n.$$

With the Dirichlet boundary condition, using the standard second-order centered difference scheme leads to the linear system

$$\frac{1}{h^2} T_{n-1} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} f_1 - \alpha/h^2 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} - \beta/h^2 \end{bmatrix}. \tag{4.18}$$

Theorem 4.8 *Let $(n-1)\times(n-1)$ matrix $S = (sin(kj\pi/n)), k, j = 1, ..., n-1$, and*

$$\lambda_j = sin^2(\frac{j\pi}{2n}) \tag{4.19}$$

for $j = 1, \cdots, n-1$. then

$$\begin{aligned} S^{-1} &= (2/n)S \\ S^{-1}T_{n-1}S &= D \end{aligned} \tag{4.20}$$

where $D = diag(\lambda_1, ..., \lambda_{n-1})$.

So we may solve a linear system of the form $T_{n-1}u = g$ as follows:

$$\begin{aligned} u &\leftarrow Sg \\ u &\leftarrow (2/n)Du \\ u &\leftarrow Su \end{aligned}$$

The two DST's can be realized through FFT.

With the Neumann boundary condition, we introduce two ghost points $x_{-1} = a - h, x_{n+1} = b + h$, and we still use the standard second-order centered difference method to approximate $\frac{d^2u}{dx^2}$ at the grid points, and for the boundary constraint we can approximate $u'(x_0) = \alpha$ and $u'(x_n) = \beta$ with

$$\frac{u_1 - u_{-1}}{2h} = \alpha. \tag{4.22}$$

and

$$\frac{u_{n+1} - u_{n-1}}{2h} = \beta \tag{4.23}$$

respectively.

Then we have the following linear system

$$\frac{1}{h^2}\bar{T}_{n+1}\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} f_0 + 2\alpha/h \\ f_1 \\ \vdots \\ f_{n-1} \\ f_n + 2\beta/h \end{bmatrix} \tag{4.24}$$

where

$$\tilde{T}_{n+1} = T_{n+1} + e_0 e_1^T + e_n e_{n-1}^T$$

and $I_{n+1} = [e_0, e_1, ..., e_{n-1}, e_n]$ is a column partitioning of I_{n+1}.

Theorem 4.9 *Let $(n+1) \times (n+1)$ matrix $C = (cos(kj\pi/n)), 0 \le k, j \le n$ and*

$$\lambda_j = -4sin^2(\frac{j\pi}{2n}) \tag{4.25}$$

for $j = 0, \cdots, n$, then

$$C^{-1}\tilde{T}_{n+1}C = D \tag{4.26}$$

where $D = diag(\lambda_0, ..., \lambda_n)$.

Note that the matrix C does not quite define the discrete cosine transform that we defined in the beginning of this subsection. We can see that the DCT of a vector $x(0:n)$ is given by $\tilde{C}x(0:n)$, where

$$\tilde{C} = C\tilde{D}^{-1}, \quad \tilde{D} = diag(2, I_{n-1}, 2).$$

We then have $\tilde{C}^{-1} = (2/n)\tilde{C}$, and

$$\tilde{D}^{-1}\tilde{C}^{-1}\tilde{T}_{n+1}\tilde{C}\tilde{D} = diag(\lambda_0, ..., \lambda_n) \tag{4.27}$$

Thus we may solving a linear system of the form $\tilde{T}_{n+1}u = g$ as follows:

$$u \leftarrow \tilde{C}g$$
$$u \leftarrow (2/n)D^{-1}u$$
$$u \leftarrow \tilde{C}u$$

The two DCTs can be realized by using FFT [34].

Remark. Note that the first diagonal element of D is 0, this is because the Poisson equation with Neumann boundary condition has solutions unique upto a constant. Therefore, we can define

$$D^{-1} = diag(\mu, 1/\lambda_1, ..., 1/\lambda_n)$$

where μ can be any real number.

Now let us consider two-dimensional Poisson equation with Dirichlet or Neumann boundary condition. Given a function $f(x,y)$ defined on

$$\Omega = [a,b] \times [c,d]$$

and $g(x,y)$ on $\partial\Omega$, we wish to determine $u(x,y)$ such that

(4.28) $$\Delta u = f, \quad \forall (x,y) \in \Omega$$

subject to each of the following specific boundary conditions

$$\text{Dirichlet}: u = g \quad \text{on} \quad \partial\Omega$$
$$\text{Neumann}: \tfrac{\partial u}{\partial \vec{n}} = g \quad \text{on} \quad \partial\Omega$$

Using uniform mesh and the standard second order centered difference scheme will lead to a linear system $Mu = g$, where

(4.29) $$M = (I_n \otimes A) + (B \otimes I_m).$$

A, B have the form T_m, T_n for Dirichlet problem, and $\tilde{T}_m, \tilde{T}_n$ for Neumann problem. Here we use $\otimes$ to represent the Kronecker product.

If $V_A^{-1} A V_A = D_A, V_B^{-1} B V_B = D_B$, it can be shown that

(4.30) $$M^{-1} = (V_b \otimes V_A)[(I_n \otimes D_A) + (D_B \otimes I_m)]^{-1}(V_B^{-1} \otimes V_A^{-1})$$

Denote a $m \times n$ matrix $f_{m\times n}$ the right hand side of the linear system resulting from the discretization of Poisson equation. Then the solution of $Mu_{m\times n} = f_{m\times n}$ can be obtained as follows:

$$u_{m\times n} \leftarrow V_A^{-1} f_{m\times n} V_B^{-1}$$
$$u_{m\times n} \leftarrow D_A^{-1} u_{m\times n} D_B^{-1}$$
$$u_{m\times n} \leftarrow V_A u_{m\times n} V_B$$

Since V_A and V_B are either type of S or C matrices as defined before, so the above steps can be realized by using FFT.

4.4 A Curved Boundary Problem

In this section, we point out that even if the computational domain is not an exact rectangle, the discrete trigonometrical transforms are still good choices as preconditioners.

Consider the following Poisson equations with Dirichlet boundary condition:

$$\Delta\psi = f, \quad (x, y) \in D = (x, y) : 0 < x < 1, c(x) < y < d(x) \tag{4.31}$$

where $\psi|_{\partial D} = 0, c(x) = 0.5(1-x^3), d(x) = 1+x^2, f = \Delta(sin^2(\pi x)sin^2(\pi\frac{y-c(x)}{d(x)-c(x)}))$. The image of the domain D is as follows:

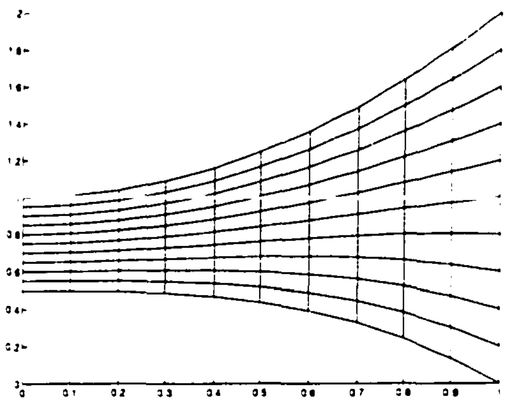

Figure 4.1: A curved boundary domain

Let us use third-order finite element method to find a approximate solution of (6.1). If we use the CG method without preconditioning for solving the resulting linear system, it diverges. If we use the discrete sine transformation as a preconditioner, we get clean high order accuracy for the PCG method, which is included in the following table:

Table 4.1: Accuracy when DST as preconditioner for Poisson equation on a curved domain (third-order triangle elements)

ncell	L^∞ error	order	L^1 error	order	L^2 error	order
$2 \cdot 20^2$	1.18E-5		2.46E-6		3.16E-6	
$2 \cdot 40^2$	8.31E-7	3.90	1.55E-7	4.01	1.98E-7	3.92
$2 \cdot 80^2$	5.47E-8	3.88	9.68E-9	3.99	1.24E-8	3.98

CHAPTER 5

COMPUTATION OF INTEGRALS AND ASSEMBLY TECHNIQUES

In this section, we will discuss the computational techniques for the integrals involved in the gauge finite method in this chapter and the simple finite element method in next chapter.

Let T_h be either a triangulation or a rectangulation of the domain Ω. All the nodes are put in some global order, i.e. each grid point has a global index, and also in each cell(i.e. a triangle or rectangle) each grid point has a local index. The following parameters and notations will be used.

$ncell$— the total number of cells

n—the total number of grid points

T_l— the lth cell in T_h, for $l = 1, \ldots, ncell$

$nsize$— the number of grid points in a single cell

$indx(l, i)$— the global index of the grid point which has local index i in the lth cell, for $l = 1, ncell, i = 1, \ldots, nsize$

$indb(i)$— the global index of the boundary points which has local index i, for $i = 1, \ldots, nbdy$, $nbdy$ is the total number of the boundary points.

$fc(l, i)$— the value of a function f at the grid point which has local index

i on lth cell, for $l = 1, \ldots, ncell, i = 1, \ldots, nsize$

$f(i)$—the value of a function f at the grid point which has global index i,for $i = 1, \ldots, n$

5.1 Assembly of the Nonlinear Terms

In the gauge method, we need to compute the nonlinear term

$$< (\vec{u}_h \cdot \nabla)\vec{u}_h, \phi >= \begin{pmatrix} \int_\Omega (u_h \frac{\partial u_h}{\partial x} + v_h \frac{\partial u_h}{\partial y})\phi_i dxdy \\ \int_\Omega (u_h \frac{\partial v_h}{\partial x} + v_h \frac{\partial v_h}{\partial y}\phi_i dxdy \end{pmatrix} \tag{5.1}$$

for $i = 1, \ldots, n$, where where $\phi_i, i = 1, \ldots, n$, are base functions of the finite element space.

In the simple finite element method, we need to compute the following nonlinear term

$$< \nabla\phi_i, \omega_h \vec{u}_h >\equiv \int_\Omega (\nabla\phi_i)^T \omega_h \vec{u}_h dxdy, \tag{5.2}$$

$i = 1, \ldots, n$.

Since the velocity function is given by $\vec{u}_h = \nabla^\perp\psi_h$, we will compute $< \nabla\phi_i, \omega_h\nabla^\perp\psi_h >$. All these nonlinear terms can be computed similarly. Here we only show how to compute the nonlinear term in the simple finite element method.

Suppose

$$\omega_h = \sum_{i=1}^n \omega_i\phi_i, \qquad \psi_h = \sum_{i=1}^n \psi_i\phi_i,$$

then the ith ($i = 1, \ldots, n$.) component of the nonlinear term is given by

$$\begin{aligned} rh(i) &\equiv < \nabla\phi_i, \omega_h\nabla^\perp\psi_h > \\ &= \int_\Omega \omega_h \frac{\partial\phi_i}{\partial x}(-\frac{\partial\psi_h}{\partial y})dxdy + \int_\Omega \omega_h \frac{\partial\phi_i}{\partial y}(\frac{\partial\psi_h}{\partial x})dxdy \\ &= -\int_\Omega(\textstyle\sum_{j=1}^n \omega_j\phi_j)\frac{\partial\phi_i}{\partial x}(\sum_{k=1}^n \psi_k\frac{\partial\phi_k}{\partial y})dxdy + \\ &+ \int_\Omega(\textstyle\sum_{j=1}^n \omega_j\phi_j)\frac{\partial\phi_i}{\partial y}(\sum_{k=1}^n \psi_k\frac{\partial\phi_k}{\partial x})dxdy \\ &= \textstyle\sum_{j,k=1}^n \omega_j\psi_k(\int_\Omega \phi_j\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_k}{\partial x}dxdy - \int_\Omega \phi_j\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_k}{\partial y}dxdy) \\ &= \textstyle\sum_{l=1}^{ncell}\sum_{j,k=1}^{nsize} \omega_j\psi_k(\int_{T_l} \phi_j\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_k}{\partial x}dxdy - \int_{T_l} \phi_j\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_k}{\partial y}dxdy) \end{aligned} \tag{5.3}$$

Since the finite element base functions have compact support on the cells in which their corresponding nodes are located, only when the ith, jth and kth nodes are on the lth cell, $\int_{T_l} \phi_j \frac{\partial\phi_i}{\partial x}\frac{\partial\phi_k}{\partial y}dxdy \neq 0$. Therefore to form $rh(i)$ we simply add together contributions from all cells which has the ith grid point.

Case 1: Triangle element

Suppose the vertices of T_l are $(x_0, y_0), (x_1, y_1), (x_2, y_2)$. Let $\hat{T}$ be the reference triangle with vertices $(0,0), (1,0), (0,1)$. Then we make the following transformation:

$$(5.4) \qquad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + J \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$$

where J is the Jacobian matrix given by

$$(5.5) \qquad J_l = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}$$

Then

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = J_l^{-1} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}$$

Let

$$(5.6) \qquad |J_l| \begin{pmatrix} \frac{\partial\hat{x}}{\partial x} \\ \frac{\partial\hat{y}}{\partial x} \end{pmatrix} \begin{pmatrix} \frac{\partial\hat{x}}{\partial y} \\ \frac{\partial\hat{y}}{\partial y} \end{pmatrix}^T = \begin{pmatrix} f_{l1} & f_{l2} \\ f_{l3} & f_{l4} \end{pmatrix}.$$

then

$$(5.7) \qquad \begin{aligned} &\int_{T_l} \phi_j \frac{\partial\phi_k}{\partial x}\frac{\partial\phi_i}{\partial y}dxdy \\ &= \int_{\hat{T}} \hat{\phi}_j \hat{\nabla}\hat{\phi}_k^T \begin{pmatrix} \frac{\partial\hat{x}}{\partial x} \\ \frac{\partial\hat{y}}{\partial x} \end{pmatrix} \begin{pmatrix} \frac{\partial\hat{x}}{\partial y} \\ \frac{\partial\hat{y}}{\partial y} \end{pmatrix}^T \hat{\nabla}\hat{\phi}_i abs(|J_l|)d\hat{x}d\hat{y} \\ &= \int_{\hat{T}} \hat{\phi}_j (f_{l1}\frac{\partial\hat{\phi}_k}{\partial\hat{x}}\frac{\partial\hat{\phi}_i}{\partial\hat{x}} + f_{l2}\frac{\partial\hat{\phi}_k}{\partial\hat{x}}\frac{\partial\hat{\phi}_i}{\partial\hat{y}} + f_{l3}\frac{\partial\hat{\phi}_k}{\partial\hat{y}}\frac{\partial\hat{\phi}_i}{\partial\hat{x}} + f_{l4}\frac{\partial\hat{\phi}_k}{\partial\hat{y}}\frac{\partial\hat{\phi}_i}{\partial\hat{y}})d\hat{x}d\hat{y} \end{aligned}$$

Define

$$\begin{aligned} coef(i,j,k,1) &= \int_{\hat{T}} \hat{\phi}_j \frac{\partial\hat{\phi}_k}{\partial\hat{x}}\frac{\partial\hat{\phi}_i}{\partial\hat{x}} d\hat{x}d\hat{y} \\ coef(i,j,k,2) &= \int_{\hat{T}} \hat{\phi}_j \frac{\partial\hat{\phi}_i}{\partial\hat{y}}\frac{\partial\hat{\phi}_k}{\partial\hat{x}} d\hat{x}d\hat{y} \\ coef(i,j,k,3) &= \int_{\hat{T}} \hat{\phi}_j \frac{\partial\hat{\phi}_i}{\partial\hat{x}}\frac{\partial\hat{\phi}_k}{\partial\hat{y}} d\hat{x}d\hat{y} \\ coef(i,j,k,4) &= \int_{\hat{T}} \hat{\phi}_j \frac{\partial\hat{\phi}_i}{\partial\hat{y}}\frac{\partial\hat{\phi}_k}{\partial\hat{y}} d\hat{x}d\hat{y} \end{aligned}$$

for $i,j,k = 1,\dots,nsize$. Then

$$\int_{T_l} \phi_j \frac{\partial\phi_k}{\partial x}\frac{\partial\phi_i}{\partial y}dxdy = \sum_{m=1}^{4} f_{lm} * coef(i,j,k,m).$$

Therefore

$$(5.8) \qquad \begin{aligned} &\textstyle\int_{T_l} \phi_j \frac{\partial\phi_i}{\partial y}\frac{\partial\phi_k}{\partial x}dxdy - \int_{T_l} \phi_j \frac{\partial\phi_i}{\partial x}\frac{\partial\phi_k}{\partial y}dxdy \\ &= \textstyle\sum_{m=1}^{4} f_{lm}(coef(i,j,k,m) - coef(k,j,i,m)) \end{aligned}$$

We make two arrays, $coef(1:nsize,1:nsize,1:nsize), factor(1:ncell,1:4)$ to store those $coef(i,j,k,m), f_{lm}$. Then assembly for the nonlinear term can be written as the following pseudo FORTRAN code:

```
Do $l = 1, ncell$
Do $i = 1, nsize$
    $temp = 0$
    $Do\ j = 1, nsize$
    $Do\ k = 1, nsize$
        $temp = temp + \omega c(l,j) * \psi c(l,k) *$
($\sum_{m=1}^{4} factor(l,m) * (coef(i,j,k,m) - coef(k,j,i,m))$)
    Enddo
    Enddo
    $rh(indx(l,i)) = rh(indx(l,i)) + temp$
Enddo
Enddo
```

Case 2 : Rectangle element

Suppose the vertices of cell T_l are $(x_0, y_0), (x_1, y_0), (x_0, y_1), (x_1, y_1)$. Let $\hat{T}$ be the reference rectangle with vertices $(0,0), (1,0), (0,1), (1,1)$. Then we make the following transform:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + J \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \tag{5.9}$$

where J is the Jacobian matrix given by

$$J_l = \begin{pmatrix} x_1 - x_0 & 0 \\ 0 & y_1 - y_0 \end{pmatrix}$$

Then

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = J_l^{-1} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}$$

It is easy to check that

$$\int_{T_l} \phi_j \frac{\partial \phi_k}{\partial x} \frac{\partial \phi_i}{\partial y} dx dy = \int_{T_l} \hat{\phi}_j \frac{\partial \hat{\phi}_k}{\partial \hat{x}} \frac{\partial \hat{\phi}_i}{\partial \hat{y}} d\hat{x} d\hat{y} \tag{5.10}$$

Define

$$\begin{aligned} coef(i, j, k, 1) &= \int_{\hat{T}} \hat{\phi}_j \frac{\partial \hat{\phi}_i}{\partial \hat{y}} \frac{\partial \hat{\phi}_k}{\partial \hat{x}} d\hat{x} d\hat{y} \\ coef(i, j, k, 2) &= \int_{\hat{T}} \hat{\phi}_j \frac{\partial \hat{\phi}_i}{\partial \hat{x}} \frac{\partial \hat{\phi}_k}{\partial \hat{y}} d\hat{x} d\hat{y} \end{aligned}$$

for $i, j, k = 1, \ldots, nsize$.

Then

$$\begin{aligned} &\textstyle\int_{T_l} \phi_j \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_k}{\partial x} dx dy - \int_{T_l} \phi_j \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_k}{\partial y} dx dy \\ &= (coef(i, j, k, 1) - coef(i, j, k, 2)) \end{aligned} \tag{5.11}$$

The assembly for the nonlinear term can be written as the following pseudo

FORTRAN code:

```
Do $l = 1, ncell$
Do $i = 1, nsize$
   $temp = 0$
   $Do\ j = 1, nsize$
   $Do\ k = 1, nsize$
      $temp = temp + wc(l, j) * wc(l, k)*$
$((coef(i, j, k, 1) - coef(i, j, k, 2))$
   Enddo
   Enddo
   $rh(indx(l, i)) = rh(indx(l, i)) + temp$
Enddo
Enddo
```

5.2 Unassembled Stiffness Matrix

The stiffness matrix is defined as $A \in \mathbf{R}^{n \times n}$, where $a_{ij} = (\nabla\phi_i, \nabla\phi_j), i, j, = 1, ..., n$. Since we will use preconditioned conjugate gradient (PCG) method to solve the derived linear systems of equations, S needs not to be formed explicitly. Instead, only the matrix vector production will need to be computed. For that purpose, we only compute the unassembled stiffness matrix $S \in \mathbf{R}^{ncell \times nsize \times nsize}$ which is defined as follows:

$$s_{lij} = \int_{T_l} (\nabla\phi_i)^T \nabla\phi_j dxdy, \tag{5.12}$$

for $l = 1, ..., ncell, i, j = 1, ..., nsize$ where ϕ_i, ϕ_j are the ith, jth base functions corresponding to the ith, jth nodes on the lth cell. As before, we will compute above the integral by mapping the lth cell T_l onto the reference cell $\dot{T}$.

Case 1: Triangle element

Suppose the vertices of T are $(x_0, y_0), (x_1, y_1), (x_2, y_2)$. As before, we have the transformation (5.4):

We have $\phi(x,y) = \hat{\phi}(\hat{x},\hat{y})$, and $\nabla\phi = J_l^{-T}\hat{\nabla}\hat{\phi}$ then

$$
\begin{aligned}
s_{kij} &= \textstyle\int_{T_l}(\nabla\phi_i)^T\nabla\phi_j dxdy \\
&= \textstyle\int_{\hat{T}}(J_l^{-T}\hat{\nabla}\hat{\phi}_i)^T(J_l^{-T}\hat{\nabla}\hat{\phi}_j)abs(|J_l|)d\hat{x}\hat{y} \\
&= \textstyle\int_{\hat{T}}(\hat{\nabla}\hat{\phi}_i)^T(J_l^{-1}J_l^{-T}abs(|J_l|)\hat{\nabla}\hat{\phi}_j)d\hat{x}\hat{y}
\end{aligned}
\tag{5.13}
$$

Denote $\begin{pmatrix} a_l & b_l \\ b_l & c_l \end{pmatrix} = abs(|J_l|)J_l^{-1}J_l^{-T}$ then

$$
s_{lij} = a_l\int_{\hat{T}}\frac{\partial\hat{\phi}_i}{\partial\hat{x}}\frac{\partial\hat{\phi}_j}{\partial\hat{y}}d\hat{x}d\hat{y} + +b_l\int_{\hat{T}}(\frac{\partial\hat{\phi}_i}{\partial\hat{x}}\frac{\partial\hat{\phi}_j}{\partial\hat{y}} + c_l\frac{\partial\hat{\phi}_i}{\partial\hat{y}}\frac{\partial\hat{\phi}_j}{\partial\hat{x}})d\hat{x}d\hat{y} + \int_{\hat{T}}\frac{\partial\hat{\phi}_i}{\partial\hat{y}}\frac{\partial\hat{\phi}_j}{\partial\hat{y}}d\hat{x}d\hat{y}
$$

For $i,j = 1,\ldots,nsize$, let

$$
\begin{aligned}
coef(i,j,1) &= \int_{\hat{T}}\frac{\partial\hat{\phi}_i}{\partial\hat{x}}\frac{\partial\hat{\phi}_j}{\partial\hat{x}}d\hat{x}d\hat{y} \\
coef(i,j,2) &= \int_{\hat{T}}(\frac{\partial\hat{\phi}_i}{\partial\hat{x}}\frac{\partial\hat{\phi}_j}{\partial\hat{y}} + \frac{\partial\hat{\phi}_i}{\partial\hat{y}}\frac{\partial\hat{\phi}_j}{\partial\hat{x}})d\hat{x}d\hat{y} \\
coef(i,j,3) &= \int_{\hat{T}}\frac{\partial\hat{\phi}_i}{\partial\hat{y}}\frac{\partial\hat{\phi}_j}{\partial\hat{y}}d\hat{x}d\hat{y}
\end{aligned}
$$

then

$$
s_{lij} = a_l * coef(i,j,1) + b_l * coef(i,j,2) + c_l * coef(i,j,3) \tag{5.14}
$$

for $l = 1,\ldots,ncell$, $i,j = 1,\ldots,nsize$.

Case 2: Rectangle element

Suppose the vertices of T_l are $(x_0,y_0),(x_0,y_1),(x_1,y_0),(x_1,y_1)$. As before, we have the transform (5.9).

We have $\phi(x,y) = \hat{\phi}(\hat{x},\hat{y})$, and $\nabla\phi = J_l^{-T}\hat{\nabla}\hat{\phi}$ then

$$
\begin{aligned}
s_{lij} &= \textstyle\int_{T_l}(\nabla\phi_i)_l^T\nabla\phi_j dxdy \\
&= \textstyle\int_{\hat{T}}(J_l^{-T}\hat{\nabla}\hat{\phi}_i)^T(J_l^{-T}\hat{\nabla}\hat{\phi}_j)abs(|J_l|)d\hat{x}\hat{y} \\
&= \textstyle\int_{\hat{T}}(\hat{\nabla}\hat{\phi}_i)^T(J_l^{-1}J_l^{-T}abs(|J_l|)\hat{\nabla}\hat{\phi}_j)d\hat{x}\hat{y}
\end{aligned}
$$

Since $\begin{pmatrix} \frac{y_1-y_0}{x_1-x_0} & 0 \\ 0 & \frac{x_1-x_0}{y_1-y_0} \end{pmatrix} = abs(|J_l|)J_l^{-1}J_l^{-T}$, then

$$s_{l_{ij}} = \frac{y_1 - y_0}{x_1 - x_0} \int_{\hat{T}} \frac{\partial \hat{\phi}_i}{\partial \hat{x}} \frac{\partial \hat{\phi}_j}{\partial \hat{y}} d\hat{x} d\hat{y} + + \frac{x_1 - x_0}{y_1 - y_0} \int_{\hat{T}} \frac{\partial \hat{\phi}_i}{\partial \hat{y}} \frac{\partial \hat{\phi}_j}{\partial \hat{y}} d\hat{x} d\hat{y}$$

For $i, j = 1, \dots, nsize$, let

$$\begin{aligned} coef(i,j,1) &= \int_{\hat{T}} \frac{\partial \hat{\phi}_i}{\partial \hat{x}} \frac{\partial \hat{\phi}_j}{\partial \hat{x}} d\hat{x} d\hat{y} \\ coef(i,j,2) &= \int_{\hat{T}} \frac{\partial \hat{\phi}_i}{\partial \hat{y}} \frac{\partial \hat{\phi}_j}{\partial \hat{y}} d\hat{x} d\hat{y} \end{aligned}$$

then

$$s_{l_{ij}} = \frac{y_1 - y_0}{x_1 - x_0} * coef(i,j,1) + \frac{x_1 - x_0}{y_1 - y_0} * coef(i,j,2) \tag{5.15}$$

for $l = 1, \dots, ncell, i, j = 1, \dots, nsize$.

5.3 Unassembled Mass Matrix

Similarly to the stiffness matrix, the mass matrix $B = (b_{ij}) \in \mathbf{R}^{n \times n}$ needs only be computed in unassembled form, where $b_{ij} = \int_\Omega \phi_i \phi_j dxdy, i, j = 1, \dots, n$, the indices of the base functions are global.

The unassembled mass matrix is defined as

$$M = (m_{l_{ij}}) \in \mathbf{R}^{ncell \times nsize \times nsize}. \tag{5.16}$$

where $m_{l_{ij}} = \int_{T_l} \phi_i \phi_j dxdy, l = 1, \dots, ncell, i, j = 1, \dots, nsize$, the indices of the base functions are local. Again we make the same mapping as before, define $coef(i,j) = \int_{\hat{T}} \hat{\phi}_i \hat{\phi}_j d\hat{x} d\hat{y}, i, j = 1, \dots, nsize$, then we have

$$m_{l_{ij}} = abs(|J_l|) * coef(i,j) \tag{5.17}$$

for $l = 1, \dots, ncell, i, j = 1, \dots, nsize$.

The above formula holds for both triangle and rectangle elements.

5.4 Assembly of Matrix-Vector Product

Suppose we have an unassembled matrix $A = (a_{kij})$, and we are going to use PCG method to solve $\hat{A}x = b$, where the ith component of x is solution corresponding to ith node points. In PCG method, the major computation is the matrix-vector product $v = \hat{A}u$. We have the following pseudo FORTRAN code for the product:

```
Initialization  $v = 0$
Do $k = 1, ncell$
Do $i = 1, nsize$
    $temp = 0$
    Do $j = 1, nsize$
        $temp = temp + a(k,i,j) * uc(k,j)$
    Enddo
    $v(indx(k,i)) = v(indx(k,i)) + temp$
Enddo
Enddo
```

In the case of the solutions on the boundary are known, we need to set the product on the boundary to zero: $v(indb(i)) = 0, i = 1, \ldots, nbdy$.

5.5 Evaluation of the Gradient Terms in Gauge Method

Given ϕ in $(k+1)$th order finite element space X_h^{k+1}, we explain how to $\nabla\phi$ in kth order finite element space X_h^k.

Suppose $\psi_i^{k+1}, i = 1, \ldots, n_{k+1}$, are base functions of X_h^{k+1}, $\psi_i^k, i = 1, \ldots, n_k$, are base functions of X_h^k. Suppose $\phi = \Sigma_{i=1}^{n_{k+1}} \phi_i^{k+1} \psi_i^{k+1}$, then $\frac{\partial \phi}{\partial x} = \Sigma_{i=1}^{n_{k+1}} \phi_i^{k+1} \frac{\partial \psi_i^{k+1}}{\partial x}$. On the other hand, let $\frac{\partial \phi}{\partial x} = \Sigma_{i=1}^{n_k} x_i \psi_i^k$, where x_i is the value of $\frac{\partial \phi}{\partial x}$ at the ith grid point in the kth order mesh. Then $\Sigma_{i=1}^{n_k} x_i \psi_i^k = \Sigma_{i=1}^{n_{k+1}} \phi_i^{k+1} \frac{\partial \psi_i^{k+1}}{\partial x}$, thus we have

$$(5.18) \qquad \Sigma_{i=1}^{n_k} x_i(\psi_i^k, \psi_j^k) = \Sigma_{i=1}^{n_{k+1}} \phi_i^{k+1} (\frac{\partial \psi_i^{k+1}}{\partial x}, \psi_j^k), j = 1, \ldots, n^k.$$

Solve this system, we can get the values of $\frac{\partial \phi}{\partial x}$ on the kth order mesh.

Note that the coefficient matrix of the above linear system is the mass matrix, of which the unassembled form has been discussed before. Now let discuss in more details the computation of the right hand side. We would like to point out that we can find by using divergence theorem that the right hand side of Neumann problem in the step 2 in gauge method has similar structure as in (5.18).

To simplify the notation, let us assume that $k = 3$. So we use mixed 3rd/4th finite elements in the gauge method.

As before we will first compute the integral by mapping the cells to the reference cell, then we will use local assembly techniques to form the right hand side.

Case 1: Triangle element

Suppose the vertices of T are $(x_0, y_0), (x_1, y_1), (x_2, y_2)$. As before, we have the transform (5.4).

Note $\psi(x, y) = \hat{\psi}(\hat{x}, \hat{y})$, denote that

$$(5.19) \qquad \begin{pmatrix} f_1 & f_2 \\ f_3 & f_4 \end{pmatrix} = |J_l| J_l^{-T},$$

then

$$\begin{aligned} &\int_{T_l} \frac{\partial \psi_i^4}{\partial x} \psi_j^3 dx dy \\ &= f_1 \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{x}} \hat{\psi}_j^3 d\hat{x} d\hat{y} + f_2 \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{y}} \hat{\psi}_j^3 d\hat{x} d\hat{y}, \\ &\int_{T_l} \frac{\partial \psi_i^4}{\partial y} \psi_j^3 dx dy \\ &= f_3 \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{x}} \hat{\psi}_j^3 d\hat{x} d\hat{y} + f_4 \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{y}} \hat{\psi}_j^3 d\hat{x} d\hat{y}, \end{aligned}$$

therefore, if we define

$$\begin{aligned} coef(i, j, 1) &= \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{x}} \hat{\psi}_j d\hat{x} d\hat{y} \\ coef(i, j, 2) &= \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{y}} \hat{\psi}_j d\hat{x} d\hat{y} \end{aligned}$$

then we have the following pseudo Fortran code for the local assembly of the right hand sides:

```
initialization $rhx(i) = 0, rhy(i) = 0, i = 1, ...., node3$
Do $k = 1, ncell$
Do $i = 1, nsize3$
    $temp1 = 0, temp2 = 0$
    Do $j = 1, nsize4$
        $temp1 = temp1 + phic(k,j) * (f_2 * coef(j,i,1) + f_1 * coef(j,i,2))$
        $temp2 = temp2 + phic(k,j) * (f_4 * coef(j,i,1) + f_3 * coef(j,i,2))$
    Enddo
    $rhx(indx3(k,i)) = rhx(indx3(k,i)) + temp1$
    $rhy(indx3(k,i)) = rhy(indx3(k,i)) + temp2$
Enddo
Enddo
```

Case 2: Rectangle element

Suppose the vertices of T_l are $(x_0, y_0), (x_0, y_1), (x_1, y_0), (x_1, y_1)$. As before, we have the transform (5.9).

We have $\phi(x,y) = \hat{\phi}(\hat{x},\hat{y})$, and $\nabla\phi = J_l^{-T}\hat{\nabla}\hat{\phi}$ denote that

$$f_1 = x_1 - x_0, \quad f_2 = y_1 - y_0$$

then

$$\int_{T_l} \frac{\partial \psi_i^4}{\partial x} \psi_j^3 dxdy = f_2 \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{x}} \hat{\psi}_j^3 d\hat{x}d\hat{y}$$

and

$$\int_{T_l} \frac{\partial \psi_i^4}{\partial y} \psi_j^3 dxdy = f_1 \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{y}} \hat{\psi}_j^3 d\hat{x}d\hat{y}$$

therefore, if we define

$$coef(i,j,1) = \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{x}} \hat{\psi}_j d\hat{x}d\hat{y}$$
$$coef(i,j,2) = \int_{\hat{T}} \frac{\partial \hat{\psi}_i^4}{\partial \hat{y}} \hat{\psi}_j d\hat{x}d\hat{y}$$

then we have the following pseudo Fortran code for the local assembly of the right hand sides:

```
initialization:  $rhx(i) = 0, rhy(j) = 0, i = 1, ...., node3$
Do $k = 1, ncell$
Do $i = 1, nsize3$
   $temp1 = 0, temp2 = 0$
   Do $j = 1, nsize4$
      $temp1 = temp1 + phic(k, j) * f_2 * coef(j, i, 1)$
      $temp2 = temp2 + phic(k, j) * f_1 * coef(j, i, 2)$
   Enddo
   $rhx(indx3(k, i)) = rhx(indx3(k, i)) + temp1$
   $rhy(indx3(k, i)) = rhy(indx3(k, i)) + temp2$
Enddo
Enddo
```

CHAPTER 6

NUMERICAL EXPERIMENTS

6.1 Accuracy Check for the Gauge Finite/Spectral Element Methods

Our first experiment is to check the convergence order of accuracy. Choose the following exact solution of the Navier-Stokes equation:

$$
(6.1) \qquad \begin{cases} u1(x, y, t) = -cos(t)sin^2(\pi x)sin(2\pi y) \\ u2(x, y, t) = cos(t)sin(2\pi x)sin^2(\pi y) \\ \phi(x, y, t) = cos(t)(2 + cos(\pi x))(2 + cos(\pi y))/4 \end{cases}
$$

Corresponding forcing terms are added to ensure that (6.1) is an exact solution.

We first use mixed 3rd/4th order gauge finite element method. that is. we use 3rd order piecewise polynomials to approximate $\vec{a}, \vec{u}$, and use 4th order piecewise polynomials to approximate ϕ.

We choose $Re = 1, \Delta t = (h)^{\frac{3}{2}}$, where h is the minimum side of the triangles. The absolute errors and the convergence orders are reported in the following table.

The second example is to use mixed 5th/6th spectral rectangle element method. Here we use rectangle elements, and the zero points of Gauss-Lobetto-

Table 6.1: Errors for Backward-Euler Gauge Finite Element Methods (mixed 3rd/4th order triangle elements), Re=1000, t=1

Gauge FEM	ncell	L^∞ error	order	L^1 error	order	L^2 error	order
p	$2 \cdot 10^2$	4.60E-4		1.27E-4		1.53E-4	
	$2 \cdot 20^2$	1.07E-4	2.10	3.53E-5	1.84	4.08E-5	1.90
	$2 \cdot 30^2$	4.61E-5	2.07	1.64E-5	1.89	1.87E-5	1.92
ul	$2 \cdot 10^2$	2.20E-3		5.37E-4		7.02E-4	
	$2 \cdot 20^2$	5.51E-4	2.00	1.29E-4	2.05	1.70E-4	2.04
	$2 \cdot 10^2$	2.45E-4	2.00	5.70E-5	2.02	7.48E-5	2.02
u2	$2 \cdot 10^2$	2.20E-3	2.00	4.97E-4	2.02	6.62E-4	2.02
	$2 \cdot 20^2$	5.51E-4	2.00	1.17E-4	2.08	1.58E-4	2.06
	$2 \cdot 30^2$	2.45E-4	2.00	5.15E-5	2.03	6.94E-5	2.03
div	$2 \cdot 10^2$	1.49E-2		1.29E-3		2.30E-3	
	$2 \cdot 20^2$	3.56E-3	2.06	7.33E-5	4.13	1.96E-4	3.55
	$2 \cdot 30^2$	1.57E-3	2.02	1.44E-5	4.02	4.93E-5	3.40

Table 6.2: Errors for Crank-Nicholson Gauge Finite Element Methods (mixed third/forth-order triangle finite elements), Re=1000, t=1

Gauge FEM	ncell	L^∞ error	order	L^1 error	order	L^2 error	order
p	$2 \cdot 20^2$	7.27E-3		2.62E-4		4.46E-4	
	$2 \cdot 40^2$	6.13E-4	3.56	2.59E-5	3.34	4.33E-5	2.47
	$2 \cdot 60^2$	1.77E-4	3.07	9.52E-6	2.47	1.41E-5	2.77
ul	$2 \cdot 20^2$	1.21E-5		1.52E-6		2.34E-6	
	$2 \cdot 40^2$	8.09E-7	3.90	1.12E-7	3.76	1.65E-7	3.82
	$2 \cdot 60^2$	1.97E-7	3.48	2.64E-8	3.56	3.83E-8	3.60
u2	$2 \cdot 20^2$	1.20E-5		1.60E-6		2.40E-6	
	$2 \cdot 40^2$	8.08E-7	3.89	1.21E-7	3.72	1.75E-7	3.78
	$2 \cdot 60^2$	1.99E-7	3.45	2.87E-8	3.55	4.12E-8	3.57
div	$2 \cdot 20^2$	1.50E-3		1.17E-4		2.13E-4	
	$2 \cdot 40^2$	1.93E-4	2.96	1.33E-5	3.14	2.33E-5	3.20
	$2 \cdot 60^2$	5.82E-5	2.96	3.79E-6	3.09	6.48E-6	3.14

Legendre polynomials, see e.g. [20], are used as grid points. Again Crank-Nicholson method is used for time discretization. $Re = 1, \Delta t = h^{2.5}$. The absolute errors and convergence orders are reported in the following table.

Table 6.3: Errors for Crank-Nicholson Gauge Spectral Element Methods (mixed 5th/6th order rectangle elements), Re=1000, t=1

Gauge SEM	ncell	L^∞ error	order	L^1 error	order	L^2 error	order
p	5^2	1.54E-4		2.63E-5		4.09E-5	
	8^2	2.76E-5	3.67	3.55E-6	4.27	5.61E-6	4.22
	10^2	9.26E-6	4.89	1.11E-6	5.20	1.76E-6	5.20
u1	5^2	1.47E-6		1.72E-7		2.98E-7	
	8^2	1.61E-7	4.70	1.60E-8	5.08	2.91E-8	4.95
	10^2	4.38E-8	5.84	3.96E-9	6.20	7.56E-9	6.05
u2	5^2	1.47E-6		1.72E-7		2.99E-7	
	8^2	4.43E-5	4.70	3.48E-6	5.08	7.32E-6	4.95
	10^2	4.38E-8	5.84	3.97E-9	6.20	7.57E-9	6.05
div($\vec{u}$)	5^2	2.94E-4		3.61E-5		6.79E-5	
	8^2	4.43E-5	4.03	3.48E-6	4.97	7.32E-6	4.74
	10^2	1.50E-5	4.84	1.09E-6	5.19	2.27E-6	5.24

In the above tables, the square numbers are the numbers of element cells, u is the velocity, p is pressure, and $div(u)$ is divergence of u. Ord refers to the order of convergence, and $L1, L2, L_\infty$ are norms of the errors . The plots of errors for the case of Crank-Nicholson gauge spectral element method are given in the pictures 6.1-6.4.

6.2 Accuracy Check for the Simple Finite/Spectral Element Methods

Our first numerical experiment is to check the convergence accuracy of the simple finite element method by choosing the following exact solution of the

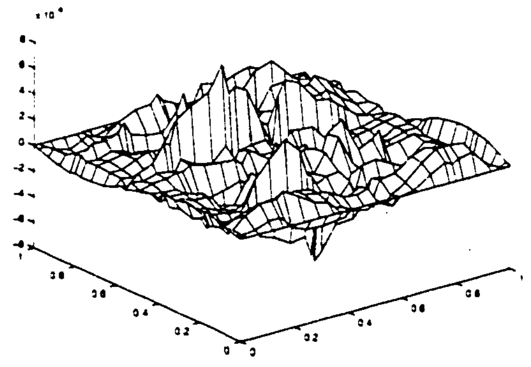

Figure 6.1: Error of the first component of velocity for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order, Re=1000, t=1, cell number=5^2

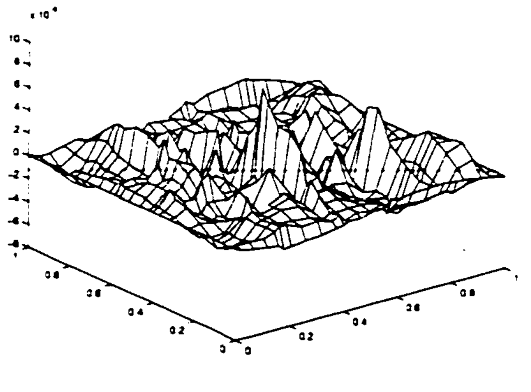

Figure 6.2: Error of the second component of velocity for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order, Re=1000, t=1 , cell number=5^2

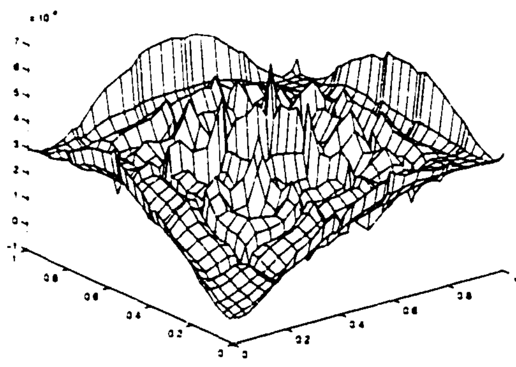
Figure 6.3: Error of pressure for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order. Re=1000. t=1 . cell number=5^2

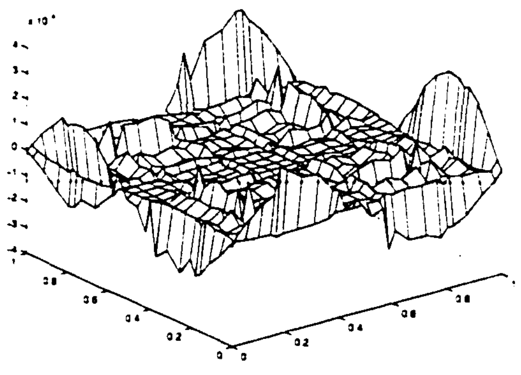
Figure 6.4: Error of divergence-free property for Crank-Nicholson gauge spectral rectangle element method of mixed 5th/6th order. Re=1000. t=1 . cell number=5^2

Navier-Stokes equations:

$$(6.2)\qquad \begin{cases} \psi = cos(t)sin^2(\pi x)sin^2(\pi y), \\ \omega = 2\pi^2 cos(t)(cos(2\pi x)sin^2(\pi y) + cos(2\pi y)sin^2(\pi x)) \end{cases}$$

where $(x, y) \in \Omega = (0, 1) \times (0, 1), t \in (0, 1)$, the force term and boundary conditions are determined by ψ and ϕ. The initial value is the exact solution.

We use finite element method(FEM) and spectral element method(SEM) for our computation. We use triangle elements for the case of FEM and rectangle element for the case of SEM.

We choose $Re = 10000, \Delta t = h/4$, where h is the minimum side of all the triangles (or rectangles). The comparison of the numerical solution and the exact solution at time 1 and the order of accuracy is reported in the following tables. The first table gives the results when the third-order FEM is used. The second table gives the results when the fifth-order SEM is used.

Table 6.4: Errors for Simple Finite Element Methods (third-order triangle finite element method), Re=1000, t=1

Simple FEM	ncell	L^∞ error	order	L^1 error	order	L^2 error	order
	$2 \cdot 20^2$	2.62E-5		5.33E-6		7.45E-6	
ψ	$2 \cdot 40^2$	1.15E-6	4.52	2.55E-7	4.38	3.65E-7	4.35
	$2 \cdot 80^2$	4.47E-8	4.68	6.80E-9	5.23	9.82E-9	5.21
	$2 \cdot 20^2$	0.20		6.44E-3		1.51E-2	
ω	$2 \cdot 40^2$	3.76E-2	2.41	4.11E-4	3.97	1.94E-3	2.96
	$2 \cdot 80^2$	9.00E-3	2.06	4.02E-5	3.36	3.30E-4	2.56

The second table is the results obtained by using spectral element method.

We also plot the errors for the vorticity and stream functions.

Table 6.5: Errors for Simple Spectral Element Methods (fifth-order rectangle elements), Re=1000, t=1

Simple SEM	ncell	L^∞ error	order	L^1 error	order	L^2 error	order
	5^2	5.31E-6		1.02E-6		1.45E-6	
ψ	10^2	5.77E-8	6.53	8.02E-9	7.00	1.26E-8	6.84
	20^2	1.77E-10	8.35	3.23E-11	7.96	4.47E-11	8.15
	5^2	9.60E-3		7.12E-4		1.41E-3	
ω	10^2	2.51E-4	5.26	9.33E-6	6.27	2.49E-5	5.82
	20^2	1.03E-6	7.93	4.21E-8	7.79	9.04E-8	8.10

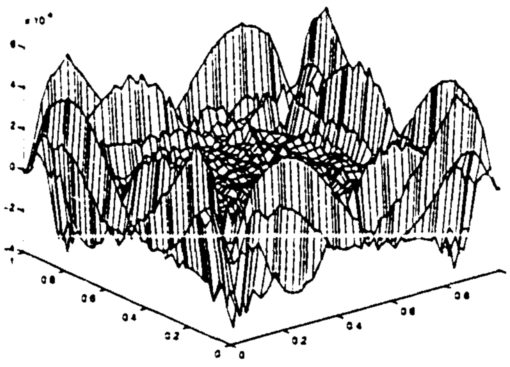

Figure 6.5: Error of the stream function in 5th-order simple spectral element method, Re=1000, t=1 , cell number=10^2

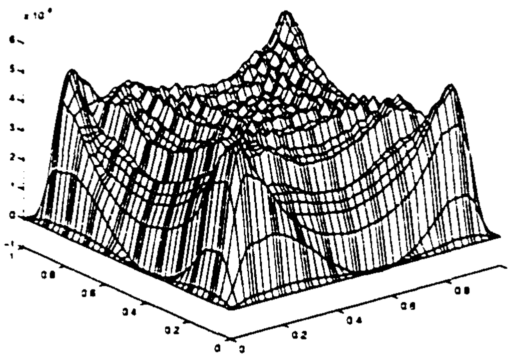

Figure 6.6: Error of the vorticity function in 5th-order simple spectral element method, Re=1000, t=1, cell number=10^2

6.3 Simulations of the Driven-Cavity Flows

Our second example is the simulation of the classic driven cavity flow problem, see e.g. [8, 9, 13]. The flow domain is $[0, 1] \times [0, 1]$, with the no-slip condition imposed. The upper boundary moves with smooth velocity $u_b(x) = 16x^2(1-x)^2$ and initial data: $\omega_0(x, y) = (y^2 - y^3)u_b(x)$. We have also tried the more conventional and difficult discontinuous boundary condition $u_b(x) = 1$. The results are presented in the following pictures.

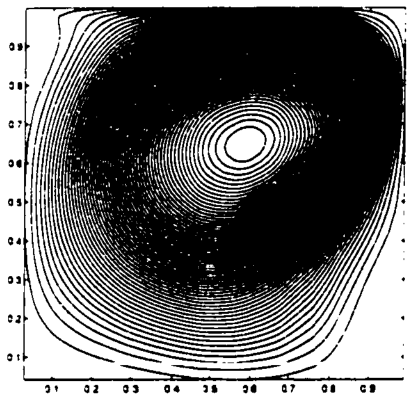

Figure 6.7: stream function, time=5, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2

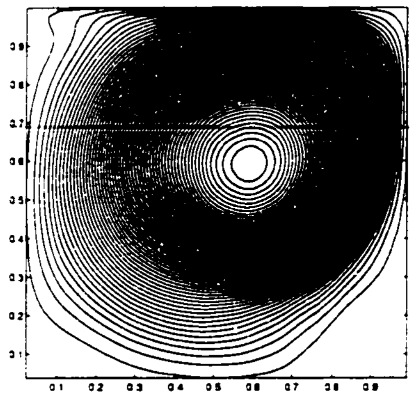

Figure 6.8: stream function, time=10, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2

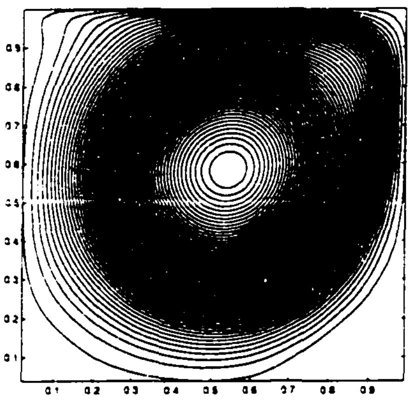

Figure 6.9: stream function, time=15, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2

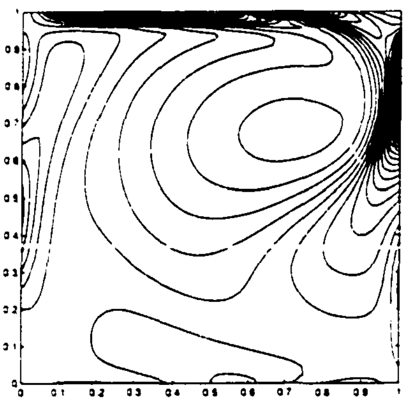

Figure 6.10: vorticity function, time=5, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method. cell number=15^2

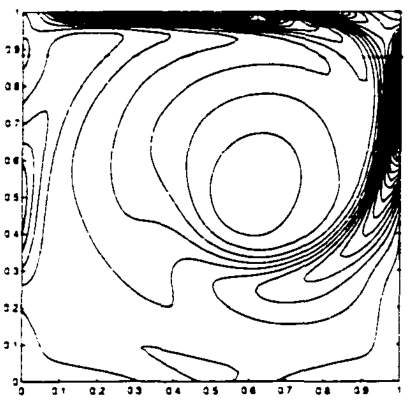

Figure 6.11: vorticity function, time=10, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method. cell number=15^2

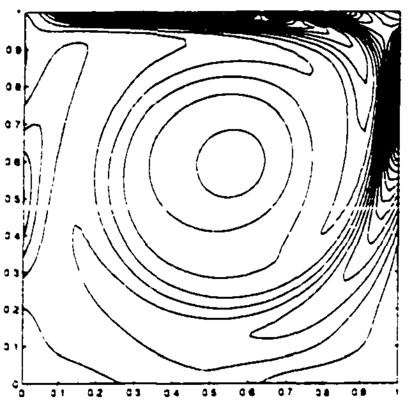

Figure 6.12: vorticity function, time=15, Re=1000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=15^2

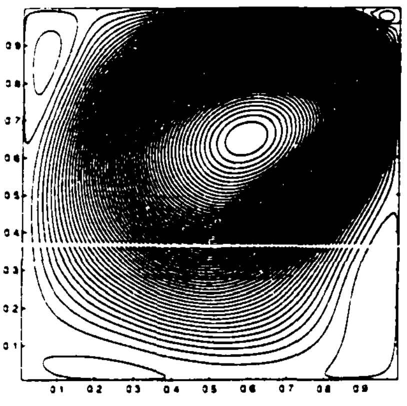

Figure 6.13: stream function, time=1, Re=100000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=25^2

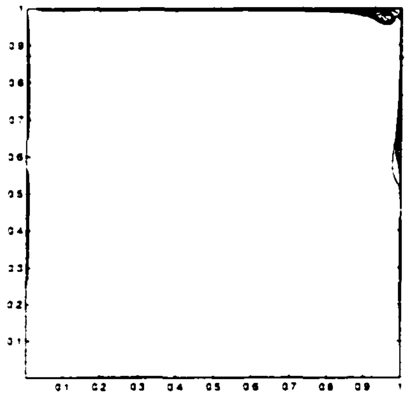

Figure 6.14: vorticity function, time=1, Re=100000, smooth boundary condition, 5th order simple spectral rectangle element method, cell number=25^2

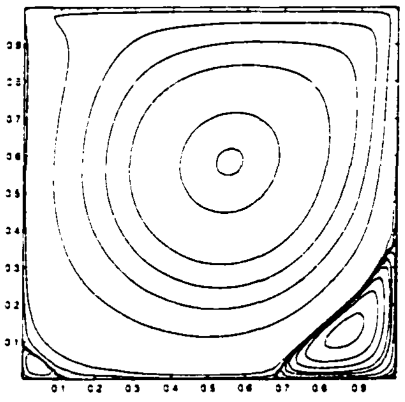

Figure 6.15: steady state of stream function,Re=1000. discontinuous boundary condition. 5th order simple spectral rectangle element method. cell number=15^2

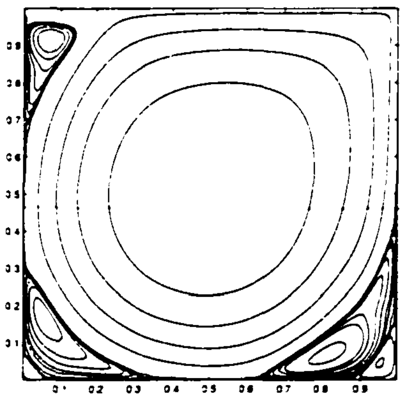

Figure 6.16: steady state of stream function, Re=7500. discontinuous boundary condition. 5th order simple spectral rectangle element method. cell number=25^2

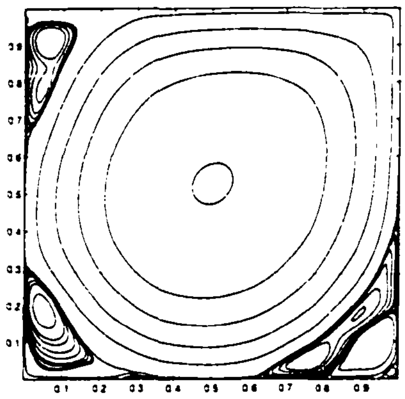

Figure 6.17: steady state of stream function, Re=10000, discontinuous boundary condition, 5th order simple spectral rectangle element method, cell number=25^2

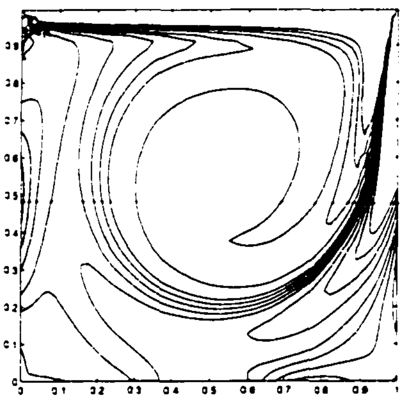

Figure 6.18: steady state of vorticity function, Re=1000, discontinuous boundary condition, 5th order simple spectral rectangle element method, cell number=15^2

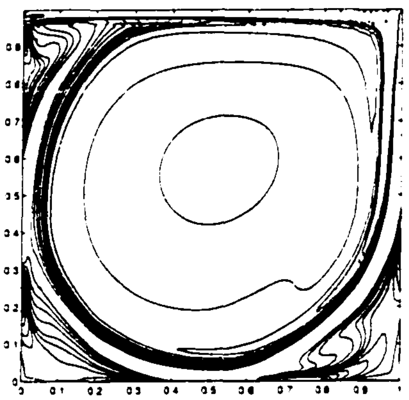

Figure 6.19: steady state of vorticity function, Re=7500, discontinuous boundary condition, 5th order simple spectral rectangle element method, cell number=25^2

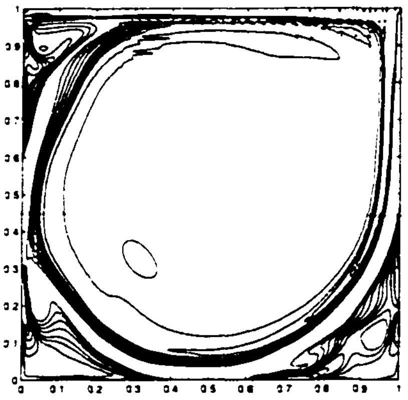

Figure 6.20: steady state of vorticity function, Re=10000, discontinuous boundary condition, 5th order simple spectral rectangle element method, cell number=25^2

REFERENCES

[1] Barragy. E.. and Carey. G.F.. *Stream function vorticity solution using high-p element-by-element techniques.* Commun. in Applied Numer. Methods, **9**:387-395. 1993.

[2] Barrett. K.E.. *A variational approach for the vector otential formulation of the Stokes and Navier-Stokes problems in three dimensional domains,* J. Math. Anal. Appl., **107**: 537-560. 1985.

[3] Buttke. T.. *Velocity methods: Langrangian numerical methods which preserve the Hamiltonian structure of incompressible fluid flow.* Vortex flows and related numerical methods. J.T. Beale. G.H. Cottet and S. Huberson er.. 1993.

[4] Campion-Renson. A.. and Crochet. M.J.. *On the stream function-vorticity finite element solutions of Navier-Stokes equations.* Int. J. Num. Meth. Emgng. **12**: 1809-1818. 1978

[5] Chorin. A.J.. *Numerical solution of the Navier-Stokes equations.* Math. Comp., **22**: 745-762. 1968.

[6] Chorin. A.J.. and Marsden. J.E.. *A mathematical introduction to fluid mechanics.* Springer-Verlag, 1993.

[7] Ciarlet. P., *The finite element method for elliptic prolems.* North-Holland. Amsterdam, 1978.

[8] E, W., and Liu, J.-G., *Vorticity boundary condition and related issues for finite difference schemes*, J. Comput. Phys., **124**: 368-382, 1996.

[9] E, W. and Liu, J.-G., *Essentially compact schemes for unsteady viscous incompressible flows*, J. Comput. Phys., **126**: 122-138, 1997.

[10] E, W., and Liu, J.-G., *Gauge method for viscous incompressible flow*, Submitted to J. Comput. Phys., 1996

[11] E, W., and Liu, J.-G., *Finite difference schemes for incompressible flows in the impulse-velocity formulation*, J. Comput. Phys., **130**: 76-76, 1997.

[12] Gear, C.W., *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1971.

[13] Ghia, U., Ghia, K.N., and Shin, C.T., *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method*, J. Comput. Phys., **48**, 387-411, 1982.

[14] Greenbaum, A., *Iterative methods for solving linear systems*, SIAM 1997.

[15] Johnson, C., *Numerical solutions of partial differential equations by the finite element method*, Cambridge University Press, Cambridge, UK, 1987.

[16] Gresho, P.M., and Sani, R.L., *On pressure boundary conditions for the incompressible Navier-Stokes equations*, Int. J. Numer. Methods Fluids, **7**: 1111-1145, 1987.

[17] Hestenes, M.R., and Stiefel, E.L., *Methods of conjugate gradients for solving linear systems.* Journal of Research of the National Bureau of Standards, Section B, **49**, 409-436, 1952

[18] Ikegawa, M., *A new finite element technique for the analysis of steady viscous flow problems,* Int. J. Num. Meth. Engng. **14**, 103-113, 1979

[19] Karniadakis, G.E., Israsei, M., and Orszag, S.A., *High-Order splitting methods for the incompressible Navier-Stokes equations,* J. Comput. Phys., **97**, 414-443, 1991

[20] Karniadakis, G.E., and S. Sherwin, *Spectral/Hp element methods for Cfd,* Oxford University Press, 1998

[21] Lanczos, C., *Solution of systems of linear quations by minimized iterations.* J. Research of the National Bureau of Standards, **49**, 33-53, 1952

[22] Liu J.-G., and E, W., *Simple finite element method in vor ticity formulation for incompressible flows,* To appear in Math. Comp.

[23] Maddocks, J.H., and Smereka, P., *An unconstrained hamiltonian formulation for incompressible fluid,* Comm. Math. Phys., 170:207-217, 1995.

[24] Oseledets, V.I., *On a new way of writing the Navier-Stokes equations: the Hamiltonian formulation,* J. Russ. Math. Survery, **44**: 210-211, 1989.

[25] R. Peyret, P., and Taylor, T.D., *Computational methods for fluid flow,* Springer-Verlag, 1990.

[26] Pyo, J.-H., *Error estimates for semi-discrete gauge methods for evolution Navier-Stokes Equations: first-order schemes,* preprint

[27] Reid, J.K., *On the method of conjugate gradients for the solution of large sparse systems of linear equations.* In J. K. Reid, editor, *Large sparse sets of linear equaitons*, Academic Press, 231-254, 1971.

[28] Russo G., and Smereka, P., *Impulse formulation of the Euler equaiton: general properties and numerical methods.* J. Fluid Mech., **391**:189-209, 1999

[29] Saad, Y., *Iterative methods for large sparse linear systems.* PWS publishing company, 1997.

[30] Stevens, W.N.R., *Finite element, stream function-vorticity solution of steady laminar natural convection*, Int. J. Num. Meth. Fluids, **2**, 349-366, 1982.

[31] Swarztrauber, P.N. *The methods of cyclic reductions, Fourier analysis, and the FACR algorithm for the discrete solution of Poisson's equaiton on a rectangle*, SIAM Rev, **19**, 490-501, 1977.

[32] Sweet, R., Lindgren, L.L., and Boisvert, R.R., *VFFTPK: a vectorized package of Fortran subgrograms for the fast Fourier transform of multiple real sequences*, Reprint, May 1990.

[33] Teman, R. *Sur l'approximation de la solution des equations de Navier-Stokes par la méthode des fractionnaries II*, Arch. Rational Mech. Anal., **33**:377-385, 1969.

[34] Van Loan, C., *Computational frameworks for the fast Fourier transform*, SIAM, 1992.

[35] Wang C., and Liu, J.-G., *Convergence of gauge method for incompressible flow*, to appear in Math. Comp.

APPENDIX A

MAPLE CODES FOR THE COMPUTATION OF SOME INTEGRALS ON THE REFERENCE TRIANGLE

1. coefficients for the nonlinear term by using forth-order triangle elements.

```
bas4:=array(1..15);
coeft:=array(1..15,1..15,1..15,1..2);
l1:=x;
l2:=y;
l3:=1-x-y;
bas4[1]:=32/3*l3*(l3-1/4)*(l3-1/2)*(l3-3/4);
bas4[2]:=128/3*l1*l3*(l3-1/4)*(l3-1/2);
bas4[3]:=64*l3*l1*(l1-1/4)*(l3-1/4);
bas4[4]:=128/3*l1*l3*(l1-1/4)*(l1-1/2);
bas4[5]:=32/3*l1*(l1-1/4)*(l1-1/2)*(l1-3/4);
bas4[6]:=128/3*l2*l3*(l3-1/2)*(l3-1/4);
bas4[7]:=128*l1*l2*l3*(l3-1/4);
```

```
bas4[8]:=128*l1*l2*l3*(l1-1/4);
bas4[9]:=128/3*l1*l2*(l1-1/4)*(l1-1/2);
bas4[10]:=64*l2*l3*(l2-1/4)*(l3-1/4);
bas4[11]:=128*l1*l2*l3*(l2-1/4);
bas4[12]:=64*l1*l2*(l1-1/4)*(l2-1/4);
bas4[13]:=128/3*l2*(l2-1/4)*(l2-1/2)*l3;
bas4[14]:=128/3*l1*l2*(l2-1/2)*(l2-1/4);
bas4[15]:=32/3*l2*(l2-1/4)*(l2-1/2)*(l2-3/4);
with(linalg);
for j from 1 to 15 do
for k from 1 to j do
for i from 1 to 15 do
        grad1:=grad(bas4[i],[x,y]);
        u1:=expand(grad1[1]*bas4[j]*bas4[k]);
        u2:=expand(grad1[2]*bas4[j]*bas4[k]);
        coeft[i,j,k,1]:=int(int(u1,y=0..1-x),x=0..1);
        coeft[i,j,k,2]:=int(int(u2,y=0..1-x),x=0..1);
od;
od;
od;
for j from 1 to 14 do
for k from j+1 to 15 do
for i from 1 to 15 do
        coeft[i,j,k,1]:=coeft[i,k,j,1];
        coeft[i,j,k,2]:=coeft[i,k,j,2];
od;
od;
od;
precision:=double;
fortran(coeft, filename=temp1f);
```

2. coefficients for the evaluation of the gradients by using mixed 3rd/4th order triangle finite elements

```
bas3:=array(1..10);
bas4:=array(1..15);
coeft:=array(1..10,1..15,1..2);
l1:=x;
l2:=y;
l3:=1-x-y;
bas3[1]:=9/2*(l3-2/3)*(l3-1/3)*l3;
bas3[2]:=27/2*l1*l3*(l3-1/3);
bas3[3]:=27/2*l1*(l1-1/3)*l3;
bas3[4]:=9/2*l1*(l1-1/3)*(l1-2/3);
bas3[5]:=27/2*l2*l3*(l3-1/3);
bas3[6]:=27*l1*l3*l2;
bas3[7]:=27/2*l1*l2*(l1-1/3);
bas3[8]:=27/2*l2*(l2-1/3)*l3;
bas3[9]:=27/2*l1*l2*(l2-1/3);
bas3[10]:=9/2*l2*(l2-1/3)*(l2-2/3);
bas4[1]:=32/3*l3*(l3-1/4)*(l3-1/2)*(l3-3/4);
bas4[2]:=128/3*l1*l3*(l3-1/4)*(l3-1/2);
bas4[3]:=64*l3*l1*(l1-1/4)*(l3-1/4);
bas4[4]:=128/3*l1*l3*(l1-1/4)*(l1-1/2);
bas4[5]:=32/3*l1*(l1-1/4)*(l1-1/2)*(l1-3/4);
bas4[6]:=128/3*l2*l3*(l3-1/2)*(l3-1/4);
bas4[7]:=128*l1*l2*l3*(l3-1/4);
bas4[8]:=128*l1*l2*l3*(l1-1/4);
bas4[9]:=128/3*l1*l2*(l1-1/4)*(l1-1/2);
bas4[10]:=64*l2*l3*(l2-1/4)*(l3-1/4);
bas4[11]:=128*l1*l2*l3*(l2-1/4);
bas4[12]:=64*l1*l2*(l1-1/4)*(l2-1/4);
bas4[13]:=128/3*l2*(l2-1/4)*(l2-1/2)*l3;
```

```
bas4[14]:=128/3*l1*l2*(l2-1/2)*(l2-1/4);
bas4[15]:=32/3*l2*(l2-1/4)*(l2-1/2)*(l2-3/4);
with(linalg);
for i from 1 to 15 do
        grad1:=grad(bas4[i],[x,y]);
        for j from 1 to 10 do
               u1:=expand(grad1[1]*bas3[j]);
               u2:=expand(grad1[2]*bas3[j]);
               coeft[i,j,1]:=int(int(u1,y=0..1-x),x=0..1);
               coeft[i,j,2]:=int(int(u2,y=0..1-x),x=0..1);
od:
od;
fortran (coeft, filename=nfort43.f);
```

APPENDIX B

MAPLE CODES FOR THE COMPUTATION OF SOME INTEGRALS OF HIGH ORDER SPECTRAL ELEMENT METHODS ON THE REFERENCE RECTANGLE

```
bas5:=array(1..36); % base functions of 5th-order spectral elements in
% 2D
bas6:=array(1..49); % base functions of 6th-order spectral elements in
% 2D
phi5:=array(1..6): % base functions of 5th-order spectral elements in
% 1D
phi6:=array(1..7): % base functions of 6th-order spectral elements in
```

```
% 1D
y5:=array(1..6): % Gauss-Lobetto grid points of 5th-order polynomial
% for interval [-1,1]
y6:=array(1..7): % Gauss-Lobetto grid points of 6th-order polynomial
% for interval [-1.1]
x5:=array(1..6);
x6:=array(1..7):
y5(1)=-1.0000000000000
y5(2)=-0.76505532392946
y5(3)=-0.28523151648065
y5(4)=0.28523151648065
y5(5)=0.76505532392946
y5(6)=1.0000000000000
y6(1)=-1.0000000000000
y6(2)=-0.83022389627857
y6(3)=-0.46884879347071
y6(4)=0.
y6(5)=0.46884879347071
y6(6)=0.83022389627857
y6(7)=1.0000000000000
% relations of base functions in 2D and 1D:
% for i from 1 to 6 do
% for j from 1 to 6 do
%       k=i+(j-1)*5;
%       bas5[k]:=phi5[i]*phi5[j]:
% od;
% od;
% similarly for the base functions of 6th-order elements
% Locate the Gauss-Lobetto points for interval [0.1]
for i from 1 to 6 do
     x5[i]:=0.5*(1+y5[i]);
```

```
od;
for j from 1 to 7 do
        x6[i]:=0.5*(1+y6[i]);
od;
% the following are base functions in 1D
f5:=(x-x5[1])*(x-x5[2])*(x-x5[3])*(x-x5[4])*(x-x5[5])*(x-x5[6]);
for i from 1 to 6 do
        phi5[i]:=f5/(x-x5[i]);
        x:=x5[i];
        e:=phi5[i];
        x:="x";
        phi5[i]:=phi5[i]/e;
od;
f6:=(x-x6[1])*(x-x6[2])*(x-x6[3])*(x-x6[4])*(x-x6[5])*(x-x6[6])* (x-x6[7]);
for i from 1 to 7 do
        phi6[i]:=f6/(x-x6[i]);
        x:=x6[i];
        e:=phi6[i];
        x:="x";
        phi6[i]:=phi6[i]/e;
od;
with(linalg);
```

compute the coefficients coef(1..36,1..36,1..2) for the stiffness matrix for 5th-order elements, similarly for 6th-order elements

```
for i from 1 to 6 do
for j from 1 to 6 do
        u:=phi5[i]*phi5[j];
        v:=diff(phi5[i])*diff(phi5[j]);
        c[i,j]:=int(u,x=0..1);
        d[i,j]:=int(v,x=0..1);
od;
```

```
od;
for i1 from 1 to 6 do
for j1 from 1 to 6 do
for i2 from 1 to 6 do
for j2 from 1 to 6 do
        i:=i1+(j1-1)*6;
        j:=i2+(j2-1)*6;
        coef[i,j,1]:=d[i1,i2]*c[j1,j2]:
        coef[i,j,2]:=d[j1,j2]*c[i1,i2]:
od:
od:
od:
od:
```

compute the coefficients for mass matrix of 5th-order elements, similarly for 6th-order elements

```
for i1 from 1 to 6 do
for j1 from 1 to 6 do
for i2 from 1 to 6 do
for j2 from 1 to 6 do
        i:=i1+(j1-1)*6;
        j:=i2+(j2-1)*6;
        coef[i,j]:=c[i1,i2]*c[j1,j2]:
od:
od;
od;
od:
```

compute the coefficients for the right hand side for the evaluation of the gradient of the gauge variable

```
% Define f5d6[7,6], f65[7,6] as
for i from 1 to 7 do
for j from 1 to 6 do
```

```
        u:=diff(phi6[i])*phi5[j];
        v:=phi6[i]*phi5[j];
        f5d6[i,j]:=int(u, x=0..1);
        f65[i,j]:=int(v, x=0..1);
od;
od;
for i1 from 1 to 7 do
for j1 from 1 to 7 do
for i2 from 1 to 6 do
for j2 from 1 to 6 do
        i:=i1+(j1-1)*7;
        j:=i2+(j2-1)*6;
        coef[i,j,1]:=f5d6[i1,i2]*f65[j1,j2];
        coef[i,j,2]:=f5d6[j1,j2]*f65[i1,i2];
od;
od;
od;
od;
```

compute the coefficients for the nonlinear term of 5th-order elements

define two arrays f55d5(6,6,6), f555(6,6,6)

```
for i from 1 to 6 do
for j from 1 to 6 do
for k from 1 to 6 do
        u:=phi5[i]*phi5[j]*diff(phi5[k]);
        v:=phi5[i]*phi5[j]*phi5[k];
        f55d5[i,j,k]:=int(u,x=0..1);
        f555[i,j,k]:=int(v,x=0..1);
od;
od;
od;
```

```
% form the coefficients
for i1 from 1 to 6 do
for j1 from 1 to 6 do
for i2 from 1 to 6 do
for j2 from 1 to 6 do
for i3 from 1 to 6 do
for j3 from 1 to 6 do
      i=i1+(j1-1)*6
      j=i2+(j2-1)*6
      k=i3+(j3-1)*6
      coef[i,j,k,1]:=f55d5[i1,i2,i3]*f555[j1,j2,j3];
      coef[i,j,k,2]:=f55d5[j1,j2,j3]*f555[i1,i2,i3];
od;
od;
od;
od;
od;
od;
```

Remark: Since the size of the coef[1..36,1..36,1..36,1..2] is very large, we can only form the arrays "f555" and "f55d5", then in the FORTRAN or C programs form the array "coef".