A TWO-LEVEL DISCRETIZATION METHOD FOR THE 
STREAMFUNCTION FORM OF THE NAVIER-STOKES 
EQUATIONS

by

Faisal Abdulkareem Fairag

BS, King Fahd University of Petroleum and Minerals, 1987
MS, King Fahd University of Petroleum and Minerals, 1989
MA, University of Pittsburgh, 1994

Submitted to the Graduate Faculty of
Arts and Sciences in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

University of Pittsburgh

1998
This dissertation was presented
by
Faisal Abdulkareem Fairag

It was defended on
February 5, 1998

and approved by

__________________________
Charles A. Hall, Ph.D.
Professor of Mathematics

__________________________
D. J. Hebert, Ph.D.
Professor of Mathematics

__________________________
Patrick Smolinski, Ph.D.
Professor of Mechanical Engineering

__________________________
William J. Layton, Ph.D.
Professor of Mathematics
Committee chairperson
A TWO-LEVEL DISCRETIZATION METHOD FOR THE STREAMFUNCTION FORMULATION OF THE NAVIER-STOKES EQUATIONS

Faisal Abdul-Karim Fairag, PhD

ABSTRACT: We analyze a two-level finite element method for the streamfunction formulation of the Navier-Stokes equations. This report presents the two-level algorithm and a priori error analysis for the case of conforming elements. The streamfunction formulation in two dimensions has the great advantages that one solves for only a single scalar variable rather than a coupled system. Further, the incompressible constraint is automatically satisfied so there are no compatibility conditions between velocity and pressure spaces. The disadvantage is that the linear system, though small, arises from a fourth order problem so it can be very ill conditioned. The nonlinear system is also, at higher Reynolds numbers, very sensitive to small perturbations. The two-level algorithm consists of solving a small nonlinear system on the coarse mesh, then solving a large linear system on the fine mesh. The basic results states that the error between the coarse and fine meshes are related superlinearly.

FORTRAN programs for this algorithm and a complete discussion of these programs are included. These programs are used to solve the Navier-Stokes equations with a known solution in a rectangular domain for a range of Reynolds numbers to compare one level vs. two level methods in terms of computer time. Also, we solve the driven flow in a square cavity. These flows have been widely used as test cases for validating incompressible fluid dynamics algorithms. Streamfunction contours are displayed showing the main features of the flow.
A posteriori error estimator for the two-level algorithm is derived which can be used as an indicator for an assessment of the reliability of the results.
To my parents
Acknowledgments

Praise be to Allah, Lord of the worlds, the Almighty, with whose gracious help it was possible to accomplish this task. I wish to express my sincere appreciation and deep gratitude to my dissertation advisor, Professor William Layton for his generous support, encouragement and suggesting the topic of this dissertation. I thank him for many stimulating discussions and introducing me to a field where he has made so many significant contributions. I am thankful to my committee members, to Professor D. J. Hebert for reading the manuscript in draft form and pointing out to me various misprints and suggestions. I thank Professor C. Hall and P. Smolinski for suggesting to add a remark in Section(5.4) and a paragraph about the reduced basis method in the introduction. This work was supported by King Fahd University of Petroleum and Minerals under sponsorship 152-SD.

I am thankful to Professor L. Tobiska for reading and commenting on chapter 2. I thank Professor J. E. Akin, Rice University, Houston, Texas for permission to reproduce Figure(5.9) taken from [2] and to Professor U. Ghia for permission to reproduce Figure(5.8) taken from [31]. I am truly thankful to Professor F. Schieweck, for generously providing the postscript file of Figure(5.4) taken from [63]. I thank Professor J. Xu for referring me to many references related to Chapter 2. I wish to acknowledge the fruitful discussions by Professor J. Maubach and V. Ervin on some computational aspects of this dissertation. I thank Professor V. John for providing me some references related to chapter 3.

It is my great pleasure to thank my wife, Wejdan Iskander, for spending endless days and nights typesetting the dissertation and for long-suffering, patience and understanding which made it possible. It is a great privilege and pleasure to thank my
parents, Fatemah Fairag and Abdul-karim Fairag, to whom this work is dedicated. From childhood and on, I have received their blessings, long-support and encouragements to continue my education.
## Contents

1 Introduction  
   1.1 Motivations ..................................................... 2  
   1.2 Navier-Stokes Equations ....................................... 4  
   1.3 Streamfunction Equation ....................................... 6  
   1.4 Two-Level Idea ................................................ 8  
   1.5 A Model Problem and Chapters Description ................... 10

2 A’ Priori Error Analysis  
   2.1 Introduction ................................................... 15  
   2.2 Notations and Preliminaries ................................... 16  
   2.3 Two Weak Formulations ....................................... 18  
   2.4 Two Level Method .............................................. 22  
   2.5 The Error Bound ............................................... 27  
   2.6 Summary ....................................................... 36

3 A Posteriori Error Estimator  
   3.1 Introduction ................................................... 38  
   3.2 Notations and Preliminaries ................................... 39  
   3.3 Error Analysis ................................................ 44
## 4 Implementation of the Two-Level FEM

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Notations and Arrays</td>
<td>57</td>
</tr>
<tr>
<td>4.3</td>
<td>Mesh Generation</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>Affine Mapping</td>
<td>67</td>
</tr>
<tr>
<td>4.5</td>
<td>Basis Functions for Morley Element</td>
<td>71</td>
</tr>
<tr>
<td>4.6</td>
<td>Basis Function for Bogner-Fox-Schmit Element</td>
<td>76</td>
</tr>
<tr>
<td>4.7</td>
<td>Discretizing the Continuous Problem</td>
<td>89</td>
</tr>
<tr>
<td>4.8</td>
<td>The Solution of ( F(C) = 0 )</td>
<td>90</td>
</tr>
<tr>
<td>4.9</td>
<td>Computation of the Jacobian</td>
<td>92</td>
</tr>
<tr>
<td>4.10</td>
<td>Calculation of the Load Vector</td>
<td>109</td>
</tr>
<tr>
<td>4.11</td>
<td>Storage of the Jacobian</td>
<td>111</td>
</tr>
<tr>
<td>4.12</td>
<td>Boundary Conditions Contribution</td>
<td>114</td>
</tr>
<tr>
<td>4.13</td>
<td>Solving the Linear System</td>
<td>115</td>
</tr>
<tr>
<td>4.14</td>
<td>COARSE-LEVEL-SOLVE Subroutine</td>
<td>119</td>
</tr>
<tr>
<td>4.15</td>
<td>FINE-LEVEL-SOLVE Subroutine</td>
<td>119</td>
</tr>
</tbody>
</table>

## 5 Numerical Tests

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>127</td>
</tr>
<tr>
<td>5.2</td>
<td>Accuracy Two Level Method vs. One Level Method</td>
<td>127</td>
</tr>
<tr>
<td>5.3</td>
<td>Robustness: Test Problem for Different Reynolds Numbers</td>
<td>133</td>
</tr>
<tr>
<td>5.4</td>
<td>Meshes Scaling Test</td>
<td>139</td>
</tr>
<tr>
<td>5.5</td>
<td>The Driven Cavity Problem</td>
<td>143</td>
</tr>
</tbody>
</table>

## A FORTRAN 90 Program Using Morley Element

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>151</td>
</tr>
</tbody>
</table>

## B FORTRAN 77 Program Using Bogner-Fox-Schmit Element

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>182</td>
</tr>
</tbody>
</table>
C MATLAB and M-files 219

D Data Files 238

6 Bibliography 262
List of Tables

2.1 Scaling of two level finite elements ........................................ 26

5.1 One level method. ..................................................................... 130
5.2 Two level method. ................................................................. 130
5.4 Values for streamline contours in Figure (5.3). ....................... 134
5.3 Two level for test problem. ..................................................... 135
5.5 $L^2$-error, $H^1$-error and $H^2$-error of $\psi$ for different pairs $(H,h)$. 141
5.6 Results for $u$-velocity along the vertical line $x = 0.5$ ............. 145
5.7 Results for $v$-velocity along the horizontal line $y = 0.5$ ........ 145
5.8 Two level for cavity flow .......................................................... 146
List of Figures

2.1 The Argyris triangular element ............................... 24
2.2 The Clough-Tocher triangular element .......................... 25
2.3 The Bogner-Fox-Schmit rectangular element .................... 25

3.1 The Clough-Tocher triangle element ............................ 44
3.2 Bogner-Fox-Schmit rectangular element ......................... 46

4.1 Triangles labeling .............................................. 61
4.2 Global node labeling ............................................ 62
4.3 Local node ordering ............................................. 62
4.4 Rectangles labeling ............................................. 66
4.5 Global node ordering ............................................ 67
4.6 Local node ordering ............................................. 68
4.7 The reference Triangle ......................................... 68
4.8 The reference rectangle ......................................... 70
4.9 Local basis functions for Morley element ....................... 75
4.10 Local basis functions for Bogner-Fox-Schmit elements \( \tilde{\phi}_1 \cdots \tilde{\phi}_6 \) .................................. 86
4.11 Local basis functions for Bogner-Fox-Schmit elements \( \tilde{\phi}_7 \cdots \tilde{\phi}_{12} \) ................................ 87
4.12 Local basis functions for Bogner-Fox-Schmit elements \( \tilde{\phi}_{13} \cdots \tilde{\phi}_{16} \) ............................. 88
4.13 \( e_t \) Triangle ................................................. 93
4.14 Nonzero element location for \( lx = 2, 3, 4, 5 \) .......... 113
4.15 Bandwidth of the Jacobian matrix .......................... 116
4.16 Triangle g ....................................................... 122
4.17 Nonzero entries for \((H, h) = (\frac{1}{2}, \frac{1}{3}), (\frac{1}{3}, \frac{1}{6})\) .... 125

5.1 Streamlines for \( h = \frac{1}{8}, \frac{1}{16}, \frac{1}{16} \) with Re=10 and the corresponding 3-D graphs using one level method ............. 131
5.2 Streamlines for \((H, h) = (\frac{1}{4}, \frac{1}{8}), (\frac{1}{7}, \frac{1}{14}), (\frac{1}{5}, \frac{1}{10})\) with Re=10 and the corresponding 3-D graphs using two level method ............. 132
5.3 Streamlines for \( H = \frac{1}{16}, h = \frac{1}{32} \) using Bogner-Fox-Schmit element. ... 136
5.4 Streamlines for \( h = \frac{1}{64} \) (courtesy F. Schieweck [63]). ........... 137
5.5 Streamlines for \( H = \frac{1}{16}, h = \frac{1}{32} \) using Morley element. ............ 138
5.6 Streamlines for \((H, h) = (\frac{1}{4}, \frac{1}{8}), (\frac{1}{4}, \frac{1}{8}), (\frac{1}{4}, \frac{1}{16})\) and \((\frac{1}{4}, \frac{1}{25})\). .......... 142
5.7 Streamlines for \( H = \frac{1}{16}, h = \frac{1}{32} \) with different values of Re numbers using two level method ............. 147
5.8 Ghia-Ghia-Shin’s streamlines, 129 \times 129, (courtesy U. Ghia [31]). 148
5.9 (a),(c) Streamlines for cavity flow using two level method, (b),(d)
Akin’s streamlines, a mesh of 40 \times 40 elements, (courtesy J. Akin [2]). 149
5.10 (above) u-velocity lines through the vertical line \( x = 0.5 \), (below) v-
velocity lines through the horizontal line \( y = 0.5 \) .................. 150
Chapter 1

Introduction
1.1 Motivations

The Navier-Stokes Equations govern the motion of many fluids like water, air and oil under some conditions. They appear in the study of many important phenomena. Sometimes they appear alone to describe the nature of the problem and sometimes they appear coupled with other equations. The Navier-Stokes Equations are used in many engineering studies. For instance, they are used in theoretical studies in aeronautical sciences, in meteorology, in thermo-hydraulics, in the petroleum industry, in physics, etc.

The exact solution of the Navier-Stokes equations is totally out of reach. A very small number of exact solutions of these equations are known. We mentioned in the beginning that the Navier-Stokes equations play an important role in several scientific and engineering fields. The needed information by engineers can be provided only through numerical computations. In this study, the computational techniques proposed are finite element methods, the most widely used technique for engineering design and analysis. Here the problem is difficult and the numerical solution of the Navier-Stokes equations will require simultaneous effort of mathematicians, numerical analysts and specialists in scientific computing.

A variety of mathematical formulations have been tested in solving the Navier-Stokes equations numerically. They are velocity-pressure models, streamfunction-vorticity model, and streamfunction model. The first model has been extensively developed, the second has received moderate attention, while the third model has received much less attention. The mathematical model studied in this thesis is the streamfunction model. The attraction of this model is that the incompressibility constraint is automatically satisfied, and there is only one unknown to solve for.

The streamfunction formulation in two dimensions has the great advantages that
one solves for only a single scalar variable rather than a coupled system. Further, the incompressible constraint is automatically satisfied so there are no compatibility conditions between velocity and pressure spaces. The disadvantage is that the linear system, though small, arises from a fourth order problem so it can be very ill posed. The nonlinear system is also, at higher Reynolds numbers, very sensitive to small perturbations. Thus global convergence of Newton’s method does not hold. In fact, in our tests we found that normal Newton iterations did not converge to the solution of the nonlinear algebraic system for even moderate Reynolds numbers. Further, at moderate to high Reynolds numbers the linearized system arising from the full Newton step is both nonsymmetric and indefinite. We experienced severe difficulties, as well, in the convergence of generalized conjugate gradient iterations approximating the solution of the linear problems. These difficulties arose principally when we were solving the system arising from the usual finite element method to compare it to the solution of the two level discretization method. In fact, none of the difficulties mentioned above occur when the two level that we propose and study is used to discretize the streamfunction formulation of the Navier-Stokes equations: the particular update is designed specifically to circumvent these difficulties associated with higher Reynolds numbers. The method begins by calculating a coarse mesh approximate solution which is interpolated to a fine mesh. Taking a partial linearization about this coarse mesh approximate solution, and beginning with it as an initial guess, an update is calculated on the fine mesh by using a partial linearization of the nonlinear system. The simplest partial linearization is to delete all the nonlinear terms. This Stokes approximation is clearly hopeless at moderate to high Reynolds numbers. We selected a partial linearization based upon the associated Oseen problem. The linear system arising from this problem on the fine mesh, while not symmetric, does have positive definite symmetric part. It is therefore not surprising that we experienced no
difficulty whatsoever in solving the associated linear system by generalized constant gradient methods. The main thrust of this thesis is to analyze this procedure: What relationships between the coarse and the fine mesh ensure that the accuracy will be the same as if the full fine-mesh non linear system were solved to truncation error? How does one estimate the computational error in this procedure and designed adaptive meshes associated with it? Is the solution quality comparable to that obtained by solving the full nonlinear system? In practical tests what efficiency advantages are obtained with the new method?

In summary, the purpose of this thesis is to present and study the à priori and a posteriori error analysis of a new two level finite element method for discretizing the streamfunction formulation of the Navier-Stokes equations. Moreover, the resulting methods will be tested on some practical problems.

1.2 Navier-Stokes Equations

Galdi [29, 30] said: ”The Navier-Stokes Equations have been written more than one hundred seventy years ago. In fact, they were proposed in 1822 by the French engineer C.M.L.H. Navier upon the basis of a suitable molecular model. It is interesting to observe. However, the law of interaction between the molecules postulated by Navier were shortly recognized to be totally inconsistent from the physical point of view for several materials and, in particular, for liquids. It was only more than twenty years later that the same equations were rederived by G.H. Stokes (1845) in a quite general way by means of the theory of continua.”

While the physical model leading to the Navier-Stokes equations is simple, the situation is different from the mathematical point of view. Actually, the exact solution of the Navier-Stokes equations is totally out of reach. A very small number of exact
solutions of these equations are known. The Navier-Stokes equations can be written as follows

$$\frac{1}{Re} \Delta u + u \cdot \nabla u + \nabla p = f \quad \text{in} \quad \Omega,$$

$$\text{div} \ u = 0 \quad \text{in} \quad \Omega,$$

$$u = 0 \quad \text{on} \quad \partial \Omega,$$

where \( u = u(x, y) \) is the velocity field evaluated at point \( x \in \Omega \), \( Re \) is the Reynolds number, \( p \) is the pressure field, and \( \Omega \) is a bounded simply connected domain in \( \mathbb{R}^2 \).

Many researchers believe that the difficulties in getting the exact solution is because of the nonlinear term \( u \cdot \nabla u \). Galdi [29, 30] presented an explanation of the difficulties of the analytical solution of the Navier-Stokes equations. The following paragraph summarizes his explanation.

First, the Navier-Stokes equations do not satisfy a specific property. He calls this property a symmetry in \( u \) and \( p \). Because of the lack of symmetry in \( u \) and \( p \), the Navier-Stokes equations do not fall in any of the classical categories of equations. The reason that the Navier-Stokes equations are so difficult is the coupled effect of the absence of symmetry and of the presence of the nonlinear term. Three types of problems appear in the mathematical and analytical treatments of these equations. These problems are existence, uniqueness, and regularity.

Many mathematicians attack the equations to study these problems and present them in papers and textbooks. One of the most accessible is An Introduction to the Navier-Stokes Initial Boundary Value Problem by Giovanni P. Galdi [28]. A valuable source of a systematic and up-to-date investigation of the fundamental properties of the Navier-Stokes equations is this series of Galdi [28] which is divided into four volumes.
1.3 Streamfunction Equation

We mentioned earlier that the Navier-Stokes equations play an important role in several scientific and engineering fields. The information needed by engineers can be provided only through numerical computations.

The mathematical model studied in this thesis is the streamfunction model. For 2D domains, the streamfunction equation can be written as follows

\[ Re^{-1} \Delta^2 \psi - \psi_y \Delta \psi_x + \psi_x \Delta \psi_y = \text{curl} \ f \quad \text{in} \quad \Omega, \]
\[ \psi = \frac{\partial \psi}{\partial \hat{n}} = 0 \quad \text{on} \quad \partial \Omega, \]

where \( \psi \) denotes the streamfunction, \( f \) is the given body force, \( Re \) is the given Reynolds number and \( \hat{n} \) represents the outward unit normal to \( \Omega \).

It is well known that for the velocity-pressure formulation of the Navier-Stokes problem the velocity and pressure approximation spaces need to be compatible to avoid instability. This condition, the Ladyzhenskaya-Babuska-Brezzi condition, arises from the study of stability and convergence properties of the velocity-pressure formulation of the Navier-Stokes problem. To satisfy this condition we cannot pick any two arbitrary discrete spaces for the velocity and pressure. For example, one idea of Arnold-Brezzi-Fortin [5] was to increase the number of degrees of freedom of the velocity element by adding bubble functions. This was done to reduce the space of pressure or to enrich the space of velocity elements.

Another still unexplored possibility is to explicitly weaken the condition \( \text{div} \ u = 0 \) by changing it to

\[ \text{div} \ u = g, \quad (\text{e.g.,} \quad g = h^2 \Delta p), \]

when \( g \) is a (well-chosen) small function. The first step in this direction has been
done in the work of Brezzi-Pitkranta [14]. One use of the streamfunction formulation avoids completely compatibility condition because the pressure is eliminated from momentum equation to yield a single nonlinear fourth order equation.

This idea of eliminating the pressure and the satisfaction of the continuity equation are also present in the Divergence Free Finite Element Method (DFFEM) (discussed, studied and analyzed in [38, 39]). The DFFEM treats the continuity equation as a constraint, thus the velocity is approximated not from the standard finite element vector space but from a discretely divergence free element subspace. In DFFEM approach, the pressures are eliminated from the calculations and the dimension of the system to be solved is reduced. For this technique to work, however, the underlying velocity-pressure approximation must still be stable.

A variety of numerical methods are used to solve the Navier-Stokes equations numerically using the streamfunction formulation. Finite difference approach for the streamfunction formulation was discussed in Goodrich-Gustafson-Halasi [34]. Cayco and Nicolaides [17, 16] gave a convergence analysis for small data (the case of global uniqueness) for this formulation of the Navier-Stokes equations. Some numerical example using the streamfunction formulation can be found in Betts-Haroutunian [12]. For a nonconforming finite element method, Baker and Jureidini [8] investigated the use of elements which are required only to be continuous, not continuously differentiable and not to satisfy the boundary conditions with a streamfunction formulation. Cayco and Nicolaides [16] presented and discussed a new weak form, which is suitable for analysis of nonconforming finite element approximations. Phillips and Malek [51] studied the Multidomain Collocation method for the streamfunction formulation of the Navier-Stokes equation.

The streamfunction formulation is a nonlinear forth order partial differential equation. The first example of a fourth order partial differential equation is the biharmonic
The biharmonic equation models a plate bending problem. The finite element method for the linear biharmonic equation was discussed and analyzed extensively in [18, 19]. The difficulty that fourth order problems pose from an algorithmic point of view is that conforming finite element methods require the use of continuously differentiable finite element functions. There are many elements which satisfy this requirement. For details concerning these elements and their approximation properties, one may consult Ciarlet [18, 19].

1.4 Two-Level Idea

The conforming finite element method of the streamfunction formulation requires the use of finite element functions that are continuously differentiable over Ω; this is a difficult constraint to satisfy in practice. Furthermore, the use of continuously differentiable finite elements leads to solving a large nonlinear system of equations. The study of nonconforming methods has received great attention from the desire to avoid the construction of finite element spaces and their bases that satisfy the $C^1$ continuity requirement. Many nonconforming elements which are applicable for biharmonic equation is described in [18, 19]. In fact, in our tests we found that nonconforming elements are much more sensitive for even moderate Reynolds numbers.

Another idea of avoiding the solution of a large nonlinear system of equations is a two level finite element method. The two level algorithm consists of solving a
small nonlinear system on the coarse mesh, then solving a linear system on the fine mesh. The computed solution from this procedure has the same quality if we solve a nonlinear system of equations in the fine mesh. The computational attraction of the methods is that they require the solution of a small system of nonlinear equations and one large linear system of equations. These types of methods were pioneered by Xu in [69, 70] for semilinear elliptic problems. The two level discretization methods have been analyzed for the Navier-Stokes equations by Layton in [41, 44, 42, 40].

Another idea of avoiding the solution of a large nonlinear system of equations is reduction method that reduces the degree of freedom. The reduced basis method is a reduction-type method that uses approximating functions that are closely related to the solution of the differential equation, see, e.g., [3, 54, 55, 4, 56, 27, 60, 58, 59, 50]. For example, two choices of reduced spaces are the Lagrange space and the Taylor space. In Navier-Stokes equations, the Lagrange space consists of solutions to the Navier-Stokes equations for different values of the Reynolds number. Hence, all basis vectors are generated by solving a large nonlinear system. The Taylor space consists of solution to the Navier-Stokes equations at a given Reynolds number, $R_e_0$. The remaining vectors are obtained by solving the equations that result by differentiating the Navier-Stokes equations with respect to the Reynolds number and evaluating at $R_e_0$. The computation of the first vector requires the solution of a large nonlinear system. For the other vectors, a large linear system is to be solved for each vector. Now, the reduced basis solution is a linear combination of three vectors. This result in a dense system of equations. This is due to the fact that the reduced basis vectors are global as opposed to the local basis vectors used in standard finite elements. Hence, the reduced basis method is feasible only if a very small number of basis vectors are needed. In the numerical solution of the Navier-Stokes equation, engineers need to know how globally close the computed solution is to the exact solution. This point
shows the need for an error estimator, which must be \textit{a posteriori} computed from the computed numerical solution and the given data of the problem. This estimator will give us an upper bound to the accuracy of the numerical approximation.

1.5 A Model Problem and Chapters Description

We first need to define some function spaces and associated norms. More details concerning these spaces can be found in [1]. Let $\Omega$ be a bounded simply connected polygonal domain in $\mathbb{R}^2$. $L^2(\Omega)$ is the Hilbert space of Lebesgue square integrable functions with norm $\| \cdot \|$ and $L^2_0(\Omega)$ is the subspace of $L^2(\Omega)$ consisting of functions with zero mean. Let $H^m(\Omega)$ be usual Sobolev space consisting of functions which together with their distributional derivatives up through order $m$ are in $L^2(\Omega)$. Denote the norm on $H^m(\Omega)$ by $\| \cdot \|_m$. Let $H^m_0(\Omega)$ be the completion of $C_0^\infty(\Omega)$ under the $\| \cdot \|_{-m}$ norm. We equip $H^m_0(\Omega)$ with the seminorm $| \cdot |_m$, which is a norm equivalent to $\| \cdot \|_m$. Also, the dual of space $H^m_0(\Omega)$ is denoted by $H^{-m}(\Omega)$, with norm $\| \cdot \|_{-m}$. Let $[H^m(\Omega)]^2$ be the space $H^m(\Omega) \times H^m(\Omega)$ and $[H^m_0(\Omega)]^2$ be the space $H^m_0(\Omega) \times H^m_0(\Omega)$ equipped with the following norm.

$\| u \|_m = (\| u_1 \|_m^2 + \| u_2 \|_m^2)^{1/2}$ and $\| u \|_m = (\| u_1 \|_m^2 + \| u_2 \|_m^2)^{1/2}$

where

$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$
For each $\phi \in H^1(\Omega)$, define
\[
curl \phi = \begin{pmatrix} \phi_y \\ -\phi_x \end{pmatrix}.
\]

For each $u \in [H^1(\Omega)]^2$, define
\[
curl u = \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y},
\]
where $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$.

Consider the Navier-Stokes equation,
\[
-\frac{1}{Re} \triangle u + u \cdot \nabla u + \nabla p = f \quad \text{in} \quad \Omega,
\]
(1.1)
together with the incompressibility constraint,
\[
\nabla \cdot u = 0 \quad \text{in} \quad \Omega,
\]
and the homogeneous no-slip boundary condition,
\[
u = 0 \quad \text{on} \quad \Gamma,
\]
where $u$ denotes the velocity field, $p$ the pressure, $f$ the given body force per unit mass, and $Re$ the given Reynolds number.

Any divergence-free velocity vector $u \in [H^1_0(\Omega)]^2$ has a unique streamfunction ([33], Theorem 3, page 22) $\psi \in H^2_0(\Omega)$, defined by
\[
curl \psi = u.
\]
(1.2)
One may eliminate the pressure from (1.1) by taking $\text{curl}$ of both sides of equation (1.1) and using equation (1.2) to obtain the streamfunction equation, namely,

$$Re^{-1} \Delta^2 \psi - \psi_y \Delta \psi_x + \psi_x \Delta \psi_y = \text{curl } f \quad \text{in } \Omega, \quad (1.3)$$

$$\psi = 0 \quad \text{on } \partial \Omega_1, \quad (1.4)$$

$$\frac{\partial \psi}{\partial \hat{n}} = 0 \quad \text{on } \partial \Omega, \quad (1.5)$$

where $\hat{n}$ represents the outward until normal to $\Omega$. A one weak form of equation (1.3) is

Find $\psi \in H^2_0(\Omega)$ such that, for all $\phi \in H^2_0(\Omega)$,

$$a(\psi, \phi) + b(\psi, \psi, \phi) = l(\phi), \quad (1.6)$$

where

$$a(\psi, \phi) = Re^{-1} \int_{\Omega} \psi_{xx} \phi_{xx} + 2\psi_{xy} \phi_{xy} + \psi_{yy} \phi_{yy} dxdy,$$

$$b(\xi, \psi, \phi) = \int_{\Omega} \left\{ (\xi_y \psi_{xy} - \xi_x \psi_{yy}) \phi_y - (\xi_x \psi_{xy} - \xi_y \psi_{xx}) \phi_x \right\} dxdy,$$

$$l(\phi) = \int_{\Omega} (f \cdot \text{curl } \phi) dxdy.$$

In this thesis I shall present à priori and à posteriori error analysis of the two level method of the formulation described in equation (1.6), including some computational tests.

I have divided the subject into five chapters. Each chapter is self-contained and basically is not dependent on the other one, except in the case noted below.

Chapter 2 discuss an algorithm and à priori error analysis of a two level method of (1.6) including some existence and uniqueness aspects. A posteriori error analysis is
presented in chapter 3. Chapter 4 focuses on the implementation of the finite element method for the algorithm which was described in chapter 2. In chapter 5 we test the code described in chapter 4 for a test problem with a known solution and a driven cavity problem.
Chapter 2

A’ Priori Error Analysis
2.1 Introduction

Convergence analysis for finite element approximation of the primitive variable formulation of the Navier-Stokes equations has been extensively developed in the last 20 years, see for example [36, 33, 32, 62]. The analogous theory for the streamfunction formulation for the Navier-Stokes equations has received much less attention. The attractions of the streamfunction formulation are that the incompressibility constraint is automatically satisfied, the pressure is not present in the weak form and there is only one scalar unknown to solve for. The standard weak formulation of the streamfunction version first appeared in 1979 in [33]. In this direction, Cayco and Nicolaides [17, 16] studied a general analysis of convergence for this standard weak formulation of the Navier-Stokes equations. The standard weak form is unsuitable for derivation or analysis of nonconforming finite element approximations. For a nonconforming finite element method, Baker and Jureidini [8] investigated the use of elements which are required only to be continuous and not to satisfy the boundary conditions with a nonstandard weak formulation. Their weak formulation extends the standard one by including appropriate integrals on inter-element boundaries and on the boundary of the problem domain. Cayco and Nicolaides [16] presented and discussed a new weak form, which is suitable for analysis of nonconforming finite element approximations. They discussed this weak form and applied it to three specific nonconforming finite element schemes.

The discretization of the streamfunction formulation still leads to a problem of solving a large and ill-conditioned nonlinear systems of algebraic equations. two level finite element discretization are presently a very promising approach for approximating the Navier-Stokes equations, see [41]. The computational attraction of the methods are that they require the solution of only a small system of nonlinear equa-
tions on coarse mesh and one linear system of equations on fine mesh. These types of methods were pioneered by Xu in [69, 70] for semilinear elliptic problems. The two level discretization methods have been recently analyzed for the Navier-Stokes equations in [41, 43, 44] and for the streamfunction formulation of the Navier-Stokes equations in [72]. The methods studied in [72] involve solving a full linearization of the streamfunction equation on the fine mesh. The purpose of this thesis is to present and analyze a two level conforming finite element method for discretizing the streamfunction formulation of the Navier-Stokes equations, which requires the solution of a partial linearization of the streamfunction equation on the fine mesh.

2.2 Notations and Preliminaries

We first need to define some function spaces and associated norms. More detailed concerning these spaces can be found in [1]. Let $\Omega$ be a bounded simply connected polygonal domain in $\mathbb{R}^2$. $L^2(\Omega)$ is the Hilbert space of Lebesgue square integrable functions with norm $\| \cdot \|_0$ and $L^2_0(\Omega)$ is the subspace of $L^2(\Omega)$ consisting of functions with zero mean. Let $H^m(\Omega)$ be the usual Sobolev space consisting of functions which together with their distributional derivatives up through order $m$ are in $L^2(\Omega)$. Denote the norm on $H^m(\Omega)$ by $\| \cdot \|_m$. Let $H^m_0(\Omega)$ be the completion of $C^\infty_0(\Omega)$ under the $\| \cdot \|_m$ norm. We equip $H^m_0(\Omega)$ with the seminorm $| \cdot |_m$, which is a norm equivalent to $\| \cdot \|_m$. Also, the dual of space $H^m_0(\Omega)$ is denoted by $H^{-m}(\Omega)$, with norm $\| \cdot \|_{-m}$. Let $[H^m(\Omega)]^2$ be the space $H^m(\Omega) \times H^m(\Omega)$ and $[H^m_0(\Omega)]^2$ be the space $H^m_0(\Omega) \times H^m_0(\Omega)$.
equipped with the following norm
\[ \| u \|_m = (\| u_1 \|_m^2 + \| u_2 \|_m^2)^{1/2} \] and
\[ | u |_m = (| u_1 |_m^2 + | u_2 |_m^2)^{1/2} \] where \( u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \).

For each \( \phi \in H^1(\Omega) \), define
\[ \text{curl } \phi = \begin{pmatrix} \phi_y \\ -\phi_x \end{pmatrix}. \]

For each \( u \in [H^1(\Omega)]^2 \), define
\[ \text{curl } u = \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y}, \] where \( u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \).

Consider the Navier-Stokes equations describing the flow of an incompressible fluid:
\[
\begin{align*}
-\text{Re}^{-1} \nabla^2 u + (u \cdot \nabla)u + \nabla p &= f, \text{ in } \Omega, \\
\nabla \cdot u &= 0, \text{ in } \Omega, \\
u &= 0, \text{ on } \partial \Omega, \\
\int_{\Omega} p \, d\Omega &= 0.
\end{align*}
\] (2.1)

Later, we will state conditions on \( f \) and \( \text{Re}^{-1} \) guaranteeing the solution to (2.1).

Any divergence-free velocity vector \( u \in [H^1_0(\Omega)]^2 \) has a unique stream function [33, Theorem 3.1, page 22] \( \psi \in H^2_0(\Omega) \), defined by

\[ \text{curl } \psi = u. \]
Moreover, the streamfunction $\psi$ satisfies

$$Re^{-1} \Delta^2 \psi - \psi_y \Delta \psi_x + \psi_x \Delta \psi_y = \text{curl} \ f, \text{ in } \Omega,$$

$$\psi = 0, \text{ on } \partial \Omega,$$

$$\frac{\partial \psi}{\partial n} = 0, \text{ on } \partial \Omega,$$

where $\hat{n}$ represents the outward unit normal to $\Omega$.

### 2.3 Two Weak Formulations

The standard weak form of equation (2.1) is

Find $u \in [H^1_0]^2, p \in L^2_0(\Omega)$, such that $\forall w \in [H^1_0(\Omega)]^2, q \in L^2_0(\Omega)$,

$$Re^{-1} \tilde{a}(u, w) + \tilde{b}(u; u, w) + \tilde{c}(w, p) = < f, w >,$$

$$\tilde{c}(u, q) = 0,$$

(2.4)

where

$$\tilde{a}(u, w) = \int_{\Omega} \nabla u : \nabla w,$$

$$\tilde{b}(u; v, w) = \int_{\Omega} ((u \cdot \nabla)v) \cdot w,$$

(2.5)

$$\tilde{c}(w, q) = \int_{\Omega} q \ \text{div} \ w,$$

and $< \cdot, \cdot >$ denotes the duality pairing in $L^2(\Omega)$. The standard weak form of equation (2.3) is:

Find $\psi \in H^2_0(\Omega)$ such that, for all $\phi \in H^2_0(\Omega)$,

$$a(\psi, \phi) + b(\psi; \psi, \phi) = l(\phi),$$

(2.6)
where

\[ a(\psi, \phi) = Re^{-1} \int_{\Omega} \Delta \psi \cdot \Delta \phi, \]

\[ b(\xi; \psi, \phi) = \int_{\Omega} \Delta \xi(\psi_y \phi_x - \psi_x \phi_y), \]

\[ l(\phi) = (f, \text{curl} \ \phi) = \int_{\Omega} f \cdot \text{curl} \ \phi. \]

(2.7)

Another equivalent formulation of equation (2.3), introduced by Cayco and Nicolaides [16], is:

Find \( \psi \in H^2_0(\Omega) \) such that, for all \( \phi \in H^2_0(\Omega) \),

\[ a_0(\psi, \phi) + b_0(\psi; \psi, \phi) = l(\phi), \]

(2.8)

where

\[ a_0(\psi, \phi) = Re^{-1} \int_{\Omega} \psi_{xx} \phi_{xx} + 2\psi_{xy} \phi_{xy} + \psi_{yy} \phi_{yy}, \]

\[ b_0(\xi; \psi, \phi) = \int_{\Omega} (\xi_y \psi_{xy} - \xi_x \psi_{yy}) \phi_y - (\xi_x \psi_{xy} - \xi_y \psi_{xx}) \phi_x, \]

(2.9)

\[ l(\phi) = (f, \text{curl} \ \phi) = \int_{\Omega} f \cdot \text{curl} \ \phi. \]

Conforming element can be used with either (2.6) or (2.8), in this case the two weak formulations produce identical results because \( a(\psi, \phi) = a_0(\psi, \phi) \) and \( b(\xi; \psi, \phi) = b_0(\xi; \psi, \phi) \) for all \( \psi, \phi, \xi \in H^2_0(\Omega) \). However, when using nonconforming approximation subspaces, (2.8) and (2.6) generate different finite element methods. Nonconforming elements should be used only with (2.8). To illustrate the reason, suppose we solve the Stokes problem with the nonconforming Morley triangle, i.e. the quadratic element whose degrees of freedom are function values at the vertices and normal derivatives at the midsides. Boundary conditions are imposed by setting.
all the degrees of freedom at the boundary to be zero. Observe that a necessary and sufficient condition for the existence of a unique solution to the discrete biharmonic equation is that the bilinear induces a norm on the trial space. This is not the case for the Morley space \[17\]. The following theorem states that the forms (2.4) and (2.6) are equivalent in the sense of having identical solutions. The reason for this is that the space of \textit{curls} of \(H^2_0(\Omega)\) functions coincides with the space of divergence-free functions in \([H^1_0(\Omega)]^2\).

The following theorem states that the problems (2.1) and (2.3) are equivalent in the sense of having identical solutions.

\textbf{Theorem 2.3.1 ([33]Theorem 2.6, page 120)}

Problems (2.4) and (2.6) are equivalent in the sense that if \((u, p)\) is a solution of (2.4) then the stream-function \(\psi\) of \(u\) satisfies (2.6); conversely if \(\psi\) is a solution of (2.6), then there exist exactly one element \(p\) of \(L^2_0(\Omega)\) such that the pair \((u = \text{curl } \psi, p)\) satisfies (2.4).

The following lemma states some basic bound for the bilinear \(a\), the trilinear \(b\) and the functional \(l\).

\textbf{Lemma 2.3.1}

\textit{Given} \(\psi, \xi, \phi \in H^2_0(\Omega)\) and \(f \in [L^2(\Omega)]^2\), \textit{there exist a} \(C > 0\) \textit{such that}

\[
a(\psi, \psi) = Re^{-1} |\psi|_2^2, \tag{2.10}
\]

\[
a(\psi, \phi) \leq Re^{-1} |\psi|_2 \cdot |\phi|_2, \tag{2.11}
\]

\[
|b(\xi, \psi, \phi)| \leq 2C^2_s |\xi|_2 \cdot |\psi|_2 \cdot |\phi|_2, \tag{2.12}
\]

\[
|b_0(\xi, \psi, \phi)| \leq C |\xi|_2 \cdot |\psi|_2 \cdot |\phi|_2, \tag{2.13}
\]

\[
|\langle f, \text{curl } \phi \rangle| \leq |f|_* \cdot |\phi|_2, \tag{2.14}
\]

\[
|\langle f, \text{curl } \phi \rangle| \leq C_p \|f\|_0 \cdot |\phi|_2, \tag{2.15}
\]
where $C_s$ is a Sobolev embedding constant and $C_p$ is a poincare constant.

Proof:

For $\psi, \xi, \phi \in H^2_0(\Omega)$, we have by direct computation, equations (2.11-2.15). Our task is now to prove (2.10). We have, by definition,

$$a(\psi, \psi) = |\nabla \psi|^2_0 = \int_{\Omega} \left\{ \left( \frac{\partial^2 \psi}{\partial x^2} \right)^2 + \left( \frac{\partial^2 \psi}{\partial y^2} \right)^2 + 2 \frac{\partial^2 \psi}{\partial x^2} \frac{\partial^2 \psi}{\partial y^2} \right\}. \quad (2.16)$$

$$|\psi|^2_2 = \int_{\Omega} \left\{ \left( \frac{\partial^2 \psi}{\partial x^2} \right)^2 + \left( \frac{\partial^2 \psi}{\partial y^2} \right)^2 + \left( \frac{\partial^2 \psi}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 \psi}{\partial y \partial x} \right)^2 \right\}. \quad (2.17)$$

Clearly, it suffices to prove (2.10) with $\psi \in D(\Omega)$; for such a function,

$$\int_{\Omega} \left( \frac{\partial^2 \psi}{\partial x \partial y} \right)^2 = -\int_{\Omega} \frac{\partial \psi}{\partial x} \cdot \frac{\partial^3 \psi}{\partial x^2 \partial y} - \int_{\Omega} \frac{\partial^2 \psi}{\partial x^2} \cdot \frac{\partial^2 \psi}{\partial y^2}, \quad (2.18)$$

as a double application of Green’s formula, and thus (2.10) is proved. ■

Let $N$ denote the finite constant

$$N := \sup_{\xi, \psi, \phi \in H^2_0(\Omega)} \frac{|b(\xi, \psi, \phi)|}{|\xi|_2 |\psi|_2 |\phi|_2}$$

and $|f|_*$ denote the dual norm:

$$|f|_* := \sup_{\phi \in H^2_0(\Omega)} \frac{(f, \text{curl } \phi)}{|\phi|_2}$$

Then we have the following theorem that can be proved using the method of [33].

Theorem 2.3.2

[33] For $N |f|_* Re^2 < 1$ and $f \in [H^{-1}(\Omega)]^2$, problem (2.6) has a unique solution $\psi$.

Moreover, there is a unique $p \in L^0_0(\Omega)$ such that $(\text{curl } \psi, p)$ solves problem (2.4).
To study (2.6) when the uniqueness condition $N \mid f \mid_{Re^2} < 1$ is not valid, we need to introduce the concept of a nonsingular solution of (2.3).

**Definition 2.3.1**

Let $X$ and $Y$ be two Banach spaces, $F$ a differentiable mapping from $X$ into $Y$, $F'$ its derivative, and let $\psi \in X$ be a solution of the equation $F(\psi) = 0$. We say that $\psi$ is a nonsingular solution if there exists a constant $\gamma > 0$ such that

$$\| F'(\psi) \cdot \phi \|_Y \geq \gamma \| \phi \|_X \quad \forall \; \phi \in X.$$  

In the streamfunction equation case, the mapping $F : H^2_0(\Omega) \rightarrow [H^2_0(\Omega)]'$ is defined by:

$$< F(\psi), \phi > = a(\psi, \phi) + b(\psi, \psi, \phi) - (f, \text{curl } \phi).$$

The nonlinear map $F$ is quadratic and can be shown to be everywhere differentiable in $H^2_0(\Omega)$ and its derivative $F'(\psi) \in \mathcal{L}(H^2_0(\Omega), [H^2_0(\Omega)]')$ is given by:

$$< F'(\psi) \cdot \phi, \xi > = a(\phi, \xi) + b(\psi, \phi, \xi) + b(\phi, \psi, \xi).$$

Hence, $\psi \in H^2_0(\Omega)$ is a nonsingular solution of (2.3) if and only if there exists a constant $\gamma > 0$ such that

$$\sup_{\phi \in H^2_0(\Omega)} \frac{a(\xi, \phi) + b(\psi, \xi, \phi) + b(\xi, \psi, \phi)}{\| \phi \|_2} \geq \gamma \| \xi \|_2, \quad \forall \; \xi \in H^2_0(\Omega). \quad (2.20)$$

**2.4 Two Level Method**

We consider the approximate solution of (2.3) by a two level finite element procedure. Let $X^h, X^H \subseteq H^2_0(\Omega)$ denote two conforming finite element meshes with
$H \gg h$. The method we consider computes an approximate solution $\psi^h$ in the finite element space $X^h$ by solving one linear system for the degrees of freedom in $X^h$. This particular linear problem requires the construction of a finite element space $X^H$ upon a very coarse mesh of width $'H \gg h'$ and then the solution of a much smaller system of nonlinear equations for an approximation in $X^H$. The solution procedure is then given as follows:

**Algorithm 2.4.1**

**Step 1.** Solve the nonlinear system on coarse mesh for $\psi^H \in X^H$:

$$a(\psi^H, \phi^H) + b(\psi^H, \psi^H, \phi^H) = (f, \text{curl } \phi^H), \text{ for all } \phi^H \in X^H. \quad (2.21)$$

**Step 2.** Solve the linear system on fine mesh for $\psi^h \in X^h$:

$$a(\psi^h, \phi^h) + b(\psi^h, \psi^h, \phi^h) = (f, \text{curl } \phi^h), \text{ for all } \phi^h \in X^h. \quad (2.22)$$

We shall give some examples of finite element spaces for the streamfunction formulation. We will impose boundary conditions by setting all the degrees of freedom at the boundary nodes to be zero and the normal derivative equal to zero at all vertices and nodes on the boundary. The inclusion $X^H \subset H^2_0(\Omega)$ requires the use of finite element functions that are continuously differentiable over $\Omega$.

**Argyris triangle:** The functions are quintic polynomials within each triangle and the 21 degrees of freedom are chosen to be the function values and the first and second derivatives at the vertices, and the normal derivative at the midsides.

**Clough-Tocher:** Here we subdivide each triangle into three triangles by joining the vertices to the centroid. In each of the smaller triangles, the functions are cubic polynomials. There are then 30 degrees of freedom needed to determine the three different
cubic polynomials associated with the three triangles. Eighteen of these are used to
ensure that, within the big triangle, the functions are continuously differentiable. The
remaining 12 degrees of freedom are chosen to be the function values and the first
derivatives at the vertices and the normal derivative at the midsides.

**Bogner-Fox-Schmit rectangle:** The functions are bicubic polynomials within each
rectangle. The degrees of freedom are chosen to be the function values, the first deriva-
tives, and the mixed second derivative at the vertices. We set the function and the
normal derivative values equal to zero at all vertices on the boundary.

**Bicubic Spline rectangle:** The functions are the product of cubic splines. These
functions are bicubic polynomials within each rectangle, twice continuously differen-
tiable over $\Omega$, and their degrees of freedom are the function values at the nodes ( plus
some additional ones on the boundary ). Below we prove that $\psi^H$ and $\psi^h$ exist in
Step 1 and Step 2. Also we will prove that Algorithm 2.4.1 produces an approximate
Figure 2.2: The Clough-Tocher triangular element

Figure 2.3: The Bogner-Fox-Schmit rectangular element
solution which satisfies the error bound

\[ | \psi - \psi^h |_2 \leq C \left\{ \inf_{w^h \in X^h} | \psi - w^h |_2 + | \ln h |^{1/2} \cdot | \psi - \psi^H |_1 \right\}. \tag{2.23} \]

As an example, consider the case of the Clough-Tocher triangle. For this element (see [15, 33, 36]) we have the following inequalities:

\[
| \psi - \psi^h |_j \leq C h^{4-j} (j = 0, 1, 2), \\
| \psi - \psi^H |_j \leq C H^{4-j} (j = 0, 1, 2).
\]

Thus if we seek an approximate solution \( \psi^h \) with the same asymptotic accuracy as \( \psi^h \) in \( | \cdot |_2 \), the above error bound shows that the superlinear scaling between coarse and fine meshes

\[ h = O(H^{3/2} \cdot | \ln H |^{1/4}) \tag{2.24} \]

suffices. Analogous scaling between coarse and fine meshes can be calculated from (2.23) by balancing error terms on the right hand side of (2.23) in the same way.

For each of the elements described above, we give in Table 2.1, the scaling between coarse and fine meshes.

| Element                  | \( | \psi - \psi^H |_2 \) | \( | \psi - \psi^H |_1 \) | Scaling                      |
|--------------------------|--------------------------|--------------------------|-----------------------------|
| Argyris triangle         | \( H^4 \)                | \( H^5 \)                | \( h \cdot \ln h \cdot ^{-1/4} = O(H^{5/2}) \) |
| Clough-Tocher triangle   | \( H^2 \)                | \( H^3 \)                | \( h \cdot \ln h \cdot ^{-1/4} = O(H^{3/2}) \) |
| Bogner-Fox-Schmit rectangle | \( H^2 \)            | \( H^3 \)                | \( h \cdot \ln h \cdot ^{-1/4} = O(H^{3/2}) \) |
| Bicubic spline rectangle | \( H^2 \)                | \( H^3 \)                | \( h \cdot \ln h \cdot ^{-1/4} = O(H^{3/2}) \) |

Table 2.1: Scaling of two level finite elements
2.5 The Error Bound

The basic bound on $b(, ,)$ and $b_0(, ,)$, given in Lemma 2.3.1

\[ | b(\psi, \phi, \xi) | \leq N | \psi |_2 \cdot | \phi |_2 \cdot | \xi |_2, \]
\[ | b_0(\psi, \phi, \xi) | \leq N | \psi |_2 \cdot | \phi |_2 \cdot | \xi |_2, \]

can be improved. For our purpose we shall be bounding $| b(\psi, \phi, \xi) |$ with $\phi$ or $\xi$ in a finite element space $X^h$ or $X^H$. Since $X^h$ and $X^H$ are subspaces of $X$, then they satisfy the following discrete Sobolev inequality: for all $\phi^h \in X^h$ (similarly for $X^H$):

\[ \| \nabla \phi^h \|_{L^\infty} \leq c \ln(h) \| \phi^h \|_2. \]

Using the above inequality and Lemma 2.3.1, we can prove the following lemma:

**Lemma 2.5.1**

For any $\phi^h \in X^h$, the following inequalities

\[ | b(\psi, \phi^h, \xi) | \leq C \ln(h) \| \psi \|_2 \cdot \| \xi \|_1 \cdot \| \phi^h \|_2, \]
\[ | b(\psi, \xi, \phi^h) | \leq C \ln(h) \| \psi \|_2 \cdot \| \xi \|_1 \cdot \| \phi^h \|_2 \]

are hold.

**Lemma 2.5.2**

The solution to (2.21) exists and satisfies $| \psi^H \|_2 \leq Re \| f \|_*$. Suppose

\[ Re^2 N \| f \|_* < 1. \]

Then, the solution $\psi^H$ to (2.21) is unique.
Proof: Set \( \phi^H = \psi^H \) in (2.21). This gives

\[
Re^{-1} | \psi^H |_2^2 = (f, \text{curl } \psi^H) \leq |f|_* |\psi^H|_2,
\]

thus \( |\psi^H|_2 \leq Re |f|_* \). This bound implies the existence of the solution to (2.21) by a compactness argument in \( X^H \). Let \( \psi_1^H \) and \( \psi_2^H \) be two solutions to (2.21), and \( z^H = \psi_1^H - \psi_2^H \). Then,

\[
Re^{-1} | z^H |_2^2 = a(z^H, z^H) + b(\psi_1^H, z^H),
\]

\[
= a(\psi_1^H, z^H) + b(\psi_1^H, \psi_1^H, z^H) - (a(\psi_2^H, z^H) + b(\psi_1^H, \psi_2^H, z^H)),
\]

\[
= b(\psi_2^H, \psi_2^H, z^H) - b(\psi_1^H, \psi_2^H, z^H),
\]

\[
\leq N |z^H|_2^2 |\psi_2^H|_2 \leq NRe |f|_* z^H |_2^2,
\]

which implies uniqueness of solutions for \( (1 - NRe^2 |f|_*) > 0 \), as

\[
Re^{-1}(1 - NRe^2 |f|_*) | z^H |_2^2 \leq 0.
\]

The next theorem gives the basic error bound after step1 in the \( | \cdot |_2 \)-seminorm. Before we state the theorem we need the following lemma.

Lemma 2.5.3

Let \( \psi \) be a nonsingular solution of (2.3) and provided \( |\psi - \psi^H|_2 \leq \frac{\gamma}{2N} \), then there is a constant \( \gamma^* = \gamma^*(\psi) \) such that

\[
\sup_{\phi \in H^2_0(\Omega)} \frac{a(\xi, \phi) + b(\psi^H, \xi, \phi) + b(\xi, \psi, \phi)}{|\phi|_2} \geq \gamma^* |\xi|_2, \quad \forall \xi \in H^2_0(\Omega). \tag{2.28}
\]
Proof:
From (2.20) simply follows that for $|\psi - \psi^H|_2$ small enough (which is the case with $0 < H \leq H_0$)

$$\sup_{\phi \in H^2_0(\Omega)} \left\{ \frac{a(\xi, \phi) + b(\psi^H, \xi, \phi) + b(\xi, \psi, \phi)}{|\phi|_2} + \frac{b(\psi - \psi^H, \xi, \phi)}{|\phi|_2} \right\} \geq \gamma |\xi|_2, \forall \xi \in H^2_0(\Omega).$$

But it follows from (2.12) that

$$\sup_{\phi \in H^2_0(\Omega)} \frac{a(\xi, \phi) + b(\psi^H, \xi, \phi) + b(\xi, \psi, \phi)}{|\phi|_2} + N |\psi - \psi^H|_2 |\xi|_2 \geq \gamma |\xi|_2, \forall \xi \in H^2_0(\Omega),$$

or

$$\sup_{\phi \in H^2_0(\Omega)} \frac{a(\xi, \phi) + b(\psi^H, \xi, \phi) + b(\xi, \psi, \phi)}{|\phi|_2} \geq (\gamma - N |\psi - \psi^H|_2) |\xi|_2, \forall \xi \in H^2_0(\Omega).$$

Hence, we have (2.20). ■

Theorem 2.5.1
(a) If the global uniqueness condition $Re^2 N |f|_* < 1$ holds, $\psi$ and $\psi^H$ both exist uniquely. The error $|\psi - \psi^H|_2$ satisfies:

$$|\psi - \psi^H|_2 \leq C(Re) \inf_{w \in X^h} |\psi - w^H|_2,$$

where $C(Re) = (1 + 2N |f|_* \cdot Re^2)(1 - N |f|_* Re^2)^{-1} \leq C(\sqrt{N |f|_*}).$

(b) If the uniqueness condition fails, suppose $\psi$ is non-singular solution of (2.6). Then, there is an $H_0 = H_0(\psi, f, Re)$ and $c = c(\psi, f, Re, N)$ such that for $H \leq H_0$,

$$|\psi - \psi^H|_2 \leq c(\psi, f, Re, N) \inf_{w \in X^h} |\psi - w^H|_2,$$  \hspace{1cm} (2.30)
where $c(\psi, f, Re, N) = \gamma^{-1}(Re^{-1} + N \cdot Re \mid f \mid_*) + 1$

**Proof:**

Detailed proof of part (a) can be found in [17]. It remains to show part (b). Subtracting (2.21) from (2.6), gives the error equation for (2.21):

$$a(\psi - \psi^H, \phi^H) + b(\psi, \phi^H) - b(\psi^H, \psi^H, \phi^H) = 0$$

Adding the following terms $b(\psi^H, \psi, \phi^H) - b(\psi^H, \psi, \phi^H)$ gives:

$$a(\psi - \psi^H, \phi^H) + b(\psi - \psi^H, \psi, \phi^H) + b(\psi^H, \psi - \psi^H, \phi^H) = 0$$

Let $w^h \in X^h$ be an approximation to $\psi$ in $X^h$ and define $\xi^h = \psi^h - w^h$ and $\eta^h = \psi - w^h$ then the above inequality becomes:

$$a(\xi^H, \phi^H) + b(\xi^H, \psi, \phi^H) + b(\psi^H, \xi^H, \phi^H) =$$

Using (2.28) gives:

$$\gamma |\xi^H|_2 \leq \sup_{\phi^H \in X^h} \left\{ |\phi^H|_2^{-1} \left( a(\eta^H, \phi^H) + b(\eta^H, \psi, \phi^H) + b(\psi^H, \eta^H, \phi^H) \right) \right\}$$

In view of (2.11, 2.12), we have:

$$|\xi^H|_2 \leq \gamma^{-1} \left( Re^{-1} + N(\mid \psi \mid_2 + \mid \psi^H \mid_2) \right) |\eta^H|_2.$$
Lemma 2.5.4

Given a solution $\psi^H$ to (2.21), then the solution to the following problem:

\[
\text{Find } \hat{\psi} \in H^2_0(\Omega) \text{ such that, for all } \phi \in H^2_0(\Omega),
\]

\[
a(\hat{\psi}, \phi) + b(\psi^H, \hat{\psi}, \phi) = l(\phi)
\]

exists uniquely and satisfies: \[ \| \hat{\psi} \|_2 \leq \text{Re} | f |_* . \]

Proof:

Introducing the continuous bilinear form $B : H^2_0(\Omega) \times H^2_0(\Omega) \to R$ given by

\[
B(\psi, \phi) = a(\psi, \phi) + b(\psi^H, \psi, \phi).
\]

$B$ is continuous and coercive. Hence, $\hat{\psi}$ exists uniquely.

Setting $\phi = \hat{\psi}$ in (2.31) implies that:

\[
\text{Re}^{-1} \| \hat{\psi} \|^2_2 = l(\hat{\psi}),
\]

\[
\| \hat{\psi} \|^2_2 = \text{Re} \frac{l(\hat{\psi})}{\| \hat{\psi} \|^2_2},
\]

\[
\leq \text{Re} \sup_{\phi \in H^2_0(\Omega)} \frac{l(\phi)}{\| \phi \|^2_2}
\]

\[
= \text{Re} | f |_* . \]

Lemma 2.5.5

Given a solution $\psi^H$ to (2.21), then the solution to (2.22) exists uniquely and satisfies:

\[
\| \psi^h \|_2 \leq \text{Re} | f |_* .
\]
Proof:

The Bilinear form $B$ is continuous and coercive on $X^h$. Hence, $\psi^h$ exists uniquely.

Setting $\phi^h = \psi^h$ in (2.22) implies that:

\[
Re^{-1} \| \psi^h \|^2_2 = l(\psi^h),
\]

\[
= Re \frac{l(\psi^h)}{\| \psi^h \|^2_2},
\]

\[
\leq Re \sup_{\phi \in H_0^2(\Omega)} \frac{l(\phi)}{\| \phi \|^2_2},
\]

\[
= Re | f |^*_s. \quad \blacksquare
\]

By Green’s formula, we obtain the following lemma.

Lemma 2.5.6

For $\psi, \xi, \phi \in H_0^2(\Omega)$, we have

\[
b(\psi, \xi, \phi) = b_0(\xi, \phi, \psi) - b_0(\phi, \xi, \psi). \tag{2.33}
\]

Proof:

Applying Green’s formula to the left hand side of (2.33) give:

\[
b(\psi, \xi, \phi) = \int_\Omega \Delta \psi (\xi_y \phi_x - \xi_x \phi_y) d\Omega,
\]

\[
= - \int_\Omega \psi_x (\xi_y \phi_x - \xi_x \phi_y)_x + \psi_y (\xi_y \phi_x - \xi_x \phi_y)_y
\]

\[
+ \int_{\partial \Omega} \frac{\partial \psi}{\partial n} \cdot (\xi_y \phi_x - \xi_x \phi_y),
\]
The main result of this paper is the following theorem. It gives the error bound after step2.

**Theorem 2.5.2**

Let $X^{h,H} \subset H^2_0(\Omega)$ be two finite element spaces. Let $\psi$ be the solution to (2.3) and $\psi^h$ the solution to (2.22). Then $\psi^h$ satisfies:

$$
| \psi - \psi^h |_2 \leq C_1 \inf_{w^h \in X^h} | \psi^h - w^h |_2 + C_2 \sqrt{\ln h} | | \psi - \psi^H |_1,
$$

where $C_1 = 2 + N \ | f |_* \ Re^2$ and $C_2 = 2N \cdot Re^2 \ | f |_* \ C_s$, $\ C_s$ is the Sobolev constant.

**Proof:**

Subtracting (2.22) from (2.3) yields:

$$
a(\psi - \psi^h, \phi^h) + b(\psi, \phi) - b(\psi^H, \psi^h, \phi^h) = 0 \ \forall \phi^h \in X^h.
$$
Using lemma 2.5.6 gives:

$$a(\psi - \psi^h, \phi^h) + b_0(\psi, \phi^h, \psi) - b_0(\phi^h, \psi, \psi)$$

$$- b_0(\psi^h, \phi^h, \psi^H) + b_0(\phi^h, \psi^h, \psi^H) = 0 \quad \forall \phi^h \in X^h.$$  

Adding the following terms:

$$-b_0(\psi^h, \phi^h, \psi) + b_0(\phi^h, \psi^h, \psi) + b_0(\psi^h, \phi^h, \psi) - b_0(\phi^h, \psi^h, \psi),$$

gives:

$$a(\psi - \psi^h, \phi^h) + b_0(\psi - \psi^h, \phi^h, \psi) + b_0(\phi^h, \psi^h - \psi, \psi)$$

$$+ b_0(\psi^h, \phi^h, \psi - \psi^H) + b_0(\phi^h, \psi^h, \psi^H - \psi) = 0.$$  

Let $w^h \in X^h$ be an approximation to $\psi$ in $X^h$ and define $\xi^h = \psi^h - w^h$ and $\eta^h = \psi - w^h$, then the above inequality becomes:

$$a(\eta^h, \phi^h) + b_0(\eta^h, \phi^h, \psi) - b_0(\phi^h, \eta^h, \psi)$$

$$+ b_0(\psi^h, \phi^h, \psi - \psi^H) + b_0(\phi^h, \psi^h, \psi^H - \psi) =$$

$$a(\xi^h, \phi^h) + b_0(\xi^h, \phi^h, \psi) - b_0(\phi^h, \xi^h, \psi).$$

Setting $\phi^h = \xi^h$ implies:

$$a(\xi^h, \xi^h) = a(\eta^h, \xi^h) + b_0(\eta^h, \xi^h, \psi) - b_0(\xi^h, \eta^h, \psi)$$

$$+ b_0(\psi^h, \xi^h, \psi - \psi^H) + b_0(\xi^h, \psi^h, \psi^H - \psi).$$
Using lemma 2.5.6 gives:

\[
a(\xi^h, \xi^h) = a(\eta^h, \xi^h) + b(\psi, \eta^h, \xi^h) + b_0(\psi^h, \xi^h, \psi - \psi^H) + b_0(\xi^h, \psi^h, \psi^H - \psi).
\]

We will bound the right hand side of the above inequality as follows:

\[
a(\eta^h, \xi^h) \leq R e^{-1} |\eta^h|_2 \cdot |\xi^h|_2,
\]

\[
b(\psi, \eta^h, \xi^h) \leq N |\psi|_2 \cdot |\eta^h|_2 \cdot |\xi^h|_2,
\]

\[
b_0(\psi^h, \xi^h, \psi - \psi^H) \leq N \cdot c |\psi^h|_2 \cdot |\xi^h|_2 \cdot |\psi - \psi^H|_1 \cdot \sqrt{\ln h},
\]

\[
b_0(\xi^h, \psi^h, \psi^H - \psi) \leq N \cdot c |\psi^h|_2 \cdot |\xi^h|_2 \cdot |\psi - \psi^H|_1 \cdot \sqrt{\ln h}.
\]

Using these bounds gives:

\[
R e^{-1} |\xi^h|_2^2 \leq R e^{-1} (1 + N |\psi|_2 \cdot R e) |\eta^h|_2 \cdot |\xi^h|_2
\]

\[
+ 2N c |\psi^h|_2 \cdot |\xi^h|_2 \cdot |\psi - \psi^H|_1 \cdot \sqrt{\ln h}.
\]

Using the bounds on $|\psi|_2$ and $|\psi^h|_2$ gives:

\[
|\xi^h|_2 \leq (1 + N R e^2 |f|_2) |\eta^h|_2
\]

\[
+ (2N R e^2 |f|_2 c) \sqrt{\ln h} |\psi - \psi^H|_1.
\]

The triangle inequality ($|\psi - \psi^h|_2 \leq |\xi^h|_2 + |\eta^h|_2$) implies:

\[
|\psi - \psi^h|_2 \leq (2 + N R e^2 |f|_2) |\eta^h|_2
\]

\[
+ (2N R e^2 |f|_2 c) \sqrt{\ln h} |\psi - \psi^H|_1.
\]

(2.37)
Hence, we have the following estimates:

\[ | \psi - \psi^h |_2 \leq C_1 \inf_{w^h \in X^h} | \psi - w^h |_2 + C_2 \sqrt{\ln h} \cdot | \psi - \psi^H |_1. \]

\[ \blacksquare \]

**Corollary 2.5.1**

Let \( X^{h,H} \) be the Clough-Tocher elements. Then \( \psi^h \) satisfies:

\[ | \psi - \psi^h |_2 \leq C_1 h^2 + C_2 \sqrt{\ln h} H^3. \]

### 2.6 Summary

Two level method for the streamfunction formulation of the Navier-Stokes equations was discussed. The method is important because of the superlinear scaling between the coarse and fine grids. The error between the coarse and fine meshes are related superlinearly via:

\[ | \psi - \psi^h |_2 \leq C \{ \inf_{w^h \in X^h} | \psi - w^h |_2 + | \ln h |^{1/2} \cdot | \psi - \psi^H |_1 \}. \]

As an example, if the Clough-Tocher triangles or the Bogner-Fox-Schmit rectangles are used, then the coarse and fine meshes are related by \( h = O(H^{3/2} \ln H)^{1/4} \).
Chapter 3

A *Posteriori* Error Estimator
3.1 Introduction

The use of self-adaptive mesh refinement techniques based on \textit{a posteriori} error estimators is needed feature in any high-level finite element codes which are used for solving a practical problem of physics or engineering such as, computational fluid dynamics, elasticity, or semiconductor device simulation. In adaptive computation, there is a great need of reliable error estimators for these types of practical problems.

Moreover, \textit{a priori} error estimates are insufficient for mesh refinement because they only give information on the asymptotic error behavior. Thus \textit{a posteriori} error estimators must be extracted from the computed numerical solution and the data of the problem. The development of such estimators has been the subject of active research over the last few years (see [66, 67] for reference). A very popular choice for such estimators is weighted residual estimators which use some mesh-dependent norm of the residual (see [6, 7]).

In these pages we would like to consider the application of \textit{a posteriori} error analysis described in [66] to a two level finite element method for the streamfunction formulation of the Navier-Stokes equations. We obtain a reliable \textit{posteriori} error estimator in this problem. The attractions of the streamfunction formulation of the Navier-Stokes equations are that the incompressibility constraint is automatically satisfied, the pressure is not presented in the weak form and there is only one scalar unknown to solve for. The standard weak formulation for the streamfunction version was apparently first studied with mathematical rigor in 1979 in [33]. In this direction, Cayco and Nicolaides [17, 16] studied a general analysis of convergence for this standard weak formulation of the Navier-Stokes equations for Reynolds numbers small enough to ensure a global uniqueness condition holds.

The discretization of the streamfunction formulation still leads to a problem of
solving a large and ill-conditioned nonlinear systems of algebraic equations. Two level finite element discretization are presently a very promising approach for approximating the Navier-Stokes equations, see [41]. The computational attraction of the methods is that they require the solution of only a small system of nonlinear equations on coarse mesh and one linear system of equations on fine mesh. These types of methods were pioneered by Xu in [69, 70] for semilinear elliptic problems and then analyzed for the Navier-Stokes equations using velocity-pressure formulation in [41, 43, 44] and for the streamfunction formulation of the Navier-Stokes equations in [26, 72].

3.2 Notations and Preliminaries

We first need to define some function spaces and associated norms. More detailed concerning these spaces can be found in [1]. Let $\Omega$ be a bounded simply connected polygonal domain in $\mathbb{R}^2$. $L^2(\Omega)$ is the Hilbert space of Lebesgue square integrable functions with norm $\| \cdot \|_0$ and $L^2_0(\Omega)$ is the subspace of $L^2(\Omega)$ consisting of functions with zero mean. Let $H^m(\Omega)$ be the usual Sobolev space consisting of functions together with their distributed derivatives up through order $m$ are in $L^2(\Omega)$. Denote the norm on $H^m(\Omega)$ by $\| \cdot \|_m$. Let $H^m_0(\Omega)$ be the completion of $C^\infty_0(\Omega)$ under the $\| \cdot \|_m$ norm. We equip $H^m_0(\Omega)$ with the seminorm $| \cdot |_m$, which is a norm equivalent to $\| \cdot \|_m$. Also, the dual of space $H^m_0(\Omega)$ is denoted by $H^{-m}(\Omega)$, with norm $\| \cdot \|_{-m}$. Let $[H^m(\Omega)]^2$ be the space $H^m(\Omega) \times H^m(\Omega)$ and $[H^m_0(\Omega)]^2$ be the space $H^m_0(\Omega) \times H^m_0(\Omega)$.
equipped with the following norm

\[ \| u \|_m = (\| u_1 \|_m^2 + \| u_2 \|_m^2)^{1/2} \text{ and} \]

\[ | u |_m = (| u_1 |_m^2 + | u_2 |_m^2)^{1/2} \text{ where } u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \]

For each \( \phi \in H^1(\Omega) \), define

\[ \text{curl } \phi = \begin{pmatrix} \phi_y \\ -\phi_x \end{pmatrix}. \]

For each \( u \in [H^1(\Omega)]^2 \), define

\[ \text{curl } u = \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y}, \text{ where } u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \]

Consider the Navier-Stokes equations describing the flow of an incompressible fluid:

\[ -Re^{-1} \Delta u + (u \cdot \nabla)u + \nabla p = f, \text{ in } \Omega, \]

\[ \nabla \cdot u = 0, \text{ in } \Omega, \]

\[ u = 0, \text{ on } \partial \Omega, \]

\[ \int_{\Omega} p \ d\Omega = 0. \]  \( (3.1) \)

Any divergence-free velocity vector \( u \in [H^1_0(\Omega)]^2 \) has a unique stream function \[33, \text{ Theorem}3.1, \text{page} \ 22\] \( \psi \in H^2_0(\Omega) \), defined by

\[ \text{curl } \psi = u. \]
Moreover, the streamfunction $\psi$ satisfies

$$
Re^{-1} \Delta^2 \psi - \psi_y \Delta \psi_x + \psi_x \Delta \psi_y = \text{curl } f, \text{ in } \Omega,
$$

$$
\psi = 0, \text{ on } \partial \Omega,
$$

$$
\frac{\partial \psi}{\partial \hat{n}} = 0, \text{ on } \partial \Omega,
$$

(3.3)

where $\hat{n}$ represents the outward unit normal to $\Omega$, which is simply the $\text{curl}$ of (3.1).

The standard weak form of equation (3.3) is:

Find $\psi \in H_0^2(\Omega)$ such that, for all $\phi \in H_0^2(\Omega)$,

$$
a(\psi, \phi) + b(\psi; \psi, \phi) = l(\phi),
$$

(3.4)

where

$$
a(\psi, \phi) = Re^{-1} \int_\Omega \Delta \psi \cdot \Delta \phi,
$$

$$
b(\xi; \psi, \phi) = \int_\Omega \Delta \xi (\psi_y \phi_x - \psi_x \phi_y),
$$

(3.5)

$$
l(\phi) = (f, \text{curl } \phi) = \int_\Omega f \cdot \text{curl } \phi.
$$

Another equivalent formulation of equation (3.3), introduced by Cayco and Nicolaides [16], is:

Find $\psi \in H_0^2(\Omega)$ such that, for all $\phi \in H_0^2(\Omega)$,

$$
a_0(\psi, \phi) + b_0(\psi; \psi, \phi) = l(\phi),
$$

(3.6)
where

\[ a_0(\psi, \phi) = \text{Re}^{-1} \int_\Omega \psi_{xx} \phi_{xx} + 2\psi_{xy} \phi_{xy} + \psi_{yy} \phi_{yy}, \]

\[ b_0(\xi; \psi, \phi) = \int_\Omega (\xi_y \psi_{xy} - \xi_x \psi_{yy}) \phi_y - (\xi_x \psi_{xy} - \xi_y \psi_{xx}) \phi_x, \]

\[ l(\phi) = (f, \text{curl } \phi) = \int_\Omega f \cdot \text{curl } \phi. \]

Conforming element can be used with either (3.4) or (3.6). In this case the two weak formulations produce identical results because \( a(\psi, \phi) = a_0(\psi, \phi) \) and \( b(\xi; \psi, \phi) = b_0(\xi; \psi, \phi) \) for all \( \psi, \phi, \xi \in H^2_0(\Omega) \). However, when using nonconforming approximating subspaces, (3.6) and (3.4) generate different finite element methods. Nonconforming elements should be used only with (3.6). (See [16, 36] for details).

Let \( \langle \cdot, \cdot \rangle \) be the \( L^2 \) inner product over \( \Omega \) and \( \langle \cdot, \cdot \rangle_e \) be the \( L^2 \) inner product over an element \( e \in \Pi^h(\Omega) \). \( \| \cdot \|_j \) and \( \| \cdot \|_{j,e} \) denote the Sobolev norm for \( H^j(\Omega) \) and \( H^j(e) \), respectively. The norm for the space \( X = H^2_0(\Omega) \) is \( \| \cdot \|_2 \). Define a nonlinear, continuous map \( F : X \to X' \) via the Riesz representation theorem: for all \( \psi, \phi \in X \)

\[ \langle F(\psi), \phi \rangle = a(\psi, \phi) + b(\psi, \psi, \phi). \]

The streamfunction formulation of the Navier-Stokes equations can be represented abstractly as:

find \( \psi \in X \) satisfying

\[ \langle F(\psi), \phi \rangle = (f, \text{curl } \phi) \quad \text{for all} \quad \phi \in X. \]

Assume that the solution \( \psi \) to (3.9) is nonsingular. Namely, \( DF(\psi) \) is invertable as
a map: \( X \to X' \) and
\[
\| DF(\psi)^{-1} \|_{L(X,X')} < \infty.
\]

Let \( \Pi^H(\Omega) \) and \( \Pi^h(\Omega) \), \( H, h > 0 \), be a two family of triangulations of \( \Omega \) which satisfies the following conditions:

1. any two triangles in \( \Pi^h(\Omega) \) share at most a common edge or a common vertex.

2. the minimal angle of all triangles in the whole family \( \Pi^h(\Omega) \) is bounded away from zero.

Denote by \( X^H \) and \( X^h \) the spaces of Clough-Tocher elements on triangles. In the Clough-Tocher triangle, we subdivide each triangle into three triangles by joining the vertices to the centroid. In each of the smaller triangles, the functions are cubic polynomials. There are then 30 degrees of freedom needed to determine the three different cubic polynomials associated with the three triangles. Eighteen of these are used to ensure that, within the big triangle, the functions are continuously differentiable. The remaining 12 degrees of freedom are chosen to be the function values and the first derivatives at the vertices and the normal derivative at the middies. See Figure (3.1) for illustration of the Clough-Tocher triangles. The basic two level discretization method can be written in the following algorithm.

**Algorithm 3.2.1**

**Step 1.** Calculate \( \psi^H \in X^H \) by solving the (small) nonlinear system
\[
< F(\psi^H), \phi^H > = (f, \text{curl } \phi^H), \quad \text{for all } \phi^H \in X^H.
\]

**Step 2.** Calculate \( \psi^h \in X^h \) by solving the linear system
\[
< F(\psi^H) + A(\psi^H)(\psi^h - \psi^H), \phi^h > = (f, \text{curl } \phi^h), \quad \text{for all } \phi^h \in X^h.
\]
where $A$ is a continuous map defined by:

$$< A(\psi)(\xi), \phi > = b(\xi, \psi, \phi).$$

### 3.3 Error Analysis

The basic two level algorithm now begins with a coarse mesh finite element space $X^H$ and a fine mesh finite element space $X^h$ and computes $\phi^h$ as follows.

**Algorithm 3.3.1**

**Step 1.** Solve the nonlinear system on coarse mesh for $\psi^H \in X^H$:

$$a(\psi^H, \phi^H) + b(\psi^H, \psi^H, \phi^H) = (f, \text{curl } \phi^H), \text{ for all } \phi^H \in X^H. \quad (3.10)$$
Step 2. Solve the linear system on fine mesh for $\psi^h \in X^h$:

$$a(\psi^h, \phi^h) + b(\psi^h, \psi^H, \phi^h) = (f, \text{curl } \phi^h), \text{ for all } \phi^h \in X^h. \quad (3.11)$$

The following relations show that the error between the coarse and fine meshes are related superlinearly via:

$$|\psi - \psi^h|_2 \leq C \left\{ \inf_{w^h \in X^h} |\psi - w^h|_2 + \ln h \right\}^{1/2} \cdot \frac{1}{2^2} \cdot |\psi - \psi^H|_1,$$

As an example, if the Bogner-Fox-Schmit rectangles, i.e the bicubic polynomials within each rectangle whose degree of freedom are chosen to be the function values the first derivatives and the mixed second derivatives at the vertices, see Figure (3.2) for illustration of the Bogner-Fox-Schmit rectangles, are used then the coarse and fine meshes are related by

$$h = O(H^{3/2} |\ln H|^{1/4}).$$

The following lemma states that $\psi^H$ in step 1 of Algorithm (3.2.1) exists uniquely and given $\psi^H$ the $\psi^h$ exists uniquely under some uniqueness condition, (see [26] for proof).

Lemma 3.3.1

(a) if the global uniqueness condition on $Re^2 N | f |_* < 1$ holds, then $\psi$ and $\psi^H$ are both unique. The error $|\psi - \psi^H|_2$ satisfies:

$$|\psi - \psi^H|_2 \leq C \inf_{w^H \in X^H} |\psi - w^H|_2,$$
Figure 3.2: Bogner-Fox-Schmit rectangular element

where

\[ C = C(Re) \quad \text{is positive constant,} \]

\[ |f|_* := \sup_{\phi \in H_0^2(\Omega)} \frac{(f, \text{curl } \phi)}{||\phi||_2}. \]

(b) If the uniqueness condition fails, suppose \( \psi \) is non-singular solution of (3.6). Then there is an \( H_0 = H_0(\psi, f, Re) \) and \( C = C(\psi, f, Re, N) \) such that for \( H \leq H_0 \).

\[ |\psi - \psi^H|_2 \leq C(\psi, f, Re, N) \inf_{w^H \in X^H} |\psi - w^H|_2, \]

where \( C \) is a positive constant.

(c) Let \( \psi \) be the solution to (3.10) and \( \psi^h \) be the solution to (3.11). Given \( \psi^H, \psi^h \) exists uniquely and satisfies:

\[ |\psi - \psi^h|_2 \leq C_1 \inf_{w^h \in X^h} |\psi^h - w^h|_2 + C_2 \sqrt{\ln h} |\psi - \psi^H|_1 \]
where $C_1$ and $C_2$ are positive constants.

The main result of this thesis is given in the next theorem. Before we state the main theorem, we begin this paragraph by introducing some additional notations which will be needed for constructing and analyzing the error estimators.

For any $e \in \pi^h(\Omega)$ we denote $\partial e$ by the set of its edges. With every edge $g \in \partial e$ we associate a unit vector $\hat{n}_g$ such that $\hat{n}_g$ is orthogonal to $g$ and equals the unit exterior normal to $\Gamma$ if $g \subset \Gamma$. Given any $g \in \partial e$ and any $\phi \in X^h$, we denote $[\phi]_g$ the jump of $\phi$ across $g$ in the direction $\hat{n}_g$:

$$
[\phi]_g(x) := \lim_{t \to 0^+} \phi(x + t\hat{n}_g) - \lim_{t \to 0^+} \phi(x - t\hat{n}_g) \quad \forall x \in g.
$$

For any triangle $e$ and edge $g$, let $h_e$ and $h_g$ be their diameter and length, respectively.

**Theorem 3.3.1**

Let $\psi^h$ be the approximate solution generated by Algorithm (3.2.1). Suppose that $\psi$ is a nonsingular solution to equation (3.9) and $X^H, X^h \subset X$ with $H, h$ small enough. Then, there are computable constants $C_1$ and $C_2$ such that the following a posteriori error bound holds:

$$
\| \psi^h - \psi \|_X \leq 2 \| DF(\psi)^{-1} \|_{L(X,X')} \{ C_1 \sum_{e \in \Pi^h(\Omega)} [h_e^4 \| r^h \|_{2,e}^2 + \sum_{g \subset \partial e} \eta_g] \\
+ C_2 \ln h \| \frac{1}{2} \cdot | \psi^h |_2 \cdot | \psi^H - \psi^h |_1 \},
$$

(3.12)
where

\[ \eta_g = h_g \| Re^{-1}[G]_g \|^2_{L^2(g)} + h_g^3 \| \mathbf{f} \times \hat{n}_g + Re^{-1}[\nabla \Delta \psi^h \cdot \hat{n}_g]_g + [\hat{F}_g \cdot \hat{n}_g] \|^2_{L^2(g)}, \]

\[ r^h = \text{curl } \mathbf{f} - [Re^{-1} \Delta^2 \psi^h \psi^h_y \Delta \psi^h_x + \psi^h_x \Delta \psi^h_y], \]

\[ \hat{F} = \begin{bmatrix} \psi^h_y \psi^h_{yx} - \psi^h_x \psi^h_{xy} \\ \psi^h_y \psi^h_{xy} - \psi^h_x \psi^h_{yy} \end{bmatrix}, \]

\[ G = \begin{bmatrix} \nabla \psi^h \cdot \hat{n}_g \\ \nabla \psi^h_y \cdot \hat{n}_g \end{bmatrix} \text{ for any edge } g. \]

**Proof**

Since \( \psi \) is a nonsingular solution, it can be shown by using the method of \([33, 32]\) that \( \psi^H \to \psi \) in \( X \) as \( X^H \) becomes dense in \( X \). In particular, for \( H \) small enough \( DF(\psi^H)^{-1} \) exists and \( \| DF(\psi^H)^{-1} \|_{L(X,X')} \to \| DF(\psi)^{-1} \|_{L(X,X')} \) as \( H \to 0 \). Also \( \psi^h \to \psi \) strongly in \( X \) as \( H, h \to 0 \).

Therefore, Proposition 7.1 of Verfurth [66] can be applied for \( H \) sufficiently small. Thus, we have

\[ \| \psi^h - \psi \|_X \leq 2 \| DF(\psi)^{-1} \|_{L(X,X')} \| F(\psi^h) - F(\psi) \|_{X'}. \]

The definition of the norm in \( X' \) gives,

\[ \| F(\psi^h) - F(\psi) \|_{X'} = \sup_{\phi \neq 0 \in X} \frac{|Q|}{\| \phi \|_2}, \]

where

\[ Q = a(\psi, \phi) + b(\psi, \psi, \phi) - a(\psi^h, \phi) - b(\psi^h, \psi^h, \phi). \]
The following approximate Galerkin orthogonality relation holds for all $\phi^h \in X^h$

$$a(\psi - \psi^h, \phi^h) + b(\psi, \phi^h) - b(\psi^h, \psi^H, \phi^h) = 0. \tag{3.15}$$

Equation (3.15) changes to

$$a(\psi - \psi^h, \phi^h) + b(\psi, \phi^h) - b(\psi^h, \psi^h, \phi^h) = b(\psi^h, \psi^H - \psi^h, \phi^h), \tag{3.16}$$

since

$$a(\psi - \psi^h, \phi^h - \phi) + b(\psi, \phi^h - \phi) - b(\psi^h, \psi^h, \phi^h - \phi)$$

$$= a(\psi - \psi^h, \phi^h) + b(\psi, \phi^h) - b(\psi^h, \psi^h, \phi^h) - Q.$$ 

Since $\phi - \phi^h \in X$, we have

$$a(\psi, \phi - \phi^h) + b(\psi, \phi - \phi^h) = (f, \text{curl}(\phi - \phi^h)). \tag{3.17}$$

Equations (3.16)-(3.17) give

$$Q = (f, \text{curl}(\phi - \phi^h)) - a(\psi^h, \phi - \phi^h)$$

$$- b(\psi^h, \psi^h, \phi - \phi^h) + b(\psi^h, \psi^H - \psi^h, \phi^h). \tag{3.18}$$
Applying Green’s theorem twice to the second term of (3.18) gives:

\[
a(\psi^h, \phi - \phi^h) = Re^{-1} \sum_{e \in \Pi^h(\Omega)} \int_e \psi^h_{xx}(\phi^h_{xx} - \phi^h_{xx}) + 2\psi^h_{xy}(\phi^h_{xy} - \phi^h_{xy}) \\
+ \psi^h_{yy}(\phi^h_{yy} - \phi^h_{yy}),
\]

\[
= Re^{-1} \sum_{e \in \Pi^h(\Omega)} \left\{ - \int_e (\phi_x - \phi^h_x) \nabla^2 \psi^h_x + (\phi_y - \phi^h_y) \nabla^2 \psi^h_y \\
+ \sum_{g \in \partial e} \int_g \left[ (\phi_x - \phi^h_x) \frac{d\psi^h_x}{dn} + (\phi_y - \phi^h_y) \frac{d\psi^h_y}{dn} \right] ds \right\},
\]

\[
= \sum_{e \in \Pi^h(\Omega)} \left\{ Re^{-1} \int_e \nabla^2 \psi^h_x (\phi - \phi^h) \\
+ \sum_{g \in \partial e} \left\{ Re^{-1} \int_g \left[ \phi_x - \phi^h_x \frac{d\psi^h_x}{dn} + \phi_y - \phi^h_y \frac{d\psi^h_y}{dn} \right] ds \\
- Re^{-1} \int_g (\phi - \phi^h) \frac{d\nabla^2 \psi^h_x}{dn} ds \right\} \right\}.
\]

where \([w]_g\) denotes the jump in \(w\) across the edge \(g\). Applying Green’s theorem to the third term of (3.18) gives:

\[
b(\psi^h, \psi^h, \phi - \phi^h) = \int_\Omega \left( \psi^h_y \psi^h_{xy} - \psi^h_x \psi^h_{yy} \right) (\phi_y - \phi^h_y) \\
- \left( \psi^h_x \psi^h_{xy} - \psi^h_y \psi^h_{xx} \right) (\phi_x - \phi^h_x),
\]

\[
= \sum_{e \in \Pi^h(\Omega)} \left\{ Re^{-1} \int_e \nabla \psi^h_x \nabla \psi^h_y (\phi - \phi^h) \\
+ \sum_{g \in \partial e} \left\{ \left[ \psi^h_y \psi^h_{xx} - \psi^h_x \psi^h_{yy} \right]_g (\phi - \phi^h) \frac{dx}{dn} \\
+ \left[ \psi^h_x \psi^h_{xy} - \psi^h_y \psi^h_{yy} \right]_g (\phi - \phi^h) \frac{dy}{dn} \right\} ds. \]
\]

(3.19)
The first term of (3.18) equals:

\[
(f, \text{curl}(\phi - \phi^h)) = \sum_{e \in \Pi^h(\Omega)} \left[ \int_e \text{curl } f(\phi - \phi^h) \right] + \sum_{g \subset \partial e} (\phi - \phi^h)f \times \hat{n}. \tag{3.21}
\]

Using (3.19)-(3.21) in (3.18) give:

\[
Q = \sum_{e \in \Pi^h(\Omega)} \left\{ \int_e [-Re^{-1} \Delta^2 \psi^h + \psi^h \Delta \psi^h_x - \psi^h \Delta \psi^h_y + \text{curl } f](\phi - \phi^h) \right. \\
+ \sum_{g \subset \partial e} \left[ \int_g (\phi - \phi^h)f \times \hat{n} \right. \\
- Re^{-1} \int_g \{(\phi_x - \phi^h_x)[\frac{d\psi^h_x}{dn}]_g + (\phi_y - \phi^h_y)[\frac{d\psi^h_y}{dn}]_g \}ds \\
+ Re^{-1} \int_g (\phi - \phi^h)[\frac{d \Delta \psi^h}{dn}]_g ds \\
+ \int_g \{[\psi^h_{yy} \psi^h_{xx} - \psi^h_{x} \psi^h_{xy}](\phi - \phi^h) \frac{dx}{dn} + [\psi^h_{y} \psi^h_{xy} - \psi^h_{y} \psi^h_{yy}]_g (\phi - \phi^h) \frac{dy}{dn} \}ds \} \\
+ b(\psi^h, \psi^H - \psi^h, \phi^h). \tag{3.22}
\]
Rewrite (3.22) as follows:

\[
Q = \sum_{e \in \Pi^h(\Omega)} \{ r^h(\phi - \phi^h) \\
+ \sum_{g \subset \partial e} \int_g (\phi - \phi^h) f \times \hat{n}_g \\
- Re^{-1} \int_g [G]_g \cdot \nabla (\phi - \phi^h) \\
+ Re^{-1} \int_g (\phi - \phi^h) [\nabla \Delta \psi^h \cdot \hat{n}_g]_g \\
+ \int_g (\phi - \phi^h) \{ \hat{F} \}_g \cdot \hat{n}_g \} \\
+ b(\psi^h, \psi^H - \psi^h, \phi^h),
\]

where

\[
r^h = \text{curl } f - [Re^{-1} \Delta^2 \psi^h - \psi^h \Delta \psi^h_x + \psi^h_x \Delta \psi^h_y],
\]

\[
\hat{F} = \begin{bmatrix}
\psi^h_y \psi^h_{xx} - \psi^h_x \psi^h_{xy} \\
\psi^h_y \psi^h_{xy} - \psi^h_x \psi^h_{yy}
\end{bmatrix},
\]

\[
G = \begin{bmatrix}
\nabla \psi^h_x \cdot \hat{n}_g \\
\nabla \psi^h_y \cdot \hat{n}_g
\end{bmatrix}, \text{ for all } g \subset \partial e.
\]

We assume that there is an operator \( R^X_h : X \to X^h \), given in section 2 of [21] which satisfies the following

\[
\| \phi - R^X_h \phi \|_{0,2,e} \leq C_1 h^2 \| \phi \|_{2,2}, \quad (3.23)
\]

\[
\| \phi - R^X_h \phi \|_{L^2(g)} + h_g \| \nabla (\phi - R^X_h \phi) \|_{L^2(g)} \leq C_2 h^{3/2} \| \phi \|_{2,2}. \quad (3.24)
\]
Take $\phi^h = R_h^* \phi$. Using the Cauchy-Schwarz inequality on each element $e$ and edge $g$ and the inequalities (3.23), (3.24) give:

$$
\| F(\psi^h) - F(\psi) \|_{X'} \leq C \sum_{e \in \Pi h(\Omega)} [h_e^4 \| r \|_{2,e}^2]
\quad + \sum_g \{ h_g \| Re^{-1}[G]_g \|_{L_2(g)}^2 + h_g^3 \| \mathbf{f} \times \mathbf{n}_g \|_{L_2(g)}^2
\quad + Re^{-1}[\nabla \Delta \psi^h \cdot \mathbf{n}_g]+[\mathbf{F}]_g \cdot \mathbf{n}_g \|_{L_2(g)}^2 \}
\quad + b(\psi^h, \psi^H - \psi^h, \phi^h).
$$

It remains to estimate the trilinear term $b(\psi^h, \psi^H - \psi^h, \phi^h)$. If the discrete Sobolev inequality holds:

$$
\| \nabla \phi^h \|_{L_\infty} \leq \tilde{C} \ln(h) \frac{1}{2} \| \phi^h \|_2 \quad \text{for all } \phi^h \in X^h,
$$

then

$$
| b(\psi^h, \psi^H - \psi^h, \phi^h) | \leq \tilde{C} \| \psi^h \|_2 \cdot \| \psi^H - \psi^h \|_1 \cdot \| \nabla \phi^h \|_{L_\infty},
\leq \tilde{C} \ln(h)^{1/2} \cdot \| \psi^h \|_2 \cdot \| \psi^H - \psi^h \|_1 \cdot \| \phi^h \|_2.
$$

Therefore,

$$
\| \psi^h - \psi \|_X \leq 2 \| DF(\psi)^{-1} \|_{L(X,X')} \left\{ C \sum_{e \in \Pi h(\Omega)} [h_e^4 \| r^h \|_{2,e}^2 + \sum_{g \subset \partial e} \eta_g] \right\}
\quad + \tilde{C} \ln(h)^{1/2} \cdot \| \psi^h \|_2 \cdot \| \psi^H - \psi^h \|_1,
$$

(3.25)

**Remarks**

1. The right-hand side of equation (3.12) can be used as an *a posteriori* error
estimator since it only involves the known data $f$, the coarse solution $\psi^H$ and the fine solution $\psi^h$, except for the common multiplier $\|DF(\psi)^{-1}\|$.

2. The first term in the estimator is related to the residual of $\psi^h$. The last term is related to the difference between the coarse approximation $\psi^H$ and the fine approximation $\psi^h$.

3. Computing a reliable estimator for the error in $\psi^h$ requires the estimation of $\|DF(\psi)^{-1}\|_{L(X',X)}$. One idea of estimating $\|DF(\psi)^{-1}\|_{L(X',X)}$ is by computing $\|DF(\psi^H)^{-1}\|_{L(X'^H,X^H)}$.

4. The evaluation of the integrals occurring on the right-hand side of equation (3.12) can be approximated by suitable quadrature formula. With normal choices the error added by a quadrature scheme is of high order.
Chapter 4

Implementation of the Two-Level FEM
4.1 Introduction

This chapter focuses on the implementation of the finite element method for the one level method of the Navier-Stokes equations. Our intention is to give enough detail to enable someone competent in programming to understand the given code in Appendix C.

For specificity, consider the one level equation of the Navier-Stokes equation.

\[ Re^{-1} \nabla^2 \psi + \psi_y \nabla \psi_x - \psi_x \nabla \psi_y = \text{curl } f, \quad (x, y) \in \Omega, \]

\[ \psi = 0, \quad \text{on } \partial \Omega, \quad (4.1) \]

\[ \frac{\partial \psi}{\partial n} = 0, \quad \text{on } \partial \Omega, \]

where \( \Omega \) is a simply connected domain in \( \mathbb{R}^2 \) with regular boundary \( \partial \Omega = \Gamma \). The variation formulation of (4.1) that is used in the finite element program is as follows. A Morley element space or Bogner-Fox-Schmit element space \( S^H \) is chosen (with no boundary conditions imposed) and \( \psi^H \in S^H \) satisfies

\[ a(\psi^H, \phi^H) + b(\psi^H, \phi^H, \phi^H) = (f, \text{curl } \phi^H) \quad \forall \phi^H \in S^H, \]

such that \( \phi^H = 0, \quad \text{on } \Gamma_H \quad (4.2) \)

and \( \frac{\partial \phi^H}{\partial n} = 0 \quad \text{on } \Gamma_H. \quad (4.3) \)
Here,

\[ a(\psi, \phi) = \sum_{e_i} \int_{e_i} \left( \psi_{xx} \phi_{xx} + 2\psi_{xy} \phi_{xy} + \psi_{yy} \phi_{yy} \right) dxdy, \]

\[ b(\psi, \xi, \phi) = \sum_{e_i} \int_{e_i} \left\{ (\psi_{y} \xi_{xx} - \psi_{x} \xi_{xy}) \phi_{x} - (\psi_{x} \xi_{yy} - \psi_{y} \xi_{xy}) \phi_{y} \right\} dxdy. \quad (4.4) \]

In this chapter, we follow [2, 37, 46, 10] in writing the description of the code. The computer program using Morley elements written in FORTRAN 90 is included in Appendix A. Appendix B lists the computer program using the Bogner-Fox-Schmit rectangles written in FORTRAN 77. In writing these programs, we follow [25, 52, 53]. We also follow MATHEMATICA Manual [68] in writing MATHEMATICA programs that appear in this chapter.

### 4.2 Notations and Arrays

In this section, we list all relevant notations and arrays used in this chapter.

- **Morley triangles**
$lx =$ the number of elements to be constructed along the x-axis and the y-axis.

$m =$ the total number of nodes on $\Omega \cup \Gamma$.

$nelem =$ the total number of triangles in $\Omega$.

$nbf =$ the total number of nodes per element

$nqpp =$ the total number of integration points in the reference element.

$nbo =$ the total number of nodes on $\Gamma$.

$in =$ the number of nodes in $\Omega$.

$rhs(m) =$ the local vector.

$ielnode(nelem, nbf) =$ global node numbers for all elements.

$xx(m) =$ x-coordinates of all nodes.

$yy(m) =$ y-coordinates of all nodes.

$ibo(nbo) =$ the set of node numbers of node in $\Gamma$.

$w(nqp) =$ the corresponding integration weights on the reference element.

$xi(nqp) =$ x-coordinate of the integration points on the reference element.

$eta(nqp) =$ y-coordinates of the integration points on the reference element.

$phi(nbf, nqp) =$ values of $\tilde{\phi}_1, \ldots, \tilde{\phi}_{nbf}$ at the integration points.

$phix(nbf, nqp) =$ values of $\tilde{\phi}_{1, \xi}, \ldots, \tilde{\phi}_{nbf, \xi}$ at the integration points.

$phiy(nbf, nqp) =$ values of $\tilde{\phi}_{1, \eta}, \ldots, \tilde{\phi}_{nbf, \eta}$ at the integration points.
Bogner-Fox-Schmit

\(lx\) = the number of squares along the x-axis.

\(m\) = the total number of nodes on \(\Omega \cup \partial \Omega\).

\(nelem\) = the total number of rectangles in \(\Omega\).

\(nbf\) = the total number of nodes per element

\(nbo\) = the total number of nodes on the boundary except the y-derivative nodes on the top.

\(nboy\) = the total y-derivative nodes on the top.
rhs(m) = the rhs vector of the linear system.

ielnode(nelem, nbf) = global node numbers for all elements.

xx(m) = x-coordinates of all nodes.

yy(m) = y-coordinates of all nodes.

ibo(nbo) = the set of node numbers of nodes in

\( \partial\Omega \) except the y-derivative nodes on the top.

iboy(nboy) = the set of node numbers of y-derivative node on the top.

\( xi(4) \) = x-coordinate of integration points on the reference.

\( eta(4) \) = y-coordinate of integration points on the reference.

\( w(4) \) = the corresponding integration weights

on the reference element.

\( q(nb,..., 4) \) = the basis function values, x-derivative,
y-derivative, xy-derivative, xx-derivative

and yy-derivative of all 16 basis functions

at the integration points.

\[ ea(i, j) = \int\int_{\hat{\Omega}} (\tilde{\phi}_{i,xx} \tilde{\phi}_{j,xx} + 2\tilde{\phi}_{i,xy} \tilde{\phi}_{j,xy} + \tilde{\phi}_{i,yy} \tilde{\phi}_{j,yy}) d\xi d\eta. \]

\[ b(i, j, k) = \int\int_{\hat{\Omega}} \{(\tilde{\phi}_{j,y} \tilde{\phi}_{k,xx} - \tilde{\phi}_{j,x} \tilde{\phi}_{k,xy}) \tilde{\phi}_{i,x} \]

\[- (\tilde{\phi}_{j,x} \tilde{\phi}_{k,yy} - \tilde{\phi}_{j,y} \tilde{\phi}_{k,xy}) \tilde{\phi}_{i,y}\} d\xi d\eta. \]

4.3 Mesh Generation

SUBROUTINE MAELNODE generates the data points associated with a regular
triangle mesh on the unit square \([0, 1] \times [0, 1]\). The number of triangular elements to
be placed along each axis is determined by \(lx\). Since there must be an even number
of elements on each level, then \( m \) is always an even number. A crude diagram of the subject mesh is presented in Figure (4.1) for case \( lx = 3 \). the triangles are labeled in increasing order from bottom left corner to the top right corner. The number of nodes per element is set by \( nbf = 6 \). In Morley element form, MAELNODE places nodes at the vertices and at the midpoints of each edge. The nodes are labeled globally in increasing order from bottom left corner to the top right corner as shown in Figure (4.2). These nodes are numbered globally and their \( x - y \) coordinates are computed by the SUBROUTINE COORD(....) and stored in the real array \( x(:) \) and \( y(:) \). For the midside nodes, we have to store the information of the direction of the normal. Thus information will be described in the next paragraph.

In addition, it is necessary to have a local node ordering. An example of a local node ordering on a triangle is shown in Figure (4.3). By convention, we order the nodes counter clockwise beginning with the vertex nodes, then the edge nodes. For the midside nodes, we assign a positive value if the arrow is outward with respect to the
Figure 4.2: Global node labeling

Figure 4.3: Local node ordering
given triangle and negative value if it is inward.

This local ordering is stored in an array $ielnode(nelem \times nbf)$ where $nelem = lx^2$, $nbf = 6$. Specifically,

$$ielnode(i, j) := \text{the global node number of node } j \text{ in triangle } i,$$

$$i = 1, 2, \ldots \ldots, nbf = 6,$$

$$j = 1, 2, \ldots \ldots, nelem = 2*lx^2.$$  

For example, the coordinates of the third node in the fifth triangle are

$$x = x(ielnode(3, 5)), \quad y = y(ielnode(3, 5)).$$

Moreover, the boundary nodes are stored in the array $ibo$. The number of boundary nodes $nbo$ for a unit square is given by the following relation $nbo = 8*lx$. An example of $ielnode$ array of Figure (4.2) are given below. An example of a local vs. global node ordering for element $e_3$ in Figure (4.1) is given in the third row of $ielnode$ array (i.e) $(3, 5, 17, 4, 11, 10)$. Notice that the last three columns in $ielnode$ have some negative values. These last three columns represent the midsides points of each triangles. There negative values say that the arrow is inward. Moreover, the negative values appear only on the even rows because only the triangle given even number has
An example of \(x, y\) and \(ibo\) arrays of Figure (4.2) are given below. An example of x-coordinate and y-coordinate of the node 25 is (0.5,0.5) which is 25-th entry in x-array and y-array. Also, \(ibo\)-array lists all the boundary node numbers of Figure.
\[ x = \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0.1667 \\
0.1667 \\
0.1667 \\
0.1667 \\
0.1667 \\
0.1667 \\
0.3333 \\
0.3333 \\
0.3333 \\
0.3333 \\
0.3333 \\
0.3333 \\
0.3333 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.6667 \\
0.8333 \\
0.8333 \\
0.8333 \\
0.8333 \\
0.8333 \\
0.8333 \\
0.8333 \\
0.8333 \\
1.0000 \\
1.0000 \\
1.0000 \\
1.0000 \\
1.0000 \\
1.0000 \\
1.0000
\end{pmatrix}, \quad y = \begin{pmatrix}
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 \\
0.8333 \\
1.0000 \\
0 \\
0.1667 \\
0.3333 \\
0.5000 \\
0.6667 
\end{pmatrix}, \quad \text{ibo} = \begin{pmatrix}
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
49 \\
48 \\
47 \\
46 \\
45 \\
44 \\
43 \\
14 \\
21 \\
28 \\
35 \\
42 \\
8 \\
15 \\
22 \\
29 \\
36
\end{pmatrix}\]
In the following lines we describe the case of Bogner-Fox-Schmit rectangles. $lx$ represents the number of rectangles along each axis. A cruel diagram of the subject is presented in Figure (4.4). The squares are labeled in increasing order from bottom left corner to the top right corner. The number of nodes per element is set by $nbf = 16$. The nodes are labelled globally in two ways. First, they are labeled globally in increasing order from bottom left corner to the top right corner as shown in Figure (4.5). We list the nodes which relates the function values first then the x-derivative then the y-derivative and finally the mixed derivatives. Second, they are labeled globally in increasing order from bottom left corner to the top right corner as shown in Figure (4.5). We list all nodes which relates to a a point then we list all nodes which relate to the next point and so on. The x-y-coordinates is computed by subtracting COORD and stored in a real vectors $x(\cdot)$ and $y(\cdot)$. An example of a local node ordering on a rectangle is shown in Figure (4.6). This local ordering is stored in an array $ielnode(m \times nbf)$ where $m = 4 \times (lx + 1)^2$. Moreover, the boundary nodes of

<table>
<thead>
<tr>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 4.4: Rectangles labeling
the top are stored in $iboy$ and the others are stored in $ibo$.

### 4.4 Affine Mapping

After we compute the $ielnode$ array and the x-coordinate and y-coordinate for each point, each triangle element is then the image of a reference element under an affine transformation. Let $\tilde{e}$ denote the reference triangle in the parameter space with vertices $(0,0)$, $(1,0)$ and $(0,1)$ as shown in Figure (4.7). The vertex nodes of $\tilde{e}$ are ordered, as indicated, counterclockwise beginning with $(0,0)$. The vertex nodes of the $l$-th element $e_l$ in $\Omega_H$ are then found by searching through the arrays $ielnode$, $xx$ and $yy$:

\begin{align*}
k_1 &= ielnode(l, 1), \\
k_2 &= ielnode(l, 2), \\
k_3 &= ielnode(l, 3),
\end{align*}

\[(4.5)\]
Figure 4.6: Local node ordering

Figure 4.7: The reference Triangle
given the global node number of the three vertex nodes. Their coordinates are then

\[ x_{k1} = xx(k1), \quad y_{k1} = yy(k1), \]
\[ x_{k2} = xx(k2), \quad y_{k2} = yy(k2), \]
\[ x_{k3} = xx(k3), \quad y_{k3} = yy(k3). \]  

(4.6)

The affine map formed using these coordinates

\[
\begin{pmatrix}
x \\
y
\end{pmatrix} = F_{2 \times 2} \begin{pmatrix}
\xi \\
\eta
\end{pmatrix} + \begin{pmatrix}
b_1 \\
b_2
\end{pmatrix},
\]

(4.7)

where specifically,

\[
x = [x_{k2} - x_{k1}] \xi + [x_{k3} - x_{k1}] \eta + x_{k1},
\]
\[
y = [y_{k2} - y_{k1}] \xi + [y_{k3} - y_{k1}] \eta + y_{k1},
\]

(4.8)

then maps \( \tilde{e} \) to \( e_l \). The affine map for mesh shown in Figure (4.7) is

\[
x = [x_{k2} - x_{k1}] \xi + x_{k1},
\]
\[
y = [y_{k3} - y_{k1}] \eta + y_{k1}.
\]

(4.9)

In addition, it maps node (1) in \( \tilde{e} \) to node(1) (with coordinates \( (x_{k1}, y_{k1}) \)) in \( e_l \), node(2) in \( \tilde{e} \) to node(2) in \( e_l \), and node(3) in \( \tilde{e} \) to node(3) in \( e_l \). Further, edge midpoints are mapped to corresponding edge midpoints.

In rectangle element, let \( \tilde{e} \) denote the reference rectangle in the parameter space with vertices (-1,1), (1,-1), (-1,1) and (1,1) as shown in Figure (4.8). The affine map
Figure 4.8: The reference rectangle

is

\[
\begin{align*}
    x &= \frac{h}{2} (\xi - 1) + x_2, \\
y &= \frac{h}{2} (\eta - 1) + y_3,
\end{align*}
\]  

(4.10)

where

\[
\begin{align*}
x_i &= x(\text{ielnode}(l, i)), \\
y_i &= y(\text{ielnode}(l, i)), \\
h &= x_2 - x_1 = y_3 - y_1.
\end{align*}
\]
4.5 Basis Functions for Morley Element

Now, consider the construction of quadratic basis functions on the reference triangle with vertices (0,0), (1,0) and (0,1), given the reference nodes $N_1 = (0, 0), N_2 = (1, 0), N_3 = (0, 1), N_4 = (1/2, 0), N_5 = (1/2, 1/2)$ and $N_6 = (0, 1/2)$. On the reference element the basis function is of the form

$$\tilde{\phi}(\xi, \eta) = c_1 \xi^2 + c_2 \eta^2 + c_3 \xi \eta + c_4 \xi + c_5 \eta + c_6,$$

and thus the values of $\tilde{\phi}$ is specified at the three vertex nodes and $\frac{\partial \tilde{\phi}}{\partial n}$ is specified at the three mid-side point nodes. Here, $\tilde{\phi}_1$ has to satisfy

$$\tilde{\phi}_1(N_1) = 1, \quad -\frac{\partial \tilde{\phi}_1}{\partial y}(N_4) = 0, \quad (4.11)$$
$$\tilde{\phi}_1(N_2) = 0, \quad \frac{1}{\sqrt{2}} \left( \frac{\partial \tilde{\phi}_1}{\partial \xi}(N_5) + \frac{\partial \tilde{\phi}_1}{\partial \eta}(N_5) \right) = 0, \quad (4.12)$$
$$\tilde{\phi}_1(N_3) = 0, \quad \frac{\partial \tilde{\phi}_1}{\partial \xi}(N_6) = 0. \quad (4.13)$$

Also, $\tilde{\phi}_5$ has to satisfy

$$\tilde{\phi}_5(N_1) = 0, \quad -\frac{\partial \tilde{\phi}_5}{\partial \eta}(N_4) = 0, \quad (4.14)$$
$$\tilde{\phi}_5(N_2) = 0, \quad \frac{1}{\sqrt{2}} \left( \frac{\partial \tilde{\phi}_1}{\partial \xi}(N_5) + \frac{\partial \tilde{\phi}_5}{\partial \eta}(N_5) \right) = 1, \quad (4.15)$$
$$\tilde{\phi}_5(N_3) = 0, \quad \frac{\partial \tilde{\phi}_5}{\partial \xi}(N_6) = 0. \quad (4.16)$$

The coefficients of the basis function $\tilde{\phi}_i$ can be precomputed with MATHEMATICA

$$f[x_, y_]:= f1 x^2 + f2 y^2 + f3 x y + f4 x + f5 y + f6$$
fx[x_,y_] = D[f[x,y], x];
fy[x_,y_] = D[f[x,y], y];
pp1 := Solve[{f[0, 0] == 1,
f[1, 0] == 0,
f[0, 1] == 0,
-fy[1/2, 0] == 0,
-fx[0, 1/2] == 0,
fx[1/2, 1/2] + fy[1/2, 1/2] == 0 Sqrt[2]},
{f1, f2, f3, f4, f5, f6}]
v1 := {f1, f2, f3, f4, f5, f6} /. pp1
r1 := {1}.v1

pp2 := Solve[{f[0, 0] == 0,
f[1, 0] == 1,
f[0, 1] == 0,
-fy[1/2, 0] == 0,
-fx[0, 1/2] == 0,
fx[1/2, 1/2] + fy[1/2, 1/2] == 0 Sqrt[2]},
{f1, f2, f3, f4, f5, f6}]
v2 := {f1, f2, f3, f4, f5, f6} /. pp2
r2 := {1}.v2

pp3 := Solve[{f[0, 0] == 0,
f[1, 0] == 0,
f[0, 1] == 1,
-fy[1/2, 0] == 0,
-fx[0, 1/2] == 0,
fx[1/2, 1/2] + fy[1/2, 1/2] == 0 Sqrt[2]},
{f1, f2, f3, f4, f5, f6}]
v3 := {f1, f2, f3, f4, f5, f6} /. pp3
r3 := {1}.v3
{f1,f2,f3,f4,f5,f6 } ]

v3 :={f1,f2,f3,f4,f5,f6} /. pp3
r3 :={1}.v3

pp4 := Solve[ { f[0,0] == 0 ,
   f[1,0] == 0 ,
   f[0,1] == 0 ,
   -fy[1/2,0] == 1 ,
   -fx[0,1/2] == 0 ,
   fx[1/2,1/2] + fy[1/2,1/2] == 0 Sqrt[2] } ,
   {f1,f2,f3,f4,f5,f6 } ]

v4 :={f1,f2,f3,f4,f5,f6} /. pp4
r4 :={1}.v4

pp5 := Solve[ { f[0,0] == 0 ,
   f[1,0] == 0 ,
   f[0,1] == 0 ,
   -fy[1/2,0] == 0 ,
   -fx[0,1/2] == 0 ,
   fx[1/2,1/2] + fy[1/2,1/2] == 1 Sqrt[2] } ,
   {f1,f2,f3,f4,f5,f6 } ]

v5 :={f1,f2,f3,f4,f5,f6} /. pp5
r5 :={1}.v5

pp6 := Solve[ { f[0,0] == 0 ,
   f[1,0] == 0 ,
   f[0,1] == 0 ,
   -fy[1/2,0] == 0 ,
   -fx[0,1/2] == 1 ,
\[ fx[1/2,1/2] + fy[1/2,1/2] == 0 \text{ Sqrt}[2] \}, \]
\[ \{f1,f2,f3,f4,f5,f6 \} \]
\[ v6:=\{f1,f2,f3,f4,f5,f6\} /. \text{pp6} \]
\[ r6:=\{1\}.v6 \]
\[ a = \{r1,r2,r3,r4,r5,r6\} \]

The basis functions are given explicitly by

\begin{align*}
\tilde{\phi}_1(\xi, \eta) &= 1 - \xi - \eta + 2\xi\eta, \\
\tilde{\phi}_2(\xi, \eta) &= \xi/2 + \xi^2/2 + \eta/2 - \xi\eta - \eta^2/2, \\
\tilde{\phi}_3(\xi, \eta) &= \xi/2 - \xi^2/2 + \eta/2 - \xi\eta + \eta^2/2, \\
\tilde{\phi}_4(\xi, \eta) &= -\eta + \eta^2, \\
\tilde{\phi}_5(\xi, \eta) &= \{-\xi + \xi^2 - \eta + 2\xi\eta + \eta^2\}/\sqrt{2}, \\
\tilde{\phi}_6(\xi, \eta) &= -\xi + \xi^2.
\end{align*}

The local basis functions on \( e_i \) are simply the image of these under the affine map (4.9). For example, the local basis functions associated with the node \( N_i \) has the following graph. Figure (4.9) shows the graph of three basis functions. These local basis functions on adjacent triangles are combined to form a global basis functions that is continuous at the vertices of each triangle and the normal derivatives are continuous at the mid-sides. Associated with vertex node \( N_i \) in \( \Omega_H \) is the global basis function that has values 1 at \( N_i \) and zero at the remaining vertex nodes and zero normal derivatives at all midside point nodes. Associated with midside nodes \( N_j \) in \( \Omega_H \) is a global basis function that has normal derivative 1 at \( N_j \) and zero at all the remaining midside nodes and zero function value at all vertex nodes.
Figure 4.9: Local basis functions for Morley element
4.6 Basis Function for Bogner-Fox-Schmit Element

Now, consider the construction of bicubic basis functions on the reference square with vertices $N_1 = (-1, 1)$, $W_2 = (-1, 1)$, $N_3 = (-1, 1)$. On the reference element the basis function is of the form

$$\tilde{\phi}(\xi, \eta) = c_1 + c_2\eta + c_3\eta^2 + c_4\eta^3 + c_5\xi + c_6\xi\eta + c_7\eta^2 + c_8\xi^2 + c_9\xi^2\eta + c_{10}\xi^2\eta^2 + c_{11}\xi^2\eta^3 + c_{12}\xi^3 + c_{13}\xi^3\eta + c_{14}\xi^3\eta^2 + c_{15}\xi^3\eta^3,$$

and thus the values of $\tilde{\phi}$ is specified at the four vertex points and $(\tilde{\phi}_\xi, \tilde{\phi}_\eta, \tilde{f}_{\xi\eta})$ are specified at the four vertex points. Here, $\tilde{\phi}_1$ has to satisfy

$$\tilde{\phi}_1(N_1) = 1,$$
$$\tilde{\phi}_1(N_j) = 0, \quad j = 2, 3, 4,$$
$$\tilde{\phi}_{1,\xi}(N_j) = 0, \quad j = 1, 2, 4,$$
$$\tilde{\phi}_{1,\eta}(N_j) = 0, \quad j = 1, \ldots, 4,$$
$$\tilde{\phi}_{1,\xi\eta}(N_j) = 0, \quad j = 1, \ldots, 4.$$

Also, $\tilde{\phi}_{16}$ has to satisfy

$$\tilde{\phi}_5(N_j) = 0, \quad j = 1, \ldots, 4,$$
$$\tilde{\phi}_{5,\xi}(N_j) = 0, \quad j = 1, \ldots, 4,$$
$$\tilde{\phi}_{5,\eta}(N_j) = 0, \quad j = 1, \ldots, 4,$$
$$\tilde{\phi}_{5,\xi\eta}(N_j) = 0, \quad j = 1, 2, 3,$$
$$\tilde{\phi}_{5,\xi\eta}(N_4) = 1.$$
The coefficients of the basis function $\tilde{\phi}_i$ can be precomputed with MATHEMATICA

\begin{verbatim}
vv = {1, y, y^2, y^3, x, x y, x y^2, x y^3, x^2, x^2 y, x^2 y^2, x^2 y^3, x^3, x^3 y, x^3 y^2, x^3 y^3};
vc = {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16};
f[x_, y_] = vv.vc;
fx[x_, y_] = D[f[x, y], x];
fy[x_, y_] = D[f[x, y], y];
fxy[x_, y_] = D[fx[x, y], y];

pp1 := Solve[
  {f[-1, -1] == 1, fx[-1, -1] == 0, fy[-1, -1] == 0, fxy[-1, -1] == 0,
   f[1, -1] == 0, fx[1, -1] == 0, fy[1, -1] == 0, fxy[1, -1] == 0,
   f[-1, 1] == 0, fx[-1, 1] == 0, fy[-1, 1] == 0, fxy[-1, 1] == 0,
   f[1, 1] == 0, fx[1, 1] == 0, fy[1, 1] == 0, fxy[1, 1] == 0
  },
  {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16}]

v1 := {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16} /. pp1
r1 := {1}.v1

pp2 := Solve[
  {f[-1, -1] == 0, fx[-1, -1] == 0, fy[-1, -1] == 0, fxy[-1, -1] == 0,
   f[1, -1] == 1, fx[1, -1] == 0, fy[1, -1] == 0, fxy[1, -1] == 0,
   f[-1, 1] == 0, fx[-1, 1] == 0, fy[-1, 1] == 0, fxy[-1, 1] == 0,
   f[1, 1] == 0, fx[1, 1] == 0, fy[1, 1] == 0, fxy[1, 1] == 0,
   f[-1, -1] == 0, fx[-1, -1] == 0, fy[-1, -1] == 0, fxy[-1, -1] == 0,
   f[1, -1] == 0, fx[1, -1] == 0, fy[1, -1] == 0, fxy[1, -1] == 0,
   f[-1, 1] == 0, fx[-1, 1] == 0, fy[-1, 1] == 0, fxy[-1, 1] == 0,
   f[1, 1] == 0, fx[1, 1] == 0, fy[1, 1] == 0, fxy[1, 1] == 0
  },
  {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16}]

v2 := {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16} /. pp2
r2 := {1}.v2
\end{verbatim}
\[
f[1,1] = 0, \, fx[1,1] = 0, \, fy[1,1] = 0, \, fxy[1,1] = 0
\]

\[
\{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
f12,f13,f14,f15,f16 \}
\]

\[
v2 := \{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
f12,f13,f14,f15,f16 \} /. \text{pp2}
\]

\[
r2 := \{1\} . v2
\]

\[
\text{pp3 := } \text{Solve}\{ \}
\]

\[
\{ f[-1,-1] = 0, \, fx[-1,-1] = 0, \, fy[-1,-1] = 0, \, fxy[-1,-1] = 0, 
\]

\[
f[1,-1] = 0, \, fx[1,-1] = 0, \, fy[1,-1] = 0, \, fxy[1,-1] = 0, 
\]

\[
f[-1,1] = 1, \, fx[-1,1] = 0, \, fy[-1,1] = 0, \, fxy[-1,1] = 0, 
\]

\[
f[1,1] = 0, \, fx[1,1] = 0, \, fy[1,1] = 0, \, fxy[1,1] = 0
\}

\[
\{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
f12,f13,f14,f15,f16 \}
\]

\[
v3 := \{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
f12,f13,f14,f15,f16 \} /. \text{pp3}
\]

\[
r3 := \{1\} . v3
\]

\[
\text{pp4 := } \text{Solve}\{ \}
\]

\[
\{ f[-1,-1] = 0, \, fx[-1,-1] = 0, \, fy[-1,-1] = 0, \, fxy[-1,-1] = 0, 
\]

\[
f[1,-1] = 0, \, fx[1,-1] = 0, \, fy[1,-1] = 0, \, fxy[1,-1] = 0, 
\]

\[
f[-1,1] = 0, \, fx[-1,1] = 0, \, fy[-1,1] = 0, \, fxy[-1,1] = 0, 
\]

\[
f[1,1] = 1, \, fx[1,1] = 0, \, fy[1,1] = 0, \, fxy[1,1] = 0
\}

\[
\{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
f12,f13,f14,f15,f16 \}
\]
v4 := {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16} /. pp4
r4 := {1}. v4

pp5 := Solve[{
  f[-1,-1] == 0, fx[-1,-1] == 1, fy[-1,-1] == 0, fxy[-1,-1] == 0,
  f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0,
  f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0,
  f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 0
},
  {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16}]

v5 := {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16} /. pp5
r5 := {1}. v5

pp6 := Solve[{
  f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0,
  f[1,-1] == 0, fx[1,-1] == 1, fy[1,-1] == 0, fxy[1,-1] == 0,
  f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0,
  f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 0
},
  {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16}]

v6 := {f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16} /. pp6
r6 := {1}. v6

pp7 := Solve[{

pp8 := Solve[
  
  f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0,
  f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0,
  f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0,
  f[1,1] == 0, fx[1,1] == 1, fy[1,1] == 0, fxy[1,1] == 0
  
  } ,

  {f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
  f12,f13,f14,f15,f16} ]

v8 := {f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
  f12,f13,f14,f15,f16} /. pp8

r8 := {1}.v8

pp9 := Solve[
  
  f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0,
  f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0,
  f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0,
  f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 0
  
  } ,

  {f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,
  f12,f13,f14,f15,f16} ]
\[
v_9 := \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}\} \]

\[
r_9 := \{1\}. v_9
\]

\[
pp_{10} := \text{Solve}\{ \\
\quad f[-1,-1] == 0, \ f[-1,-1] == 0, \ f[-1,-1] == 0, \ fxy[-1,-1] == 0, \\
\quad f[1,-1] == 0, \ f[1,-1] == 0, \ f[1,-1] == 1, \ fxy[1,-1] == 0, \\
\quad f[-1,1] == 0, \ f[-1,1] == 0, \ f[-1,1] == 0, \ fxy[-1,1] == 0, \\
\quad f[1,1] == 0, \ f[1,1] == 0, \ f[1,1] == 0, \ fxy[1,1] == 0
\}, \\
\quad \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}\} \}
\]

\[
v_{10} := \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}\} \]

\[
r_{10} := \{1\}. v_{10}
\]

\[
pp_{11} := \text{Solve}\{ \\
\quad f[-1,-1] == 0, \ f[-1,-1] == 0, \ f[-1,-1] == 0, \ fxy[-1,-1] == 0, \\
\quad f[1,-1] == 0, \ f[1,-1] == 0, \ f[1,-1] == 0, \ fxy[1,-1] == 0, \\
\quad f[-1,1] == 0, \ f[-1,1] == 0, \ f[-1,1] == 1, \ fxy[-1,1] == 0, \\
\quad f[1,1] == 0, \ f[1,1] == 0, \ f[1,1] == 0, \ fxy[1,1] == 0
\}, \\
\quad \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}\} \}
\]

\[
v_{11} := \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11},
\]
\[ f_{12}, f_{13}, f_{14}, f_{15}, f_{16} \] / . pp11

\[ r_{11} := \{1\}.v_{11} \]

\[ pp_{12} := \text{Solve[} \{ \]
\[ f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0, \]
\[ f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0, \]
\[ f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0, \]
\[ f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 1, fxy[1,1] == 0 \}
\]

\[ v_{12} := \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, \]
\[ f_{12}, f_{13}, f_{14}, f_{15}, f_{16} \} \]

\[ r_{12} := \{1\}.v_{12} \]

\[ pp_{13} := \text{Solve[} \{ \]
\[ f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 1, \]
\[ f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0, \]
\[ f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0, \]
\[ f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 0 \}
\]

\[ v_{13} := \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, \]
\[ f_{12}, f_{13}, f_{14}, f_{15}, f_{16} \} \]

\[ r_{13} := \{1\}.v_{13} \]

\[ pp_{14} := \text{Solve[} \{ \]
\[ f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0, \]
\[ f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 1, \\
 f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0, \\
 f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 0 \\
\] , \\
\{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11, \\
f12,f13,f14,f15,f16 \} \]

\[ v14 := \{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11, \\
f12,f13,f14,f15,f16 \} /. \text{pp14} \]

\[ r14 := \{1\}.v14 \]

\[ \text{pp15} := \text{Solve[ } \{ \\
 f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0, \\
 f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0, \\
 f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 1, \\
 f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 0 \\
\} , \\
\{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11, \\
f12,f13,f14,f15,f16 \} \} \]

\[ v15 := \{f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11, \\
f12,f13,f14,f15,f16 \} /. \text{pp15} \]

\[ r15 := \{1\}.v15 \]

\[ \text{pp16} := \text{Solve[ } \{ \\
 f[-1,-1] == 0, fx[-1,-1] == 0, fy[-1,-1] == 0, fxy[-1,-1] == 0, \\
 f[1,-1] == 0, fx[1,-1] == 0, fy[1,-1] == 0, fxy[1,-1] == 0, \\
 f[-1,1] == 0, fx[-1,1] == 0, fy[-1,1] == 0, fxy[-1,1] == 0, \\
 f[1,1] == 0, fx[1,1] == 0, fy[1,1] == 0, fxy[1,1] == 1 \\
\} , \\
\] \]
\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}\}

v_{16} := \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}\} /\!. \ p_{p16}

r_{16} := \{1\}. v_{16}

a = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15}, r_{16}\};

\phi[1, x_, y_] = v_{v}. r_1;
\phi[2, x_, y_] = v_{v}. r_2;
\phi[3, x_, y_] = v_{v}. r_3;
\phi[4, x_, y_] = v_{v}. r_4;
\phi[5, x_, y_] = v_{v}. r_5;
\phi[6, x_, y_] = v_{v}. r_6;
\phi[7, x_, y_] = v_{v}. r_7;
\phi[8, x_, y_] = v_{v}. r_8;
\phi[9, x_, y_] = v_{v}. r_9;
\phi[10, x_, y_] = v_{v}. r_{10};
\phi[11, x_, y_] = v_{v}. r_{11};
\phi[12, x_, y_] = v_{v}. r_{12};
\phi[13, x_, y_] = v_{v}. r_{13};
\phi[14, x_, y_] = v_{v}. r_{14};
\phi[15, x_, y_] = v_{v}. r_{15};
\phi[16, x_, y_] = v_{v}. r_{16};
The basis functions are given explicitly by

\[
\begin{align*}
\tilde{\phi}_1(\xi, \eta) &= (-1 + \xi)^2 (2 + \xi) (-1 + \eta)^2 (2 + \eta), \\
\tilde{\phi}_2(\xi, \eta) &= -((1 + \xi)^2 (1 + \eta)^2 (2 + \eta)), \\
\tilde{\phi}_3(\xi, \eta) &= -((1 + \xi)^2 (2 + \xi) (-2 + \eta) (1 + \eta)^2), \\
\tilde{\phi}_4(\xi, \eta) &= (-2 + \xi) (1 + \xi)^2 (-2 + \eta) (1 + \eta)^2, \\
\tilde{\phi}_5(\xi, \eta) &= (-1 + \xi)^2 (1 + \xi) (-1 + \eta)^2 (2 + \eta), \\
\tilde{\phi}_6(\xi, \eta) &= (-1 + \xi) (1 + \xi)^2 (2 + \eta) (1 + \eta)^2, \\
\tilde{\phi}_7(\xi, \eta) &= -((1 + \xi)^2 (1 + \xi) (-2 + \eta) (1 + \eta)^2), \\
\tilde{\phi}_8(\xi, \eta) &= -((1 + \xi)^2 (-2 + \eta) (1 + \eta)^2), \\
\tilde{\phi}_9(\xi, \eta) &= (-1 + \xi)^2 (2 + \xi) (-1 + \eta)^2 (1 + \eta), \\
\tilde{\phi}_{10}(\xi, \eta) &= -((2 + \xi) (1 + \xi)^2 (-1 + \eta)^2 (1 + \eta)), \\
\tilde{\phi}_{11}(\xi, \eta) &= (-1 + \xi)^2 (2 + \xi) (-1 + \eta) (1 + \eta)^2, \\
\tilde{\phi}_{12}(\xi, \eta) &= -((2 + \xi) (1 + \xi)^2 (-1 + \eta) (1 + \eta)^2), \\
\tilde{\phi}_{13}(\xi, \eta) &= (-1 + \xi)^2 (1 + \xi) (-1 + \eta)^2 (1 + \eta), \\
\tilde{\phi}_{14}(\xi, \eta) &= (-1 + \xi) (1 + \xi)^2 (-1 + \eta)^2 (1 + \eta), \\
\tilde{\phi}_{15}(\xi, \eta) &= (-1 + \xi)^2 (1 + \xi) (-1 + \eta) (1 + \eta)^2, \\
\tilde{\phi}_{16}(\xi, \eta) &= (-1 + \xi) (1 + \xi)^2 (-1 + \eta) (1 + \eta)^2.
\end{align*}
\]

The local basis functions on \( e_l \) are simply the image of these under the affine map (4.10). Figures (4.10, 4.11, 4.12) show the graphs of these functions.
Figure 4.10: Local basis functions for Bogner-Fox-Schmit elements ($\tilde{\phi}_1 \cdots \tilde{\phi}_6$)
Figure 4.11: Local basis functions for Bogner-Fox-Schmit elements ($\tilde{\phi}_7 \cdots \tilde{\phi}_{12}$)
Figure 4.12: Local basis functions for Bogner-Fox-Schmit elements ($\tilde{\phi}_{13} \cdots \tilde{\phi}_{16}$)
4.7 Discretizing the Continuous Problem

Let \( \{ \phi_i : i = 1, \ldots, m \} \) denote the global basis functions, defined on \( \Omega_H \). We want to approximate \( \psi(x, y) \) by a function \( \psi^H \) in the space \( S^H = \text{span} \{ \phi_i : i = 1, \ldots, m \} \). As \( \psi^H(x, y) \) lies in \( S^H \) we may expand:

\[
\psi^H(x, y) = \sum_{j=1}^{m} c_j \phi_j(x, y). \tag{4.18}
\]

Now, the problem is computing the coefficients \( c_j \)'s in (4.18). Taking the variation of (4.2) over \( S^H \) gives the variational equations which \( \psi^H(x, y) \) satisfies:

\[
a(\psi^H, \phi^H) + b(\psi^H, \psi^H, \phi^H) = (f, \text{curl} \ \phi^H), \quad \text{for all} \quad \phi^H \in S^H. \tag{4.19}
\]

Setting \( \phi^H = \phi_i(x, y) \) in (4.19) and substituting (4.18) into (4.19) yields

\[
\sum_{j=1}^{m} c_j a(\phi_j, \phi_i) + \sum_{j=1}^{m} \sum_{k=1}^{m} c_j c_k b(\phi_j, \phi_k, \phi_i) = (f, \text{curl} \ \phi_i), \quad i = 1, \ldots, m. \tag{4.20}
\]

Alternately, we may represent (4.20) as a \( m \times m \) nonlinear system:

\[
G(C) = AC + \begin{bmatrix}
C^T B^{(1)} C \\
C^T B^{(2)} C \\
\vdots \\
C^T B^{(m)} C
\end{bmatrix} = L, \tag{4.21}
\]
where

\[ A_{ij} = a(\phi_j, \phi_i), \]
\[ B_{jk}^{(i)} = b(\phi_j, \phi_k, \phi_i), \]
\[ L_i = (f, \text{curl } \phi_i), \]

\[ C = \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_{m-1} \\ c_m \end{pmatrix} \]

In another form, we may rewrite equation (4.21) as:

\[ F(C) = 0, \]
where \[ F(C) = G(C) - L. \] (4.22)

4.8 The Solution of \( F(C) = 0 \)

In this section, we discuss how to solve the resulting nonlinear system of equations.

We use the Newton’s method for nonlinear systems.
Consider the Newton’s iteration

\[ c^{n+1} = c^n - J(c^n)^{-1} F(c^n), \]

where \( F : \mathbb{R}^m \rightarrow \mathbb{R}^m \) is defined in (b4.22) and \( J(c) \) is defined by

\[
J(c) = \begin{bmatrix}
\frac{\partial f_1(c)}{\partial x_1} & \cdots & \frac{\partial f_1(c)}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_n(c)}{\partial x_1} & \cdots & \frac{\partial f_n(c)}{\partial x_n}
\end{bmatrix}
\]

It is known that the \((n+1)\)-th error is proportional to the square of the \(n\)-th error. So that the convergence is very rapid once the errors are small. So we need to pick a good initial guess to solve the nonlinear system of equations. Choosing \( c^0 \) to be the zero vector means that the first iteration will give us a solution to a discretized biharmonic equation. Hence, \( c^1 \) is a good choice for the initial guess.

One requirement which may be imposed upon any iterative method is that it be norm-reducing in sense that

\[ \| F(c^{n+1}) \| \leq \| F(c^{n+1}) \|, \quad k = 0, 1, \cdots, \]

holds in some norm. The Newton method does not necessarily satisfy this requirement. One simple modification of Newton method is adding the following IF state-
If \((\| F(c^{n+1}) \| \geq \| F(c^n) \|)\) then
\[ c^{n+1} = c^n - \frac{1}{2} J(c^n)^{-1} F(c^n) \]

else
\[ c^{n+1} = c^n - J(c^n)^{-1} F(c^n) \]

end if

Using the Newton method involves the computing of the Jacobian matrix which will be described in the next section.

### 4.9 Computation of the Jacobian

The elements of the Jacobian matrix are given by

\[
J_{ij} = \frac{\partial F_i}{\partial C_i} = A_{ij} + \sum_{k=1}^{m} C_k (b_{kj}^{(i)} + b_{jk}^{(i)}),
\]

\[
= a(\phi_j, \phi_i) + \sum_{k=1}^{m} C_k \{ b(\phi_k, \phi_j, \phi_i) + b(\phi_j, \phi_k, \phi_i) \}. \tag{4.23}
\]

We assemble \(A_{ij}\), \(b_{kj}^{(i)}\) and \(b_{jk}^{(i)}\) element by element as

\[
A_{ij} = \sum_{l=1}^{nelem} A_{ij}^{(l)},
\]

\[
b_{kj}^{(i)} = \sum_{l=1}^{nelem} \{ b_{kj}^{(i)} \}^{(l)},
\]
where

\[
A_{ij}^{(l)} = \int_{e_l} \{ \phi_{j,xx} \phi_{i,xx} + 2 \phi_{j,xy} \phi_{i,xy} + \phi_{j,yy} \phi_{i,yy} \} \, dxdy,
\]

\[
(b_{kj}^{(i)})^{(l)} = \int_{e_l} \{ (\phi_{k,y} \phi_{j,xx} - \phi_{k,x} \phi_{j,xy}) \phi_{i,x} - (\phi_{k,x} \phi_{j,yy} - \phi_{k,y} \phi_{j,xy}) \phi_{i,y} \} \, dxdy.
\]

We assemble \( A_{ij}^{(l)} \), \((b_{kj}^{(i)})^{(l)}\) and \((b_{jk}^{(i)})^{(l)}\) by mapping the \( e_l \) back to the reference element \( \tilde{e} \) as follows. Given \( e_l \) we look up the global node numbers in \( ielnode \) array and then find x-y coordinates of the vertex nodes in \( xx, yy \) arrays. With these we calculate the affine map of \( e_l \) to \( \tilde{e} \) (i.e)

\[
\begin{pmatrix}
    x \\
    y
\end{pmatrix} = \begin{pmatrix}
    x_2 - x_1 & 0 \\
    0 & y_3 - y_1
\end{pmatrix} \begin{pmatrix}
    \eta \\
    \xi
\end{pmatrix} + \begin{pmatrix}
    x_1 \\
    y_1
\end{pmatrix}.
\]

(4.24)

The \( f_{ij} \)'s and \( d_i \)'s are determined from the x-y coordinates of the vertex nodes of \( e_l \).

Using (4.24) we now make a change of variables reducing \( A_{ij}^{(l)} \), \((b_{kj}^{(i)})^{t}\) and \((b_{jk}^{(i)})^{t}\) to an
integral over \( \tilde{e} \). After applying the chain rule, the final results for the case of Morley elements are:

\[
A_{ij}^{(l)} = h^{-2} \int_{\tilde{e}} \tilde{\phi}_{i,\xi} \tilde{\phi}_{j,\xi} + 2 \tilde{\phi}_{i,\xi} \tilde{\phi}_{j,\eta} + \tilde{\phi}_{i,\eta} \tilde{\phi}_{j,\eta} = h^{-2} \tilde{A}_{ij},
\]

\[
(b_{jk}^{(i)})^{(l)} = h^{-2} \int_{\tilde{e}} \{ (\tilde{\phi}_{k,\eta} \tilde{\phi}_{j,\xi} - \tilde{\phi}_{k,\xi} \tilde{\phi}_{j,\eta}) \tilde{\phi}_{i,\xi} - (\tilde{\phi}_{k,\xi} \tilde{\phi}_{j,\eta} - \tilde{\phi}_{k,\eta} \tilde{\phi}_{j,\xi}) \tilde{\phi}_{i,\eta} \} = h^{-2} \tilde{b}_{jk}^{(i)},
\]

where \( h = x_2 - x_1 = y_3 - y_1 \). In our case we have a regular finite element. In Bogner-Fox-Schmit, we have:

\[
A_{ij}^{(l)} = \frac{4}{h^2} \tilde{A}_{ij},
\]

\[
(b_{jk}^{(i)})^{(l)} = \frac{4}{h^2} \tilde{b}_{jk}^{(i)},
\]

\[
(b_{kj}^{(i)})^{(l)} = \frac{4}{h^2} \tilde{b}_{kj}^{(i)},
\]

where \( h = 2/lx \). Hence, \( A_{ij}^{(l)} \), \( (b_{jk}^{(i)})^{(l)} \) and \( (b_{kj}^{(i)})^{(l)} \) are the same for any element \( e_i \) in the finite element space. The entries of the matrix \( A \) and the three-dimension array \( b \) for the case of Morley triangles can be computed exactly with MATHEMATICA

\[
f[x_,y_]:=f1 x^2+f2 y^2+f3 x y+f4 x+f5 y+f6
\]

\[
fx[x_,y_]=D[f[x,y],x];
\]

\[
fy[x_,y_]=D[f[x,y],y];
\]

\[
pp1:=\text{Solve[} \{ f[0,0] == 1 , f[1,0] == 0 , f[0,1] == 0 , -fy[1/2,0] == 0 , -fx[0,1/2] == 0 ,
\]
\[ fx[1/2,1/2] + fy[1/2,1/2] == 0 \sqrt{2} \] ,
\{ f1,f2,f3,f4,f5,f6 \} 
\]
v1:={f1,f2,f3,f4,f5,f6} /. pp1
r1:={1}.v1

pp2:=Solve[ { f[0,0] == 0 ,
          f[1,0] == 1 ,
          f[0,1] == 0 ,
          -fy[1/2,0] == 0 ,
          -fx[0,1/2] == 0 ,
          fx[1/2,1/2] + fy[1/2,1/2] == 0 \sqrt{2} } ,
\{ f1,f2,f3,f4,f5,f6 \} ]
v2:={f1,f2,f3,f4,f5,f6} /. pp2
r2:={1}.v2

pp3:=Solve[ { f[0,0] == 0 ,
          f[1,0] == 0 ,
          f[0,1] == 1 ,
          -fy[1/2,0] == 0 ,
          -fx[0,1/2] == 0 ,
          fx[1/2,1/2] + fy[1/2,1/2] == 0 \sqrt{2} } ,
\{ f1,f2,f3,f4,f5,f6 \} ]
v3:={f1,f2,f3,f4,f5,f6} /. pp3
r3:={1}.v3

pp4:=Solve[ { f[0,0] == 0 ,
          f[1,0] == 0 ,
          f[0,1] == 0 ,
          -fy[1/2,0] == 1 ,
          -fx[0,1/2] == 0 ,
          fx[1/2,1/2] + fy[1/2,1/2] == 0 \sqrt{2} } ,
\{ f1,f2,f3,f4,f5,f6 \} ]
-fx[0,1/2] == 0 ,
fx[1/2,1/2] + fy[1/2,1/2] == 0 Sqrt[2] }
{f1,f2,f3,f4,f5,f6 }

v4:={f1,f2,f3,f4,f5,f6} /. pp4
r4:={1}.v4

pp5:=Solve[ { f[0,0] == 0 ,
f[1,0] == 0 ,
f[0,1] == 0 ,
-fy[1/2,0] == 0 ,
-fx[0,1/2] == 0 ,
fx[1/2,1/2] + fy[1/2,1/2] == 1 Sqrt[2] } ,
{f1,f2,f3,f4,f5,f6 }

v5:={f1,f2,f3,f4,f5,f6} /. pp5
r5:={1}.v5

pp6:=Solve[ { f[0,0] == 0 ,
f[1,0] == 0 ,
f[0,1] == 0 ,
-fy[1/2,0] == 0 ,
-fx[0,1/2] == 1 ,
fx[1/2,1/2] + fy[1/2,1/2] == 0 Sqrt[2] } ,
{f1,f2,f3,f4,f5,f6 }

v6:={f1,f2,f3,f4,f5,f6} /. pp6
r6:={1}.v6

a = {r1,r2,r3,r4,r5,r6}
e1=f[x,y] /. pp1 ;
phi[1,x_,y_] = {1}.e1
e2 = f[x, y] /. pp2;
phi[2, x_, y_] = {1}. e2

e3 = f[x, y] /. pp3;
phi[3, x_, y_] = {1}. e3

e4 = f[x, y] /. pp4;
phi[4, x_, y_] = {1}. e4

e5 = f[x, y] /. pp5;
phi[5, x_, y_] = {1}. e5

e6 = f[x, y] /. pp6;
phi[6, x_, y_] = {1}. e6

phix[1, x_, y_] = D[phi[1, x, y], x];
phiy[1, x_, y_] = D[phi[1, x, y], y];
phix[2, x_, y_] = D[phi[2, x, y], x];
phiy[2, x_, y_] = D[phi[2, x, y], y];
phix[3, x_, y_] = D[phi[3, x, y], x];
phiy[3, x_, y_] = D[phi[3, x, y], y];
phix[4, x_, y_] = D[phi[4, x, y], x];
phiy[4, x_, y_] = D[phi[4, x, y], y];
phix[5, x_, y_] = D[phi[5, x, y], x];
phiy[5, x_, y_] = D[phi[5, x, y], y];
phix[6, x_, y_] = D[phi[6, x, y], x];
phiy[6, x_, y_] = D[phi[6, x, y], y];

phixy[1, x_, y_] = D[phix[1, x, y], y];
phixy[2, x_, y_] = D[phix[2, x, y], y];
\begin{verbatim}
phixy[3,x_,y_] = D[ phix[3,x,y] , y ];
phixy[4,x_,y_] = D[ phix[4,x,y] , y ];
phixy[5,x_,y_] = D[ phix[5,x,y] , y ];
phixy[6,x_,y_] = D[ phix[6,x,y] , y ];
phixx[1,x_,y_] = D[ phix[1,x,y] , x ];
phixx[2,x_,y_] = D[ phix[2,x,y] , x ];
phixx[3,x_,y_] = D[ phix[3,x,y] , x ];
phixx[4,x_,y_] = D[ phix[4,x,y] , x ];
phixx[5,x_,y_] = D[ phix[5,x,y] , x ];
phixx[6,x_,y_] = D[ phix[6,x,y] , x ];
phiyy[1,x_,y_] = D[ phiy[1,x,y] , y ];
phiyy[2,x_,y_] = D[ phiy[2,x,y] , y ];
phiyy[3,x_,y_] = D[ phiy[3,x,y] , y ];
phiyy[4,x_,y_] = D[ phiy[4,x,y] , y ];
phiyy[5,x_,y_] = D[ phiy[5,x,y] , y ];
phiyy[6,x_,y_] = D[ phiy[6,x,y] , y ];
lphi[1,x_,y_] = phixx[1,x,y] + phiyy[1,x,y];
lphi[2,x_,y_] = phixx[2,x,y] + phiyy[2,x,y];
lphi[3,x_,y_] = phixx[3,x,y] + phiyy[3,x,y];
lphi[4,x_,y_] = phixx[4,x,y] + phiyy[4,x,y];
lphi[5,x_,y_] = phixx[5,x,y] + phiyy[5,x,y];
lphi[6,x_,y_] = phixx[6,x,y] + phiyy[6,x,y];

Table[ {phi[i,0,0],phi[i,1,0],phi[i,0,1],-phiy[i,1/2,0],
(phix[i,1/2,1/2]+phiy[i,1/2,1/2])/Sqrt[2],-phix[i,0,1/2]},{i,6}]  
g[i_,j_] := phixx[i,x,y]*phixx[j,x,y]
\end{verbatim}
\[ + 2\, \phi_{ixy}[i,x,y]\phi_{ixy}[j,x,y] \]
\[ + \phi_{iyy}[i,x,y]\phi_{iyy}[j,x,y] \]
\[
A = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[g[i,j],\{y,0,x\}],\{x,0,1\}\right],\{i,6\},\{j,6\}\right] 
\[
gg[i_-,j_-,k_-] := (\phi_{iy}[k,x,y]\phi_{ixy}[j,x,y] 
- \phi_{ix}[k,x,y]\phi_{iyy}[j,x,y])*\phi_{i}[i,x,y] 
- (\phi_{ix}[k,x,y]\phi_{ixy}[j,x,y] 
- \phi_{iy}[k,x,y]\phi_{ixx}[j,x,y])*\phi_{ix}[i,x,y] \right) 
\[
b1 = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[gg[1,j,k],\{y,0,x\}],\{x,0,1\}\right],\{j,6\},\{k,6\}\right] 
\[
b2 = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[gg[2,j,k],\{y,0,x\}],\{x,0,1\}\right],\{j,6\},\{k,6\}\right] 
\[
b3 = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[gg[3,j,k],\{y,0,x\}],\{x,0,1\}\right],\{j,6\},\{k,6\}\right] 
\[
b4 = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[gg[4,j,k],\{y,0,x\}],\{x,0,1\}\right],\{j,6\},\{k,6\}\right] 
\[
b5 = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[gg[5,j,k],\{y,0,x\}],\{x,0,1\}\right],\{j,6\},\{k,6\}\right] 
\[
b6 = \text{Table}\left[\text{Integrate}\left[\text{Integrate}[gg[6,j,k],\{y,0,x\}],\{x,0,1\}\right],\{j,6\},\{k,6\}\right] 
\]
The entries are given explicitly by

$$\tilde{A} = \begin{pmatrix}
4.000000 & -2.000000 & -2.000000 & .000000 & 2.828430 & .000000 \\
2.000000 & 2.000000 & .000000 & -1.000000 & -1.1414210 & 1.000000 \\
2.000000 & .000000 & 2.000000 & 1.000000 & -1.1414210 & -1.000000 \\
.000000 & -1.000000 & 1.000000 & 2.000000 & 1.1414210 & .000000 \\
2.828430 & -1.1414210 & -1.1414210 & 1.1414210 & 4.000000 & 1.1414210 \\
.000000 & 1.000000 & -1.000000 & .000000 & 1.1414210 & 2.000000
\end{pmatrix}$$

$$\tilde{B}^{(1)} = \begin{pmatrix}
.000000 & .000000 & .000000 & .000000 & .4714045 & .000000 \\
.000000 & .1666667 & -.1666667 & .1666667 & .000000 & .1666667 \\
.000000 & -1.1666667 & .1666667 & -.1666667 & -.4714045 & -.1666667 \\
.000000 & -.3333333 & .3333333 & .000000 & -.4714045 & -.3333333 \\
.000000 & -.2357023 & .2357023 & .2357023 & .000000 & -.2357023 \\
.000000 & .000000 & .000000 & .3333333 & .000000 & .000000
\end{pmatrix}$$

$$\tilde{B}^{(2)} = \begin{pmatrix}
.000000 & -.3333333 & .3333333 & .000000 & -1.178511 & -.3333333 \\
.1666667 & -.2500000 & .0833333 & -.1666667 & .5892556 & .000000 \\
-.1666667 & .5833333 & -.4166667 & .1666667 & .5892556 & .3333333 \\
.000000 & .4166667 & -.4166667 & .000000 & .5892556 & .3333333 \\
.2357023 & -.2357023 & .000000 & -.2357023 & .000000 & .000000 \\
.3333333 & -.4166667 & .0833333 & -.3333333 & .5892556 & .000000
\end{pmatrix}$$
\[
\hat{B}^{(3)} = \begin{pmatrix}
.000000 & .333333 & -.333333 & .000000 & .7071068 & .333333 \\
-.166667 & .083333 & .083333 & .000000 & -.5892556 & -.166667 \\
.166667 & -.416667 & .250000 & .000000 & -.1178511 & -.166667 \\
.000000 & -.083333 & .083333 & .000000 & -.1178511 & .000000 \\
-.2357023 & .4714045 & -.2357023 & .000000 & .000000 & .2357023 \\
-.333333 & .416667 & -.083333 & .000000 & -.5892556 & .000000 \\
\end{pmatrix}
\]

\[
\hat{B}^{(4)} = \begin{pmatrix}
.000000 & .000000 & .000000 & .333333 & .000000 & .000000 \\
.166667 & -.166667 & .000000 & -.166667 & .000000 & .000000 \\
-.166667 & .166667 & .000000 & -.166667 & .000000 & .000000 \\
-.333333 & .333333 & .000000 & .000000 & .000000 & .000000 \\
-.2357023 & .2357023 & .000000 & .2357023 & .000000 & .000000 \\
.000000 & .000000 & .000000 & .000000 & .000000 & .000000 \\
\end{pmatrix}
\]

\[
\hat{B}^{(5)} = \begin{pmatrix}
.4714045 & -1.1785113 & .7071068 & .000000 & .000000 & -.4714045 \\
.000000 & .5892556 & -.5892556 & .000000 & .8333333 & .4714045 \\
-.4714045 & .5892556 & -.1178511 & .000000 & -.8333333 & .000000 \\
.000000 & -.5892556 & .5892556 & .000000 & -.8333333 & -.4714045 \\
.666667 & -1.666667 & 1.000000 & .000000 & .000000 & -.666667 \\
.4714045 & -.5892556 & .1178511 & .000000 & .8333333 & .000000 \\
\end{pmatrix}
\]
\[
\hat{B}^{(6)} = \begin{pmatrix}
.000000 & -.333333 & .333333 & .000000 & -.4714045 & -.333333 \\
.166667 & .000000 & -.166667 & .000000 & .4714045 & .166667 \\
-.166667 & .333333 & -.166667 & .000000 & .000000 & .166667 \\
.000000 & .000000 & .000000 & .000000 & .000000 & .000000 \\
.2357023 & -.4714045 & .2357023 & .000000 & .000000 & -.2357023 \\
.333333 & -.333333 & .000000 & .000000 & .4714045 & .000000 \\
\end{pmatrix}
\]

In Bogner-Fox-Schmit, the entries of the matrix $A$ and the three-dimensional array can be computed exactly with MATHEMATICA

\[
\text{phix}[1,x_,y_] = \text{D}[\phi[1,x,y], x];
\]
\[
\text{phiy}[1,x_,y_] = \text{D}[\phi[1,x,y], y];
\]
\[
\text{phix}[2,x_,y_] = \text{D}[\phi[2,x,y], x];
\]
\[
\text{phiy}[2,x_,y_] = \text{D}[\phi[2,x,y], y];
\]
\[
\text{phix}[3,x_,y_] = \text{D}[\phi[3,x,y], x];
\]
\[
\text{phiy}[3,x_,y_] = \text{D}[\phi[3,x,y], y];
\]
\[
\text{phix}[4,x_,y_] = \text{D}[\phi[4,x,y], x];
\]
\[
\text{phiy}[4,x_,y_] = \text{D}[\phi[4,x,y], y];
\]
\[
\text{phix}[5,x_,y_] = \text{D}[\phi[5,x,y], x];
\]
\[
\text{phiy}[5,x_,y_] = \text{D}[\phi[5,x,y], y];
\]
\[
\text{phix}[6,x_,y_] = \text{D}[\phi[6,x,y], x];
\]
\[
\text{phiy}[6,x_,y_] = \text{D}[\phi[6,x,y], y];
\]
\[
\text{phix}[7,x_,y_] = \text{D}[\phi[7,x,y], x];
\]
\[
\text{phiy}[7,x_,y_] = \text{D}[\phi[7,x,y], y];
\]
\[
\text{phix}[8,x_,y_] = \text{D}[\phi[8,x,y], x];
\]
\[
\text{phiy}[8,x_,y_] = \text{D}[\phi[8,x,y], y];
\]
\[
\text{phix}[9,x_,y_] = \text{D}[\phi[9,x,y], x];
\]
\begin{verbatim}
phiy[9,x_,y_] = D[ phi[9,x,y] , y ];
phix[10,x_,y_] = D[ phi[10,x,y] , x ];
phiy[10,x_,y_] = D[ phi[10,x,y] , y ];
phix[11,x_,y_] = D[ phi[11,x,y] , x ];
phiy[11,x_,y_] = D[ phi[11,x,y] , y ];
phix[12,x_,y_] = D[ phi[12,x,y] , x ];
phiy[12,x_,y_] = D[ phi[12,x,y] , y ];
phix[13,x_,y_] = D[ phi[13,x,y] , x ];
phiy[13,x_,y_] = D[ phi[13,x,y] , y ];
phix[14,x_,y_] = D[ phi[14,x,y] , x ];
phiy[14,x_,y_] = D[ phi[14,x,y] , y ];
phix[15,x_,y_] = D[ phi[15,x,y] , x ];
phiy[15,x_,y_] = D[ phi[15,x,y] , y ];
phix[16,x_,y_] = D[ phi[16,x,y] , x ];
phiy[16,x_,y_] = D[ phi[16,x,y] , y ];

phixy[1,x_,y_] = D[ phix[1,x,y] , y ];
phixy[2,x_,y_] = D[ phix[2,x,y] , y ];
phixy[3,x_,y_] = D[ phix[3,x,y] , y ];
phixy[4,x_,y_] = D[ phix[4,x,y] , y ];
phixy[5,x_,y_] = D[ phix[5,x,y] , y ];
phixy[6,x_,y_] = D[ phix[6,x,y] , y ];
phixy[7,x_,y_] = D[ phix[7,x,y] , y ];
phixy[8,x_,y_] = D[ phix[8,x,y] , y ];
phixy[9,x_,y_] = D[ phix[9,x,y] , y ];
phixy[10,x_,y_] = D[ phix[10,x,y] , y ];
\end{verbatim}
phixy[11, x_, y_] = D[ phix[11, x, y], y ];
phixy[12, x_, y_] = D[ phix[12, x, y], y ];
phixy[13, x_, y_] = D[ phix[13, x, y], y ];
phixy[14, x_, y_] = D[ phix[14, x, y], y ];
phixy[15, x_, y_] = D[ phix[15, x, y], y ];
phixy[16, x_, y_] = D[ phix[16, x, y], y ];

phixx[1, x_, y_] = D[ phix[1, x, y], x ];
phixx[2, x_, y_] = D[ phix[2, x, y], x ];
phixx[3, x_, y_] = D[ phix[3, x, y], x ];
phixx[4, x_, y_] = D[ phix[4, x, y], x ];
phixx[5, x_, y_] = D[ phix[5, x, y], x ];
phixx[6, x_, y_] = D[ phix[6, x, y], x ];
phixx[7, x_, y_] = D[ phix[7, x, y], x ];
phixx[8, x_, y_] = D[ phix[8, x, y], x ];
phixx[9, x_, y_] = D[ phix[9, x, y], x ];
phixx[10, x_, y_] = D[ phix[10, x, y], x ];
phixx[11, x_, y_] = D[ phix[11, x, y], x ];
phixx[12, x_, y_] = D[ phix[12, x, y], x ];
phixx[13, x_, y_] = D[ phix[13, x, y], x ];
phixx[14, x_, y_] = D[ phix[14, x, y], x ];
phixx[15, x_, y_] = D[ phix[15, x, y], x ];
phixx[16, x_, y_] = D[ phix[16, x, y], x ];

phiyy[1, x_, y_] = D[ phiy[1, x, y], y ];
phiyy[2, x_, y_] = D[ phiy[2, x, y], y ];
phiyy[3,x_,y_] = D[ phiy[3,x,y] , y ];
phiyy[4,x_,y_] = D[ phiy[4,x,y] , y ];
phiyy[5,x_,y_] = D[ phiy[5,x,y] , y ];
phiyy[6,x_,y_] = D[ phiy[6,x,y] , y ];
phiyy[7,x_,y_] = D[ phiy[7,x,y] , y ];
phiyy[8,x_,y_] = D[ phiy[8,x,y] , y ];
phiyy[9,x_,y_] = D[ phiy[9,x,y] , y ];
phiyy[10,x_,y_] = D[ phiy[10,x,y] , y ];
phiyy[11,x_,y_] = D[ phiy[11,x,y] , y ];
phiyy[12,x_,y_] = D[ phiy[12,x,y] , y ];
phiyy[13,x_,y_] = D[ phiy[13,x,y] , y ];
phiyy[14,x_,y_] = D[ phiy[14,x,y] , y ];
phiyy[15,x_,y_] = D[ phiy[15,x,y] , y ];
phiyy[16,x_,y_] = D[ phiy[16,x,y] , y ];

1phi[1,x_,y_] = phixx[1,x,y] + phiyy[1,x,y];
1phi[2,x_,y_] = phixx[2,x,y] + phiyy[2,x,y];
1phi[3,x_,y_] = phixx[3,x,y] + phiyy[3,x,y];
1phi[4,x_,y_] = phixx[4,x,y] + phiyy[4,x,y];
1phi[5,x_,y_] = phixx[5,x,y] + phiyy[5,x,y];
1phi[6,x_,y_] = phixx[6,x,y] + phiyy[6,x,y];
1phi[7,x_,y_] = phixx[7,x,y] + phiyy[7,x,y];
1phi[8,x_,y_] = phixx[8,x,y] + phiyy[8,x,y];
1phi[9,x_,y_] = phixx[9,x,y] + phiyy[9,x,y];
1phi[10,x_,y_] = phixx[10,x,y] + phiyy[10,x,y];
1phi[11,x_,y_] = phixx[11,x,y] + phiyy[11,x,y];
\[\text{lphi}[12, x\_ y\_] = \text{phixx}[12, x, y] + \text{phiyy}[12, x, y];\]

\[\text{lphi}[13, x\_ y\_] = \text{phixx}[13, x, y] + \text{phiyy}[13, x, y];\]

\[\text{lphi}[14, x\_ y\_] = \text{phixx}[14, x, y] + \text{phiyy}[14, x, y];\]

\[\text{lphi}[15, x\_ y\_] = \text{phixx}[15, x, y] + \text{phiyy}[15, x, y];\]

\[\text{lphi}[16, x\_ y\_] = \text{phixx}[16, x, y] + \text{phiyy}[16, x, y];\]

\[\text{g}[i\_ j\_] := \text{phixx}[i, x, y]*\text{phixx}[j, x, y] + 2* \text{phixy}[i, x, y]*\text{phixy}[j, x, y] + \text{phiyy}[i, x, y]*\text{phiyy}[j, x, y] ;\]

\[\text{A} = \text{Table}[\text{Integrate}[\text{Integrate}[\text{g}[i, j], \{y, -1, 1\}], \{x, -1, 1\}], \{i, 16\}, \{j, 16\}] ;\]

\[\text{gg}[i\_ j\_ k\_] := (\text{phiy}[k, x, y]*\text{phixy}[j, x, y] - \text{phix}[k, x, y]*\text{phiyy}[j, x, y])\*\text{phiy}[i, x, y] - (\text{phix}[k, x, y]*\text{phixy}[j, x, y] - \text{phiy}[k, x, y]*\text{phixx}[j, x, y])\*\text{phix}[i, x, y] ;\]

\[\text{b1} = \text{Table}[\text{Integrate}[\text{Integrate}[\text{gg}[1, j, k], \{y, -1, 1\}], \{x, -1, 1\}], \{j, 16\}, \{k, 16\}] ;\]

\[\text{b2} = \text{Table}[\text{Integrate}[\text{Integrate}[\text{gg}[2, j, k], \{y, -1, 1\}], \{x, -1, 1\}], \{j, 16\}, \{k, 16\}] ;\]

\[\text{b3} = \text{Table}[\text{Integrate}[\text{Integrate}[\text{gg}[3, j, k], \{y, -1, 1\}], \{x, -1, 1\}], \{j, 16\}, \{k, 16\}] ;\]

\[\text{b4} = \text{Table}[\text{Integrate}[\text{Integrate}[\text{gg}[4, j, k], \{y, -1, 1\}], \{x, -1, 1\}], \{j, 16\}, \{k, 16\}] ;\]
gg[4,j,k],{y,-1,1},{x,-1,1},{j,16},{k,16];
b5 = Table[Integrate[Integrate[
  gg[5,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b6 = Table[Integrate[Integrate[
  gg[6,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b7 = Table[Integrate[Integrate[
  gg[7,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b8 = Table[Integrate[Integrate[
  gg[8,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b9 = Table[Integrate[Integrate[
  gg[9,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b10 = Table[Integrate[Integrate[
  gg[10,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b11 = Table[Integrate[Integrate[
  gg[11,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b12 = Table[Integrate[Integrate[
  gg[12,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b13 = Table[Integrate[Integrate[
  gg[13,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b14 = Table[Integrate[Integrate[
  gg[14,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b15 = Table[Integrate[Integrate[
  gg[15,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
b16 = Table[Integrate[Integrate[
  gg[16,j,k],{y,-1,1}],{x,-1,1}],{j,16},{k,16}];
Now, the SUBROUTINE JACOBS is utilized to evaluate the Jacobian. Given the i-th iteration \((c_1, \ldots, c_m)\) of the solution we then assemble the Jacobian as follows:

```
for k=1:nelem
    for ir =1:nbf
        ii = elnode(k,ir);
        i = abs(ii);
        for is =1:nbf
            jj=elnode(k,is);
            d=sign(ii*jj);
            j = abs(jj);
            a(i,j) = a(i,j) + (d/Re)*ea(ir,is)/h^2;
            sum = 0;
            for ik=1:nbf
                k3=elnode(k,ik);
                kk= abs(k3);
                d = sign(ii*jj*k3);
                brack = b(ir,is,ik) + b(ir,ik,is);
                sum=sum + d*sol(kk)*brack/h^2;
            end
            a(i,j) = a(i,j) + sum;
        end
    end
end
```
4.10 Calculation of the Load Vector

In each Newton’s iteration, the load vector $\text{rhs}$ is calculated by the same strategy. We set

$$\text{rhs}_i = \sum_{j=1}^{m} A_{ij} c_j + C^T B^{(i)} C - L_i, \quad i = 1, \ldots, m.$$  \hfill (4.25)

In the previous section, we discuss how to compute $A_{ij}$ and $B^{(i)}_{jk}$. It remains to describe the computation of $L_i$. We have

$$L_i = (f, \text{curl } \phi_i) \quad i = 1, \ldots, m,$$

$$= \sum_l \int \int_{e_l} \{f_1(x,y)\phi_{i,y}(x,y) - f_2(x,y)\phi_{i,x}(x,y)\} dx dy. \hfill (4.26)$$

We assemble the above integral element by element as

$$L_i = \sum_{l=1}^{\text{nelem}} \int \int_{e_l} \{f_1(x,y)\phi_{i,y}(x,y) - f_2(x,y)\phi_{i,x}(x,y)\} dx dy,$$

$$= \sum_{l=1}^{\text{nelem}} L^{(l)}_i. \hfill (4.27)$$

We compute $L_i$ by mapping the $e_l$ back to the reference element $\tilde{e}$ as follows. Given $e_l$ we look up the global node numbers in ielnode array and then find x-y coordinates of the vertex nodes in xx, yy arrays with these we calculate the affine map of $e_l$ to $\tilde{e}$. Using equation (4.24) we now make a change of variables reducing $L^{(l)}_i$ to an integral over $\tilde{e}$. After applying the chain rule on (4.27), the final results for the case
of Morley elements are:

\[ L^{(l)}_i = \int\int_{\mathcal{E}} \{ f_1(x, y)\phi_{i,y}(x, y) - f_2(x, y)\phi_{i,x}(x, y) \} dx dy, \]

\[ = \int\int_{\mathcal{E}} \{ \tilde{f}_1(\xi, \eta)\tilde{\phi}_{i,\eta} - \tilde{f}_2(\xi, \eta)\tilde{\phi}_{i,\xi} \} (x_2 - x_1)(y_3 - y_1) d\xi d\eta, \]

\[ = (\int\int_{\mathcal{E}} \tilde{f}_1(\xi, \eta)\tilde{\phi}_{i,\eta} d\xi d\eta)(x_2 - x_1) - (\int\int_{\mathcal{E}} \tilde{f}_2(\xi, \eta)\tilde{\phi}_{i,\xi} d\xi d\eta)(y_3 - y_1). \]

In Bogner-Fox-Schmit, we have

\[ L^{(l)}_i = \frac{h}{2} \int\int_{\mathcal{E}} \tilde{f}_1(\xi, \eta)\tilde{\phi}_{i,\eta} d\xi d\eta - \frac{h}{2} \int\int_{\mathcal{E}} \tilde{f}_2(\xi, \eta)\tilde{\phi}_{i,\xi} d\xi d\eta, \]

where \( h = 2/lx \). The above integral can not usually be evaluated exactly as we did in computing the entries of the Jacobian, so it is necessary to use a numerical integration scheme of sufficient order that does not degrade the accuracy of the basic method. Numerical integration is simply a procedure that approximates an integral by a summation. A geometrical interpretation of this is that the area under curve is the sum of the products of certain heights, \( f(x_i, y_i) \) times some corresponding widths, \( w_i \). In the terminology of numerical integration, the location of the points, \( x_i \), where the heights are computed are called abscissa and the widths, \( w_i \), are called weights.

We use Numerical integration formula with 7 points. Indeed, 7 points numerical integration formulas turn out to integrals all polynomials of degree 4 exactly. The
numerical integration points (abscissa) $(\xi, \eta)$ and the weights $w_i$ are given below

$$
\xi = \begin{pmatrix}
0 \\
1/2 \\
1 \\
1/2 \\
0 \\
1/3
\end{pmatrix}, \quad \eta = \begin{pmatrix}
0 \\
1/2 \\
1 \\
1/2 \\
0 \\
1/3
\end{pmatrix}, \quad \omega = \begin{pmatrix}
1/40 \\
1/15 \\
1/40 \\
1/15 \\
1/40 \\
9/40
\end{pmatrix}.
$$

In the case of rectangles, we use numerical integration formula with 4 points. Indeed, 4 points numerical integration formulas turn out to integrals all bicubic polynomials exactly. The numerical integration points (abscissa) $(\xi, \eta)$ and the weights $w_i$ are below

$$
\xi = \begin{pmatrix}
-\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} \\
-\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}}
\end{pmatrix}, \quad \eta = \begin{pmatrix}
-\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}} \\
\frac{1}{\sqrt{3}}
\end{pmatrix}, \quad w = \begin{pmatrix}
1 \\
1 \\
1 \\
1
\end{pmatrix}.
$$

4.11 Storage of the Jacobian

The efficiency of any iterative linear solver is determined primarily by the performance of the matrix-vector product and therefore on the storage scheme used for the matrix. The coefficient matrix of the Jacobian is sparse. Large-scale linear systems of the form $Ax = b$ can be most efficiently solved if the zero elements of $J$ are not stored. There are many methods for solving the non-zero elements of a sparse matrix. The Compressed Row and Column Storage Formats are most general. They make absolutely no assumptions about the sparsity structure of the matrix. The compressed
Row Storage Formats create 3 vectors: one for floating-point numbers (val), and the other two for integers (col, row). The val vector stores the values of the nonzero elements of the matrix, as they are traversed in a row-wise fashion. The col vector stores the column indexes of the element in val vector. The row vector stores the locations in the val vector that start a row. The compressed Column Storage is identical to the Compressed Row Storage except that the columns of the matrix are stored instead of the rows. In other words, the Compressed Column Storage format is the Compressed Row Storage format of the transpose of the matrix.

In MATLAB, there is a built-in function sparse (...) which build sparse matrix from non-zero and indices. \([i, j, s] = \text{find} (A)\) returns a column vector \(s\) of the nonzero entries in \(A\) and two column vector, \(i\) and \(j\), of the indices of the nonzero elements. All of MATLAB’s built-in arithmetic, logical, and indexity operations can be applied to sparse matrices. Operations in sparse matrices return sparse matrices.

In MATLAB’s code version we use this built-in function which ease the programming and it reduces the code lines. In FORTRAN version, we use the same storage format. We create 3 vectors: one for floating-point (val), and the other two for integers (col, row). The val vector stores the values of the nonzero elements of the matrix. The col vector stores the column indexes of the elements in the val vector. The row vector stores the row indexes of the element in the val vector.

In version II, we do not store the system square matrix \(j\) so its products with the vector \(x\) must be evaluated as a vector, say \(g\), which is summed over element products of \(A^{(l)}\) and \(x^{(l)}\). SUBROUTINE AX is utilized to evaluate the product with element level products, then pull out to give the product. Figure (4.14) shows the locations of the nonzero elements of the Jacobian for different values for \(lx\) for the case of Morley element. The SUBROUTINE STORE \((a, ia, ja, nnz, i, j, cont, index)\) stores the nonzero element value in vector \(a\) and the row-index in the vector \(ia\) and the column
Figure 4.14: Nonzero element location for $lx=2,3,4,5$
indices in the vector $ja$.

4.12 Boundary Conditions Contribution

In SUBROUTINE JACOBS, we assemble the contributions to the Jacobian that are independent of the boundary conditions. The matrix resulting from the calculations in SUBROUTINE JACOBS is called the primitive stiffness matrix. Finally, in SUBROUTINES BOUNDARY-A(...) and BOUNDARY-RHS(...) the boundary conditions are added on as an extra set of linear equations. The boundary node number are generated by the SUBROUTINE MAELNODE and stored in the vector $ibo$. The calculation works as follows. Now, we list some information about the Jacobian matrix.

**Morley elements**

- Number of boundary points = $4lx = bp$
- Number of normal boundary points = $4lx = bn$
- Number of interior points = $(lx - 1)^2 = ip$
- Number of normal interior points = $lx(3lx - 2) = in$

Number of non-zero elements in the Jacobian corresponds to boundary points = $12(bp - 4) + 2(6) + 2(9)$
Number of non-zero elements in the Jacobian corresponds to normal boundary points = 6\( (bn) \)

Number of non-zero elements in the Jacobian corresponds to interior points = 19\( (ip) \)

Number of non-zero elements in the Jacobian corresponds to normal interior points = 9\( (in) \)

Number of non-zero entries of the Jacobian = 46\( (lx) \) + 16\( x \) + 1

Bandwidth of the matrix = 4\( x \) + 2, 1, 4\( x \) + 2.

Bogner-Fox-Schmit

Number of boundary nodes = 16\( lx \)

Number of interior nodes = 4\( (lx - 1) \)²

Number of nonzero elements in the Jacobian corresponding to boundary nodes = 96\( lx \) - 32

Number of nonzero elements in the Jacobian corresponding to interior nodes = 144\( (lx - 1) \)²

Number of nonzero entries of the Jacobian = 16\( (9lx^2 - 12x + 7) \)

Notice that Figure (4.15) shows the Bandwidth of the Jacobian matrix for the case of Morley element.

4.13 Solving the Linear System

After we compute the matrix \( J \) and the \( rhs \) at the \( i \)-th Newton's iteration, we need to solve this linear system to compute the next iteration. The resulting linear system is non-symmetric whose symmetric part is positive definite. Moreover, the resulting
Many methods exist for solving linear system of equations in [11]. The trick is to find the most effective method. Unfortunately, a method that works well for one problem type may not work as well for another. In some cases, it may not work at all. We restrict our solver to iterative methods, which work by repeatedly improving an approximate solution until it is accurate enough. These methods do not need the structure of the matrix. They need only a matrix-vector product. Thus one need only supply a SUBROUTINE for computing the product.

Many iterative methods have been developed to solve a large non-symmetric system of equations. Some of them are Generalized Minimal Residual (GMRES), Bi Conjugate Gradient (BiCG), Conjugate Gradient Square Method (CGS) and Bi Conjugate Gradient Stabilized (Bi-CGSTAB). We choose the Bi Conjugate Gradient Stabilized method. Bi-CGSTAB requires two matrix-vector products and four inner products. In our case, Bi-CGSTAB converges faster than CGS and BiCG. The MATLAB code
for the Bi Conjugate Gradient Stabilized Method is given in [11] and shown below

function [x,k] = bicgstab(b,tol,c)
k=0;
x = zeros([length(b),1]) ;

r = c*x ;
r = b-r ;
rh= r ;
for i=1:2
    row = rh'*r ;
    if(row == 0 ) break; end;
    if(i == 1)
        p = r ;
    else
        beta = (row/rowold)*(alpha/w) ;
        p = r + beta*(p-w*v) ;
    end
    p = p ;
    v = c*p ;
    alpha = row/(rh'*v) ;
    s = r - alpha*v ;
    if( norm(s)<tol)
        k=i ;
        x=x+alpha*p ;
        disp('I stop because norm(s)<tol');
        break ;
end
s = s ;
t = c*s ;
w = (t'*s)/(t'*t) ;
up = alpha*p +w*s ;
x = x + up ;
r = s - w*t ;
if(w == 0 )
    disp('I can not continue iteration because w = 0');
    break ;
end
if( norm(up)<tol & norm(r)<tol )
    k = i ;
    disp('I stop because norm(r,up)<tol');
    break
end
rowold=row ;
end

The stopping criterion for the SUBROUTINE BICGSTAB(–) are

1. \( ||c^{(i)} - c^{(i+1)}|| \) is small enough to stop.

2. \( ||\text{residual}|| \) is small enough to stop.

3. The integer max-iteration is the maximum number of iteration that algorithm will be permitted to perform.
4.14 COARSE-LEVEL-SOLVE Subroutine

In the main program, we call SUBROUTINE MAELNODE (–) which generate the geometry information. It returns ielnode vector which represents the information about the global node ordering and it returns ibo vector which has information about the boundary node. Then we call COORD(–) SUBROUTINE which return xx-vector and yy-vector which have the x-coordinate and y-coordinate of each node. Then we call MAEA(–) SUBROUTINE which returns the elements stiffness matrix. Then we call quad7(–) SUBROUTINE which returns the abscissa and the weight for numerical integration formula. Then it calls PHIQ(–) SUBROUTINE which returns the function values, the x-derivative and the y-derivative of the reference element basis functions at the integration points. Then it calls MAB(–) SUBROUTINE which returns the element 3-dimentional array B.

After all these SUBROUTINE calls, COARSE-LEVEL-SOLVE SUBROUTINE is called to compute, \( \psi^H \), an approximation of the streamfunction on the coarse mesh. Inside the coarse-level-solver SUBROUTINE we start Newton's iteration. In each iteration we make 5 calls to 5 SUBROUTINES until we satisfy the shopping criteria. First call is JACOB(–) SUBROUTINE which compute the Jacobian . Then we call FORCE(–) SUBROUTINE to compute \( rhs \)-vector. Then we call BOUNDARY-A(–) and BOUNDARY RHS(–) SUBROUTINES to take care of the boundary conditions. The last call is BICGSTAB(–) which will solve the linear system.

4.15 FINE-LEVEL-SOLVE Subroutine

Here we consider the implementation of the two level element method for one level formulation. After we compute the approximate solution \( \psi^H \) in the coarse mesh, we
have to compute $\psi^h$ on the fine mesh. For specificity, consider the following problem, solve the linear system on fine mesh for $\psi^h \in X^h$,

$$a(\psi^h, \phi^h) + b(\psi^H, \psi^h, \phi^h) = (f, \text{curl } \phi^h) \quad \text{for all } \phi^h \in X^h. \quad (4.28)$$

Here,

$$a(\psi^h, \phi^h) = \sum_l \int_{e_l} (\psi^h_{xx} \phi^h_{xx} + 2\psi^h_{xy} \phi^h_{xy} + \psi^h_{yy} \phi^h_{yy}) dxdy,$$

$$b(\psi^H, \psi^h, \phi^h) = \sum_l \int_{e_l} \left\{ (\psi^H_y \psi^h_{xx} - \psi^H_x \psi^h_{xy}) \phi^h_x - (\psi^H_x \psi^h_{yy} - \psi^H_y \psi^h_{xy}) \phi^h_y \right\} dxdy. \quad (4.29)$$

The solution of the above equation requires

1. generating the fine mesh.

2. the assembly of the linear system.

3. the solution of the resulting linear system.

To generate the fine mesh, one input a new lx number into the SUBROUTINE MAELNODE(–) and COORD(–). Then, SUBROUTINE MAELNODE(–) and SUBROUTINE COORD(–) will generate $ielnode$ array, $ibo$ vector, $xx$-vector and $yy$-vector for the new mesh. The $ielnode$ array has a global node label map for all elements for the new mesh. The $ibo$-vector contains the boundary number. $xx$-vector and $yy$-vector have the x-coordinate and y-coordinate of all nodes.

To complete item (2), we need to assemble the matrix $A$ and the $rhs$ of the linear system. To assemble the matrix $A$, we have to compute the value of $\psi^H$ on the new nodes because the values of $\psi^H$ is only known at the coarse mesh nodes.

Let $\{\phi_i^h : i = 1, \ldots, m_h\}$ denote the global basis functions, defined on $\bar{\Omega}_h$. We want to approximate $\psi(x, y)$ by a function $\psi^h$ in the space $S^h = \text{span}\{\phi_i^h : i = 1, \ldots, m_h\}$. 
As $\psi^h(x, y)$ lies in $S^h$ we may expand:

$$\psi^h(x, y) = \sum_{i=1}^{m_h} c_j \phi_j^h(x, y).$$  \hspace{1cm} (4.30)

Now, the problem is computing the coefficients $c_j$'s in (4.30). Taking the variation of (4.28) over $S^h$ given the variational equations which $\psi^h(x, y)$ satisfies:

$$a(\psi^h, \phi^h) + b(\psi^H, \psi^h, \phi^h) = (f, \text{curl } \phi^h) \quad \text{for all } \phi^h \in S^h. \quad (4.31)$$

Setting $\phi^h = \phi_i^h(x, y)$ in (4.31) and substituting (4.30) yields

$$\sum_{j=1}^{m_h} c_j a(\phi_j^h, \phi_i^h) + \sum_{j=1}^{m_h} c_j b(\psi^H, \phi_j^h, \phi_i^h) = (f, \text{curl } \phi_i^h) \quad i = 1, \ldots, m. \quad (4.32)$$

We assemble $b(\psi^H, \phi_j^h, \phi_i^h)$ element by element as

$$b(\psi^H, \phi_j^h, \phi_i^h) = \sum_{i=1}^{n_{elem}} b^{(i)}(\psi^H, \phi_j^h, \phi_i^h). \quad (4.33)$$

Now, the triangle $e_i$ is inside a bigger triangle $g$ which is an element of the coarse mesh, as shown in Figure (4.16).

Hence, we have

$$\psi^H = \sum_{k=1}^{6} d_k \phi_k^{(H)}, \quad (4.34)$$

where $\phi_1^{(H)}, \ldots, \phi_6^{(H)}$ are the local basis functions of the triangle $g$. $d_1, d_2$ and $d_3$ are the function values of $\psi^H$ at the vertices of the triangle $g$, $d_4, d_5$ and $d_6$ are the normal derivatives values of $\psi^H$ at the midsides points of the triangle $g$. 

Substituting (4.34) in (4.33) yields

\[ b(\psi^H, \phi_j^h, \phi_i^h) = \sum_{l=1}^{\text{nelem}_h} \sum_{k=1}^6 d_k b^{(l)}(\phi_k^H, \phi_j^h, \phi_i^h). \]  

(4.35)

We compute \( b^{(l)}(\phi_k^H, \phi_j^h, \phi_i^h) \) as follows

\[ b^{(l)}(\phi_k^H, \phi_j^h, \phi_i^h) = \int_{e_l} \left\{ (\phi_{k,y}^H \phi_{j,xy}^h - (\phi_{k,x}^H \phi_{j,yy}^h) \phi_{i,y}^h + \right. \]
\[ - (\phi_{k,x}^H \phi_{j,xy}^h - (\phi_{k,y}^H \phi_{j,xx}^h)) \phi_{i,x}^h \} \right\} dxdy. \]  

(4.36)
By chain rule calculation we have

\[ \phi^h_{x} = \frac{1}{x_2 - x_1} \tilde{\phi}_\xi = \frac{1}{h} \tilde{\phi}_\xi, \]
\[ \phi^h_{y} = \frac{1}{y_3 - y_1} \tilde{\phi}_\eta = \frac{1}{h} \tilde{\phi}_\eta, \]
\[ \phi^h_{xy} = \frac{1}{(x_2 - x_1)(y_3 - y_1)} \tilde{\phi}_{\xi\eta} = \frac{1}{h^2} \tilde{\phi}_{\xi\eta}, \]
\[ \phi^h_{xx} = \frac{1}{(x_2 - x_1)^2} \tilde{\phi}_{\xi\xi} = \frac{1}{h^2} \tilde{\phi}_{\xi\xi}, \]
\[ \phi^h_{yy} = \frac{1}{(y_3 - y_1)^2} \tilde{\phi}_{\eta\eta} = \frac{1}{h^2} \tilde{\phi}_{\eta\eta}, \]
\[ \phi^H_{k,x} = \frac{1}{x_2^H - x_1^H} \tilde{\phi}_\xi = \frac{1}{H} \tilde{\phi}_\xi, \]
\[ \phi^H_{k,y} = \frac{1}{y_3^H - y_1^H} \tilde{\phi}_\eta = \frac{1}{H} \tilde{\phi}_\eta. \] (4.37)

Substituting (4.37) in (4.36) yields

\[ b^{(i)}(\phi^H_k, \phi^h_j, \phi^h_i) = \sum_{l=1}^{n_{elem}} \sum_{k=1}^{6} \frac{d_k}{Hh} \tilde{b}^{(i)}_{jk}. \] (4.38)

Hence, using (4.35) and (4.38) gives

\[ b(\psi^H, \phi^h_j, \phi^h_i) = \sum_{l=1}^{n_{elem}} \sum_{k=1}^{6} \frac{d_k}{Hh} \tilde{b}^{(i)}_{jk}. \]
Finally, we assemble $a(\phi^h_j, \phi^h_i)$ and $(f, \text{curl } \phi^h_i)$ as before in section 4.9 and section 4.10 by mapping the $e_l$ back to the reference element $\tilde{e}$ and applying the chain rule. The final results are:

$$a(\phi^h_j, \phi^h_i) = \sum_{l=1}^{\text{nelem}_h} h^{-2} \tilde{A}_{ij},$$

$$(f, \text{curl } \phi^h_i) = \sum_{l=1}^{\text{nelem}_h} L^{(l)}_i,$$

where

$$L^{(l)}_i = (\int_{\tilde{e}} \tilde{f}_1(\xi, \eta) \tilde{\phi}_{i,\eta} d\xi d\eta)(x_2 - x_1) - (\int_{\tilde{e}} \tilde{f}_2(\xi, \eta) \tilde{\phi}_{i,\xi} d\xi d\eta)(y_3 - y_1).$$

At this point, we have assembled the linear system. Figure (4.17) shows the location of the nonzero elements of the resulting matrix for different values of $h, H$. In Bogner-Fox-Schmit, we have by chain rule

$$\phi^h_x = 2h^{-1} \tilde{\phi}_\xi, \quad \phi^H_x = 2H^{-1} \tilde{\phi}_\xi, \quad (4.39)$$

$$\phi^h_y = 2h^{-1} \tilde{\phi}_\eta, \quad \phi^H_y = 2H^{-1} \tilde{\phi}_\eta, \quad (4.40)$$

$$\phi^h_{xx} = 4h^{-2} \tilde{\phi}_{\xi\xi}, \quad \phi^h_{yy} = 4h^{-2} \tilde{\phi}_{\eta\eta}, \quad \phi^h_{xy} = 4h^{-2} \tilde{\phi}_{\xi\eta}. \quad (4.41)$$

Substituting the above equation in (4.36) yields

$$b^l(\phi^H_k, \phi^h_j, \phi^h_i) = \frac{4}{Hh} \tilde{b}^i_{jk}. \quad (4.36)$$

Now, we are ready to solve the linear system by an iterative solver. The last call is BICGSTAB(−) which will solve the resulting linear system.
Figure 4.17: Nonzero entries for \((H, h) = (\frac{1}{2}, \frac{1}{4}), (\frac{1}{3}, \frac{1}{6})\)
Chapter 5

Numerical Tests
5.1 Introduction

Although the theoretical result in chapter (2) looks quite nice it has an essential drawback from the practical point of view. There are several open questions left from theory. How does the constant $C$ in the estimate (2.23) depend on the Reynolds number $Re$ and how do the error norms $\| \psi - \psi^h \|_h$ behave if we keep the mesh sizes $H$ and $h$ fixed and increase $Re$? How much time in practical settings we save when we use a two level method instead of a one level method? Will the solution’s quality differ in using the two level method instead of the one level method?

In section (5.2) we test the code described in chapter (2) with problems which have a known solution to compare one level vs. two level methods. Section (5.3) contains the graphs of the test problem with known solution with different values of Reynolds numbers. This problem have been tested in [63] with $Re = 10, 50, 100, 1000$ and 2000. In section (5.5) we solve the driven cavity problems. These flows have been widely used as test cases for validating incompressible fluid dynamics algorithms. We will compare our streamlines with the streamlines in [31, 2].

5.2 Accuracy Two Level Method vs. One Level Method

We consider as a test example the 2D Navier-Stokes problem as the unit square $\Omega = [0, 1]^2$ where we define the right hand side by

$$f := -\frac{1}{Re} \Delta u + (u \cdot \nabla)u + \nabla p,$$
with the following prescribed exact solution

\[
u = \begin{pmatrix}
\psi_y \\
\psi_x
\end{pmatrix},
\]

\[
\psi(x, y) = x^2(x - 1)^2y^2(y - 1)^2,
\]

\[
p(x, y) = x^3 + y^3 - 0.5.
\]

For this test problem, all requirements of the theory concerning the geometry of the domain and the smoothness of the data are satisfied. Moreover, the streamfunction \( \psi(x, y) \) satisfies the boundary conditions of the streamfunction equation of the Navier-Stokes equation (i.e)

\[
\psi(x, y) = 0 \quad \text{where} \quad x = 0, x = 1, y = 0, \text{or} \ y = 1, \quad (5.1)
\]

\[
\frac{\partial \psi(x, y)}{\partial n} = 0 \quad \text{where} \quad x = 0, x = 1, y = 0, \text{or} \ y = 1. \quad (5.2)
\]

Also the exact solution is very smooth and does not depend on the Reynolds number. Our goal in this test is to validate the code and the properties and merits of the two level method as compared with the one level method. In all numerical calculations in this section we have used the Bogner-Fox-Schmit elements with \( Re = 10 \) and \( tol = 10^{-3} \). We pick three values of \( h \). They are \( \frac{1}{8}, \frac{1}{14} \) and \( \frac{1}{16} \). All the computations in this section are done using 133Mhz Intel Pentium Processor with 32 MHz Ram running Windows NT. The cpu-time, number of Newton’s iterations, number of Bicgstab iterations, the \( L^2 \)-error, \( H^1 \)-error and \( H^2 \)-error of the streamfunction \( \psi \) for the one level method for different values of \( h \) are tabulated in Table (5.1). Table (5.2) shows cpu-time, number of Newton’s iterations and the number of Bicgstab’s iterations for each linear solver for two level method. Figure (5.1) shows, for fixed
$Re = 10$ using one level method, the streamlines for $h = \frac{1}{8}, \frac{1}{14}, \frac{1}{16}$ and the corresponding three-dimensional plot of the streamfunction $\psi(x, y)$.

**Remarks 1**

1. From Table (5.1) and Table (5.2), the cpu-time for two level method is much smaller than the corresponding cpu-time for one level method. For $h = \frac{1}{8}, \frac{1}{14}$, we save about 57%. For $h = \frac{1}{16}$ we save about 73%. For example, in $h = \frac{1}{16}$ we need to solve a nonlinear system of 1156 equations which requires solving three linear systems of equations of order 1156. The corresponding two level method requires solving a nonlinear system of 324 equations and a linear system of 1156 equations. We anticipate the savings to increase as the mesh is further refined.

2. From Figure (5.1) and Figure (5.2) both graphs are exactly similar, which means that the two level method produces a solution with the same quality as the one level method.

3. From Table (5.1) and Table (5.2) both $H^1$-error and $H^2$-error are of the same order, which means that the velocity field is of the same error and quality in both methods since $u = \psi_y$ and $v = -\psi_x$.

4. From Table (5.1) and Table (5.2) $L^2$-error of the one level method table is somewhat better than the $L^2$-error of the two level method table.

5. In this test we used the approximate solution produced from step 1 in assembling the matrix of step 2. We used the zero vector as an initial guess for solving the linear system of step 2.
<table>
<thead>
<tr>
<th>$h$</th>
<th>cpu time</th>
<th># of Newton’s iteration</th>
<th># of Bicgstab</th>
<th>$| \psi - \psi^h |_{0,h}$</th>
<th>$| \psi - \psi^h |_{1,h}$</th>
<th>$| \psi - \psi^h |_{2,h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{8}$</td>
<td>69.74</td>
<td>3</td>
<td>26,30,24</td>
<td>$3.94 \times 10^{-5}$</td>
<td>$4.23 \times 10^{-2}$</td>
<td>$1.41 \times 10^{-1}$</td>
</tr>
<tr>
<td>$\frac{1}{14}$</td>
<td>294.35</td>
<td>3</td>
<td>50,56,55</td>
<td>$2.41 \times 10^{-5}$</td>
<td>$1.74 \times 10^{-2}$</td>
<td>$8.87 \times 10^{-2}$</td>
</tr>
<tr>
<td>$\frac{1}{16}$</td>
<td>992.23</td>
<td>3</td>
<td>63,75,67</td>
<td>$3.55 \times 10^{-5}$</td>
<td>$1.41 \times 10^{-2}$</td>
<td>$8.08 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

Table 5.1: One level method.

<table>
<thead>
<tr>
<th>$H, h$</th>
<th>cpu time</th>
<th># of Bicgstab in coarse</th>
<th># of Bicgstab in fine</th>
<th>$| \psi - \psi^h |_{0,h}$</th>
<th>$| \psi - \psi^h |_{1,h}$</th>
<th>$| \psi - \psi^h |_{2,h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{4}, \frac{1}{8}$</td>
<td>29.91</td>
<td>10,18,15</td>
<td>50</td>
<td>$1.40 \times 10^{-3}$</td>
<td>$4.09 \times 10^{-2}$</td>
<td>$1.42 \times 10^{-1}$</td>
</tr>
<tr>
<td>$\frac{1}{7}, \frac{1}{14}$</td>
<td>128.28</td>
<td>22,24,21</td>
<td>118</td>
<td>$2.31 \times 10^{-4}$</td>
<td>$1.73 \times 10^{-2}$</td>
<td>$8.70 \times 10^{-2}$</td>
</tr>
<tr>
<td>$\frac{1}{8}, \frac{1}{16}$</td>
<td>267.33</td>
<td>26,30,24</td>
<td>169</td>
<td>$1.70 \times 10^{-4}$</td>
<td>$1.41 \times 10^{-2}$</td>
<td>$7.94 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

Table 5.2: Two level method.
Figure 5.1: Streamlines for $h = \frac{1}{8}, \frac{1}{14}, \frac{1}{16}$ with $Re=10$ and the corresponding 3-D graphs using one level method
Figure 5.2: Streamlines for \((H, h) = \left(\frac{1}{4}, \frac{1}{8}\right), \left(\frac{1}{7}, \frac{1}{11}\right), \left(\frac{3}{5}, \frac{1}{10}\right)\) with Re=10 and the corresponding 3-D graphs using two level method.
5.3 Robustness: Test Problem for Different Reynolds Numbers

In this section, we consider the example which was described in the previous section. The exact solution is very smooth and does not depend on the Reynolds number. The point of these tests is to increase the Reynolds number with $h$ fixed and test the robustness of the method. Schieweck [63] tested his code with this problem. He used two nonconforming finite element approximations of upwind type for the velocity-pressure formulation. Our goal in this test is to determine the validation of the code and the norm behavior when $Re$ is varying and compare our streamlines with Schieweck’s streamlines.

The numerical computations of this section was obtained using a Sun Ultra 2 with 2 200Mhz ultrasparc processors running Solaris 2.5.1. In all numerical calculations, we used the Bogner-Fox-Schmit rectangles and Morley triangles. The streamlines for $Re = 10, 50, 100, 200, 1000$ and $2000$ were obtained with $17 \times 17$ grid points on the coarse mesh and $33 \times 33$ grid on the fine mesh. Hence, a mesh of 256 elements and a mesh of 1024 elements were used in this test for the case of Bogner-Fox-Schmit rectangles. In Morley elements, a mesh of 512 elements and a mesh of 2048 elements were used.

The initial guess in solving the nonlinear system of equation on the coarse mesh was the zero vector or the zero function except the cases where $Re = 200, 1000$ and 2000. In this case the one level method with the choice of initial guess with the same tolerance failed. So we used the first iterates of the Newton’s method when we solve the problem with $Re = 100$ as the initial guess for these cases. The number of Bicgstab steps needed for converging the linear system in the fine mesh and the number of Bicgstab steps needed for solving the nonlinear system of equations in
the coarse mesh are given in Table (5.3). Moreover, Table (5.3) represents the $L^2$-error, $H^1$-error and $H^2$-error. The streamlines are plotted in Figure (5.3) for $Re = 10, 50, 100, 200, 1000$ and $2000$. The values of $\psi$ along the center contours and the corner contours are listed in Table (5.4). These streamlines graphs are compared with results of Schieweck [63] with two nonconforming finite element approximations of upwind type for the velocity-pressure formulation. Schieweck’s streamlines are shown in Figure (5.4) for the fixed mesh size $h = 1/64$ and $Re = 10, 50, 100, 200, 1000$ and $2000$. The streamlines generated by using Morley elements are plotted in Figure (5.5).

<table>
<thead>
<tr>
<th>values of $\psi$ along the center contours</th>
<th>values of $\psi$ along the corner contours</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7109e-3</td>
<td>-8.2031e-5</td>
</tr>
<tr>
<td>3.5150e-3</td>
<td>-3.9063e-5</td>
</tr>
<tr>
<td>2.7344e-3</td>
<td>-3.9063e-6</td>
</tr>
<tr>
<td>1.9531e-3</td>
<td>-3.9063e-7</td>
</tr>
<tr>
<td>1.1719e-3</td>
<td>-3.9063e-8</td>
</tr>
<tr>
<td>3.9063e-4</td>
<td></td>
</tr>
<tr>
<td>7.0313e-5</td>
<td></td>
</tr>
<tr>
<td>3.9063e-5</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Values for streamline contours in Figure (5.3).
<table>
<thead>
<tr>
<th>$Re$</th>
<th>$(H,h)$</th>
<th># Bicgstab in coarse</th>
<th># of Bicgstab in fine</th>
<th>$| \psi - \psi^h |_{0,h}$</th>
<th>$| \psi - \psi^h |_{1,h}$</th>
<th>$| \psi - \psi^h |_{2,h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$(\frac{1}{16}, \frac{1}{32})$</td>
<td>61,66,43</td>
<td>576</td>
<td>$4.21 \times 10^{-5}$</td>
<td>$4.84 \times 10^{-3}$</td>
<td>$5.27 \times 10^{-2}$</td>
</tr>
<tr>
<td>50</td>
<td>$(\frac{1}{16}, \frac{1}{32})$</td>
<td>54,83,173,292,82</td>
<td>1676</td>
<td>$1.11 \times 10^{-4}$</td>
<td>$4.82 \times 10^{-3}$</td>
<td>$5.19 \times 10^{-2}$</td>
</tr>
<tr>
<td>100</td>
<td>$(\frac{1}{16}, \frac{1}{32})$</td>
<td>53,254,421,668,1287</td>
<td>4356</td>
<td>$4.44 \times 10^{-4}$</td>
<td>$5.00 \times 10^{-3}$</td>
<td>$5.11 \times 10^{-2}$</td>
</tr>
<tr>
<td>200</td>
<td>$(\frac{1}{16}, \frac{1}{32})$</td>
<td>53,254,421,668,1287,454</td>
<td>4356</td>
<td>$3.06 \times 10^{-4}$</td>
<td>$4.52 \times 10^{-3}$</td>
<td>$5.07 \times 10^{-2}$</td>
</tr>
<tr>
<td>1000</td>
<td>$(\frac{1}{16}, \frac{1}{32})$</td>
<td>53,254,421,668,1287,1156</td>
<td>4356</td>
<td>$9.62 \times 10^{-4}$</td>
<td>$1.08 \times 10^{-2}$</td>
<td>$1.83 \times 10^{-1}$</td>
</tr>
<tr>
<td>2000</td>
<td>$(\frac{1}{16}, \frac{1}{32})$</td>
<td>53,254,421,668,1287,1156,1156</td>
<td>4356</td>
<td>$1.93 \times 10^{-3}$</td>
<td>$2.59 \times 10^{-2}$</td>
<td>$4.93 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

Table 5.3: Two level for test problem.
Figure 5.3: Streamlines for $H = \frac{1}{16}, h = \frac{1}{32}$ using Bogner-Fox-Schmit element.
Figure 5.4: Streamlines for $h = \frac{1}{64}$ (courtesy F. Schieweck [63]).
Figure 5.5: Streamlines for $H = \frac{1}{16}, h = \frac{1}{32}$ using Morley element.
Remarks 2

1. Figures (5.3, 5.4, 5.5) show that the increasing in the Reynolds number will affect the streamlines.

2. Figure (5.3) shows increasing in Reynolds number will increase the number of corner contours.

3. Figure (5.3) shows better quality of the solutions if they are compared to Figure (5.4) and (5.5). This has two explanations. Firstly, the use of high order polynomials. In Bogner-Fox-Schmit, the polynomial is bicubic. Secondly, the use of conforming elements will decrease the error.

4. From Table (5.3), $H^2$-error and $H^1$-error are of the same order. However, increasing the Reynolds number will effect the $L^2$-error.

5.4 Meshes Scaling Test

The error between the coarse and fine meshes are related superlinearly via:

$$| \psi - \psi^h |_2 \leq C \left\{ \inf_{w^h \in X^h} | \psi - w^h |_2 + | \ln h |^{1/2} \cdot | \psi - \psi^H |_1 \right\}.$$  

If the Bogner-Fox-Schmit rectangles are used, then the coarse and fine meshes are related by

$$h = O(H^{3/2} \ln H^{1/4}).$$  \hspace{1cm} (5.3)

In this section, we will test the scaling (5.3) by fixing $H = \frac{1}{4}$. From equation (5.3), the computed values of $h$ is $\frac{1}{8}$ or $\frac{1}{5}$. The idea here is how much can we reduce the value of $h$, and how well the quality of the solution be. For this purpose, we consider
the example in section (5.2). In all numerical calculations in this section, we have used the Bogner-Fox-Schmit elements with $Re = 100$ and $tol = 10^{-6}$.

We pick four pairs of $(H, h)$. They are $(\frac{1}{4}, \frac{1}{8}), (\frac{1}{4}, \frac{1}{5}), (\frac{1}{4}, \frac{1}{16})$ and $(\frac{1}{4}, \frac{1}{20})$. All the computations in this section are done using 133 MHz Intel Pentium processor with 32 MHz Ram running Windows NT. The $L^2$-error, $H^1$-error and $H^2$-error of the stream-function $\psi$ these four pairs of $(H, h)$ are tabulated in Table (5.5). Figure (5.6) shows the streamlines for $(H, h) = (\frac{1}{4}, \frac{1}{8}), (\frac{1}{4}, \frac{1}{5}), (\frac{1}{4}, \frac{1}{16})$ and $(\frac{1}{4}, \frac{1}{20})$.

**Remark 1**

Table (5.5) shows that the error norms increase if we keep the coarse mesh size fixed and decrease the fine mesh size. This effect can be seen more clearly in Figure (5.6). Figure (5.6) shows that the number of negative streamlines increases if we decrease the fine mesh size $h$. One explanation for this effect is due to the error bound of this problem, namely,

$$|\psi - \psi^h| \leq C_1 h^2 + C_2 \sqrt{\ln h} |H^3|$$

where $C_2 > C_1$.

If we decrease the fine mesh size $h$ then the first term in the above error bound decreases. However, the second term increases and hence it becomes the dominate term. To get the optimum value for $h$ one may need to balance between these two terms. The error bounds for all four cases are given below.
\[ |\psi - \psi^h|_2 \leq C_1(0.0156) + C_2(0.0225) \quad \text{for} \quad (H, h) = \left( \frac{1}{4}, \frac{1}{8} \right), \]

\[ |\psi - \psi^h|_2 \leq C_1(0.00694) + C_2(0.0246) \quad \text{for} \quad (H, h) = \left( \frac{1}{4}, \frac{1}{12} \right), \]

\[ |\psi - \psi^h|_2 \leq C_1(0.00391) + C_2(0.026) \quad \text{for} \quad (H, h) = \left( \frac{1}{4}, \frac{1}{16} \right), \]

\[ |\psi - \psi^h|_2 \leq C_1(0.0025) + C_2(0.027) \quad \text{for} \quad (H, h) = \left( \frac{1}{4}, \frac{1}{20} \right). \]

<table>
<thead>
<tr>
<th>\text{Re}</th>
<th>(H,h)</th>
<th>| \psi - \psi^h |_{0,h}</th>
<th>| \psi - \psi^h |_{1,h}</th>
<th>| \psi - \psi^h |_{2,h}</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 \left( \frac{1}{4}, \frac{1}{8} \right)</td>
<td>5.53\times10^{-8}</td>
<td>2.37\times10^{-5}</td>
<td>9.89\times10^{-5}</td>
<td></td>
</tr>
<tr>
<td>100 \left( \frac{1}{4}, \frac{1}{12} \right)</td>
<td>2.83\times10^{-7}</td>
<td>2.52\times10^{-5}</td>
<td>1.09\times10^{-4}</td>
<td></td>
</tr>
<tr>
<td>100 \left( \frac{1}{4}, \frac{1}{16} \right)</td>
<td>1.14\times10^{-7}</td>
<td>2.59\times10^{-5}</td>
<td>1.13\times10^{-4}</td>
<td></td>
</tr>
<tr>
<td>100 \left( \frac{1}{4}, \frac{1}{20} \right)</td>
<td>3.41\times10^{-7}</td>
<td>2.65\times10^{-5}</td>
<td>1.16\times10^{-4}</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: \( L^2 \)-error, \( H^1 \)-error and \( H^2 \)-error of \( \psi \) for different pairs \((H, h)\).
Figure 5.6: Streamlines for $(H, h) = \left( \frac{1}{4}, \frac{1}{8} \right)$, $(\frac{1}{4}, \frac{1}{8})$, $(\frac{1}{4}, \frac{1}{16})$ and $(\frac{1}{4}, \frac{1}{20})$. 
5.5 The Driven Cavity Problem

Cavity flows have been a subject of study for some time. These flows have been widely used as test cases for validating incompressible fluid dynamics algorithms. Corner singularities for two-dimensional fluid flows are very important, since most examples of physical interest have corners. For example, singularities of most elliptic problems develop when the boundary contour is not smooth. In this section, we will study the driven flow in a rectangular cavity when the top surface moves with a constant velocity along its length. The upper corners where the moving surface meets the stationary walls are singular points of the flow at which the horizontal velocity is multi-valued. The lower corners are also weakly singular points.

We consider a domain $\Omega = [0,1] \times [0,1]$ with no-slip boundary conditions, i.e, $u = v = 0$ in all boundaries except $y = 1$ where $u = 1$. This problem have been studied and addressed by many researchers including Ghia, Ghia, Shin [31], J.E Akin [2] and Betts-Haroutunian [12]. Ghia’s algorithm is based on the time dependent streamfunction equation using a multi grid technique. The numerical computational in this section was obtained using a Sun Ultra 2 with 2 200 MHz ultrasparc processors running Solaris 2.5.1. Bogner-Fox-Schmit elements are used with $17 \times 17$ grid points on the coarse mesh and $33 \times 33$ grid points on the fine mesh. The streamlines for $Re = 1, 10, 50$ and $100$ are plotted in Figure (5.7, 5.8). These results are compared with results of Ghia [31] obtained with $129 \times 129$ grid. Moreover, these results are illustrated with Akin’s streamlines in Figure (5.9). Table (5.6) and Table (5.7) list the numerical values corresponding to the velocity profiles for lines passing through the geometric center of the cavity. Table (5.8) represents the number of Bicgstab steps in coarse mesh and the number of Bicgstab steps in the fine mesh.
Remarks 3

1. From Figure (5.9), good agreement can be found in the case of $Re = 1$.

2. Good agreement can be found between the case of $Re = 10$ in Figure (5.7) and the case of $Re = 100, 400$ in Figure (5.8). Moreover, The corresponding $u$-velocity and $v$-velocity agree, too Figure (5.10).
<table>
<thead>
<tr>
<th>y</th>
<th>Re=1</th>
<th>Re=10</th>
<th>Re=50</th>
<th>Re=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0625</td>
<td>-0.0387</td>
<td>-0.0759</td>
<td>-0.2725</td>
<td>-0.4057</td>
</tr>
<tr>
<td>0.1250</td>
<td>-0.0700</td>
<td>-0.1473</td>
<td>-0.4184</td>
<td>-0.4439</td>
</tr>
<tr>
<td>0.1875</td>
<td>-0.0977</td>
<td>-0.2213</td>
<td>-0.3736</td>
<td>-0.3653</td>
</tr>
<tr>
<td>0.2500</td>
<td>-0.1239</td>
<td>-0.2846</td>
<td>-0.2975</td>
<td>-0.3020</td>
</tr>
<tr>
<td>0.3125</td>
<td>-0.1493</td>
<td>-0.3104</td>
<td>-0.2366</td>
<td>-0.2371</td>
</tr>
<tr>
<td>0.3750</td>
<td>-0.1730</td>
<td>-0.2850</td>
<td>-0.1755</td>
<td>-0.1722</td>
</tr>
<tr>
<td>0.4375</td>
<td>-0.1929</td>
<td>-0.2212</td>
<td>-0.1141</td>
<td>-0.1074</td>
</tr>
<tr>
<td>0.5000</td>
<td>-0.2048</td>
<td>-0.1428</td>
<td>-0.0524</td>
<td>-0.0423</td>
</tr>
<tr>
<td>0.5625</td>
<td>-0.2027</td>
<td>-0.0636</td>
<td>0.0103</td>
<td>0.0241</td>
</tr>
<tr>
<td>0.6250</td>
<td>-0.1789</td>
<td>0.0140</td>
<td>0.0752</td>
<td>0.0930</td>
</tr>
<tr>
<td>0.6875</td>
<td>-0.1238</td>
<td>0.0911</td>
<td>0.1437</td>
<td>0.1658</td>
</tr>
<tr>
<td>0.7500</td>
<td>-0.0265</td>
<td>0.1663</td>
<td>0.2173</td>
<td>0.2442</td>
</tr>
<tr>
<td>0.8125</td>
<td>0.1258</td>
<td>0.2365</td>
<td>0.2993</td>
<td>0.3302</td>
</tr>
<tr>
<td>0.8750</td>
<td>0.3472</td>
<td>0.3092</td>
<td>0.3877</td>
<td>0.4281</td>
</tr>
<tr>
<td>0.9375</td>
<td>0.6443</td>
<td>0.4925</td>
<td>0.4483</td>
<td>0.5306</td>
</tr>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 5.6: Results for $u$-velocity along the vertical line $x = 0.5$.

<table>
<thead>
<tr>
<th>x</th>
<th>Re=1</th>
<th>Re=10</th>
<th>Re=50</th>
<th>Re=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.0903</td>
<td>0.1647</td>
<td>0.3373</td>
<td>0.4211</td>
</tr>
<tr>
<td>0.1250</td>
<td>0.1467</td>
<td>0.2377</td>
<td>0.4151</td>
<td>0.4511</td>
</tr>
<tr>
<td>0.1875</td>
<td>0.1721</td>
<td>0.2715</td>
<td>0.3717</td>
<td>0.3710</td>
</tr>
<tr>
<td>0.2500</td>
<td>0.1724</td>
<td>0.2805</td>
<td>0.2934</td>
<td>0.2985</td>
</tr>
<tr>
<td>0.3125</td>
<td>0.1532</td>
<td>0.2610</td>
<td>0.2222</td>
<td>0.2262</td>
</tr>
<tr>
<td>0.3750</td>
<td>0.1192</td>
<td>0.2128</td>
<td>0.1537</td>
<td>0.1552</td>
</tr>
<tr>
<td>0.4375</td>
<td>0.0738</td>
<td>0.1440</td>
<td>0.0861</td>
<td>0.0851</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.0201</td>
<td>0.0654</td>
<td>0.0193</td>
<td>0.0156</td>
</tr>
<tr>
<td>0.5625</td>
<td>-0.0388</td>
<td>-0.0158</td>
<td>-0.0478</td>
<td>-0.0543</td>
</tr>
<tr>
<td>0.6250</td>
<td>-0.0984</td>
<td>-0.0988</td>
<td>-0.1159</td>
<td>-0.1254</td>
</tr>
<tr>
<td>0.6875</td>
<td>-0.1518</td>
<td>-0.1894</td>
<td>-0.1865</td>
<td>-0.1987</td>
</tr>
<tr>
<td>0.7500</td>
<td>-0.1890</td>
<td>-0.2959</td>
<td>-0.2598</td>
<td>-0.2748</td>
</tr>
<tr>
<td>0.8125</td>
<td>-0.1984</td>
<td>-0.4011</td>
<td>-0.3338</td>
<td>-0.3542</td>
</tr>
<tr>
<td>0.8750</td>
<td>-0.1700</td>
<td>-0.4904</td>
<td>-0.4263</td>
<td>-0.4340</td>
</tr>
<tr>
<td>0.9375</td>
<td>-0.1011</td>
<td>-0.2316</td>
<td>-0.5515</td>
<td>-0.6031</td>
</tr>
<tr>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Table 5.7: Results for $v$-velocity along the horizontal line $y = 0.5$. 
<table>
<thead>
<tr>
<th>$Re$</th>
<th>$(\frac{1}{16}, \frac{1}{32})$</th>
<th># Bicgstab in coarse</th>
<th># of Bicgstab in fine</th>
<th>residual on coarse</th>
<th>residual on fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>176,192,181,136</td>
<td>1055</td>
<td></td>
<td>$3.53 \times 10^{-6}$</td>
<td>$9.61 \times 10^{-7}$</td>
</tr>
<tr>
<td>10</td>
<td>363,423,479,470,493</td>
<td>2436</td>
<td></td>
<td>$8.00 \times 10^{-7}$</td>
<td>$2.00 \times 10^{-7}$</td>
</tr>
<tr>
<td>50</td>
<td>5695,6205,6673,5885,5560</td>
<td>34048</td>
<td></td>
<td>$2.93 \times 10^{-6}$</td>
<td>$5.49 \times 10^{-7}$</td>
</tr>
<tr>
<td>100</td>
<td>27290,26481,24830,18283</td>
<td>43560</td>
<td></td>
<td>$2.63 \times 10^{-7}$</td>
<td>$5.26 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Table 5.8: Two level for cavity flow
Figure 5.7: Streamlines for $H = \frac{1}{16}, h = \frac{1}{32}$ with different values of Re numbers using two level method.
Figure 5.8: Ghia-Ghia-Shin’s streamlines, 129 × 129, (courtesy U. Ghia [31]).
Figure 5.9: (a),(c) Streamlines for cavity flow using two level method, (b),(d) Akin’s streamlines, a mesh of $40 \times 40$ elements, (courtesy J. Akin [2]).
Figure 5.10: (above) $u$-velocity lines through the vertical line $x = 0.5$, (below) $v$-velocity lines through the horizontal line $y = 0.5$. 
Appendix A

FORTRAN 90 Program Using Morley Element
PROGRAM FINE_STEP_TEST
!------------------------------------------------------------------------
! THIS PROGRAM IS FOR SOLVING THE NAVIER-STOKES
! EQUATIONS USING THE TWO LEVEL FINITE ELEMENT
! METHOD ON THE STREAM FUNCTION FORMULATION
! FOR A FULL DESCRIPTION OF THE METHOD SEE
! F. FAIRAG, TWO-LEVEL FINITE ELEMENT METHOD
! FOR THE STREAM FUNCTION FORMULATION OF THE
! NAVIER-STOKES EQUATIONS, TO BE APPEAR IN
! INTERNATIONAL JOURNAL COMPUTERS AND MATHEMATICS
! WITH APPLICATIONS.
!
! THERE ARE TWO VERSION OF THIS CODE. THIS ONE
! USES THE MORLEY ELEMENT. THE OTHER USES THE
! BOGNER-FOX-SCHMEDIT ELEMENTS. FOR A DESCRIPTION ABOUT
! THESE ELEMENTS SEE THE ABOVE REFERENCE OR PH. G.
! CIARLET, THE FINITE ELEMENT METHOD FOR ELLIPTIC
!
! CONSIDER THE FOLLOWING EQUATION :
!------------------------------------------------------------------------
\begin{equation}\begin{split}
re^{-1} \bigtriangleup^{2} \psi - \psi_{y} \bigtriangleup \psi_{x} + \psi_{x} \bigtriangleup \psi_{y} & = \vec{\text{curl}} \vec{f}, \text{ in } \Omega, \\
\psi & = 0, \text{ on } \partial \Omega, \\
\frac{\partial \psi}{\partial \hat{n}} & = 0, \text{ on } \partial \Omega.
\end{split}\end{equation}
\text{where } \hat{n} \text{ represents the outward unit normal to } \Omega.$$
!------------------------------------------------------------------------
LAP (F) = LAPLACIAN OF F = F_{XX} + F_{YY}

R(F,G) = F \times \text{LAP}(G) - G \times \text{LAP}(F)

MORLEY TRINGLES:

- WHITHEN EACH TRINGLE THE FUNCTIONS ARE QUADRATIC POLYNOMIALS. THERE ARE SIX DEGREES OF FREEDOM ASSOCIATED WITH EACH TRINGLE, AND THESE ARE CHOSEN TO BE THE FUNCTION VALUE AT THE VERTICES OF THE TRINGLE AND THE NORMAL DERIVATIVE AT MIDSIDES.

SYMBOLS:

- LX = # OF ELEMENTS ALONG X-AXIS
- LXO = # OF ELEMENTS ALONG X-AXIS IN THE COARSE MESH
- RE = REYNOLDS NUMBER
- NELEM = # OF ELEMENTS
- NPTS = # OF POINTS
- NBO = # OF BOUNDARY POINTS
- NBF = # OF BASIS FUNCTIONS
- M = # OF DEGREES OF FREEDOM
- NNZ = # OF NONZERO ENTRIES IN THE JACOBIAN MATRIX
- MAX_NWTN = MAXIMUM NUMBER OF NEWTON ITERATIONS TO SOLVE THE NONLINEAR SYSTEM RESULTING FROM COARSE MESH
ARRAYS :

IELNODE ( NELEM,NBF ) = GLOBAL NODE NUMBERS FOR ALL ELEMENTS

IBO ( NBO ) = THE SET OF NODE NUMBERS OF NODES ON BOUNDARY.

XX ( M ) = X-COORDINATES OF ALL NODES

YY ( M ) = Y-COORDINATES OF ALL NODES

XI ( 7 ) = X-COORDINATES OF INTEGRATION POINTS ON THE TRIANGLE (0,0),(1,0),(0,1)

ETA ( 7 ) = Y-COORDINATES OF INTEGRATION POINTS ON THE TRIANGLE (0,0),(1,0),(0,1)

W ( 7 ) = THE CORRESPONDING INTEGRATION WEIGHTS

PHI( NBF , NQPT ) = VALUES OF BASIS FUNCTIONS AT THE INTEGRATION POINTS.

PHIX( NBF , NQPT ) = X-DERIVATIVE VALUES OF BASIS FUNCTIONS AT THE INTEGRATION POINTS.

PHIY( NBF , NQPT ) = Y-DERIVATIVE VALUES OF BASIS FUNCTIONS AT THE INTEGRATION POINTS.

--------------------------------------------------

USE MSIMSL
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
PARAMETER( LXO=12,LX=2*LXO, &
NBF=6,NQPT=7, &
MAX_NWTN = 100,TOL=0.0000001D0, &
MO=(2*LX0+1)**2,M=(2*LX+1)**2, &

MO=(2*LX0+1)**2,M=(2*LX+1)**2, &

MO=(2*LX0+1)**2,M=(2*LX+1)**2, &
NNZ = 46*LX**2 + 16*LX + 1, NBO = 8*LX, &
NELEM = 2*LX*LX, NELEMO = 2*LXO*LXO, &
NBOO = 8*LXO, NPT = 2*LX + 1, &
NPTO = 2*LXO + 1, NIN = M - NBO, NINO = MO - NBOO &
)
DIMENSION IA(NNZ), JA(NNZ), IELNODE(NELEM, NBF), &
      A(NNZ), XX(M), YY(M), SOLO(MO), EA(NBF, NBF), &
      B(NBF, NBF, NBF), IBO(NBO), IN(NIN), &
      XI(NQPT), ETA(NQPT), W(NQPT), &
      PHI(NBF, NQPT), PHIX(NBF, NQPT), &
      PHIY(NBF, NQPT), SOLN(M), RHS(M), &
      ER(MAX_NWTN, 2), VRE(5)
DIMENSION INO(NINO), IBOO(NBOO), IELNODEO(NELEMO, NBF)
!
!
VRE = (/1.0D0, 100.0D0, 100.0D0, 1000.0D0, 2000.0D0/)
DO IRE = 1, 1
   RE = VRE(IRE)
!
!IF(RE == 10.0) THEN
   OPEN(UNIT= 32, FILE="SOLO.DAT", STATUS="OLD")
   READ(32, *) SOLO
!ELSE IF (RE == 50.0) THEN
!   OPEN(UNIT= 32, FILE="SOLO50.DAT", STATUS="OLD")
!   READ(32, *) SOLO
!ELSE IF (RE == 100.0) THEN
!   OPEN(UNIT= 32, FILE="SOLO100.DAT", STATUS="OLD")
!   READ(32, *) SOLO
!ELSE IF (RE == 1000.0) THEN
!   OPEN(UNIT= 32, FILE="SOLO1000.DAT", STATUS="OLD")
!   READ(32, *) SOLO
!ELSE
!   OPEN(UNIT= 32, FILE="SOLO2000.DAT", STATUS="OLD")
!   READ(32, *) SOLO
CALL MAELNODE(IELNODE, IN, IBO, LX, NBO, NIN, M, NELEM, NBF, NPT)
CALL MAELNODE(IELNODEO, INO, IBOO, LXO, NBOO, NINO, MO, NELEMO, NBF, NPTO)
CALL COORD(XX, YY, LX, M)
CALL MAEA(EA, NBF)
CALL QUAD7(NQPT, XI, ETA, W)
CALL PHIQ(PHI, PHIX, PHIY, NBF, NQPT, XI, ETA)
CALL MAB(B, NBF)

H = 1.0D0/DBLE(LX)
IA = 0 ; JA = 0 ; A = 0.0D0

TIME1 = CPSEC()
CALL FINE_LEVEL_SOLVE(SOLN, &
                      SOLO, IELNODE, NELEMO, &
                      LX, LXO, RE, TOL, M, MO, IN, IBO, NIN, &
                      NBO, NBF, NELEM, XX, YY, PHI, PHIX, &
                      PHIY, XI, ETA, W, NQPT, EA, NNZ, B, &
                      IELNODEO, KK)

TIME2 = CPSEC()
XCPU = TIME2 - TIME1
CALL PRINTOUT1(SOLN, ER, KK, XCPU, &
               LX, RE, TOL, MAX, M, NIN, NQPT, &
               NBO, NBF, NELEM, NNZ)

PRINT*,' I FINISH SUCCESSFULLAY'
END DO
END PROGRAM FINE_STEP_TEST

SUBROUTINE FINE_LEVEL_SOLVE(SOLN, &
                           SOLO, IELNODE, NELEMO, &
                           LX, LXO, RE, TOL, M, MO, IN, IBO, NIN, &
                           NBO, NBF, NELEM, XX, YY, PHI, PHIX, &
                           PHIY, XI, ETA, W, NQPT, EA, NNZ, B, &
                           IELNODEO, KK)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION SOLN(M),IELNODE(NELEM,NBF),SOLO(MO),
     IBO(NBO),IN(NIN),XX(M),
     YY(M),PHI(NBF,NQPT),PHIX(NBF,NQPT),
     PHIY(NBF,NQPT),XI(NQPT),ETA(NQPT),
     W(NQPT),EA(NBF,NBF),
     B(NBF,NBF,NBF),IELNODEO(NELEMO,NBF)

INTENT(OUT) :: SOLN
INTENT(IN) :: IELNODE,IN,LX,RE,TOL,M,
     IBO,NIN,NBO,NBF,NELEM,XX,YY,
     PHI,PHIX,PHIY,XI,ETA,W,NQPT,
     EA,B,SOLO,IELNODEO

! LOCAL
DIMENSION A(NNZ),IA(NNZ),JA(NNZ),RHS(M)
!
!
CALL STIFF(A,IA,JA,NNZ,LX,NBF,M,RE,IELNODE,XX,YY,&
     LXO,MO,IELNODEO,SOLO,EA,B,&
     NELEM,NELEMO)
!
!
CALL RHS2L(RHS,LX,M,NIN,NBO,RE,&
     NBF,IELNODE,IN,IBO,&
     XX,YY,PHI,PHIX,PHIY,&
     XI,ETA,W,NQPT,NELEM)
!
!
CALL BOUNDARY_A(A,IA,JA,IBO,NNZ,NBO)
CALL BOUNDARY_RHS(RHS,IBO,M,NBO)
!
!
IF(RE == 10.0) THEN
  OPEN(UNIT= 55,FILE="A10.DAT",STATUS="NEW")
  OPEN(UNIT= 56,FILE="RHS10.DAT",STATUS="NEW")
ELSE IF( RE == 50.0 ) THEN
  OPEN(UNIT= 55,FILE="TA50L32.DAT",STATUS="NEW")
  OPEN(UNIT= 56,FILE="TRHS50L32.DAT",STATUS="NEW")
ELSE IF( RE == 100.0 ) THEN
  OPEN(UNIT= 55,FILE="TA100L32.DAT",STATUS="NEW")
  OPEN(UNIT= 56,FILE="TRHS100L32.DAT",STATUS="NEW")
ELSE IF( RE == 1000.0 ) THEN
  OPEN(UNIT= 55,FILE="A1000.DAT",STATUS="NEW")
  OPEN(UNIT= 56,FILE="RHS1000.DAT",STATUS="NEW")
ELSE
OPEN(UNIT= 55,FILE="A2000.DAT",STATUS="NEW")
OPEN(UNIT= 56,FILE="RHS2000.DAT",STATUS="NEW")
END IF
DO I=1,NNZ
  WRITE(55,*) A(I),IA(I),JA(I)
END DO
DO I=1,M
  WRITE(56,*) RHS(I)
END DO
!
SOLN=0.0D0
CALL BICGSTAB(SOLN,KK,RHS,TOL,A,IA,JA,NNZ,M,1)
!
!
END SUBROUTINE FINE_LEVEL_SOLVE
!
!
SUBROUTINE PRINTOUT1(SOL,ER,KK,CPU, &
  LX,RE,TOL,MAX,M,NIN,NQPT, &
  NBO,NBF,NELEM,NNZ)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION SOL(M),ER(MAX,2),ITER(MAX)
INTENT(IN) :: SOL,ER,KK,CPU,LX, &
  RE,TOL,MAX,M,NIN,NQPT, &
  NBO,NBF,NELEM, &
  NNZ
IF(RE == 10.0) THEN
  OPEN(UNIT= 31,FILE="STATN10.DAT",STATUS="NEW")
  OPEN(UNIT= 32,FILE="SOLN10.DAT",STATUS="NEW")
ELSE IF( RE == 50.0 ) THEN
  OPEN(UNIT= 31,FILE="TS50L32.DAT",STATUS="NEW")
  OPEN(UNIT= 32,FILE="TSOL50L32.DAT",STATUS="NEW")
ELSE IF( RE == 100.0 ) THEN
  OPEN(UNIT= 31,FILE="TS100L32.DAT",STATUS="NEW")
  OPEN(UNIT= 32,FILE="TSOL100L32.DAT",STATUS="NEW")
ELSE IF( RE == 1000.0 ) THEN
  OPEN(UNIT= 31,FILE="STATN1000.DAT",STATUS="NEW")
  OPEN(UNIT= 32,FILE="SOLN1000.DAT",STATUS="NEW")
ELSE
  OPEN(UNIT= 31,FILE="STATN2000.DAT",STATUS="NEW")
  OPEN(UNIT= 32,FILE="SOLN2000.DAT",STATUS="NEW")
END IF
WRITE(31,*) ' ------------------------------------'
WRITE(31,*)
WRITE(31,*) ' LX = ', LX
WRITE(31,*)
WRITE(31,*) ' RE = ', RE
WRITE(31,*)
WRITE(31,*) ' # ELEMENTS = ', NELEM
WRITE(31,*)
WRITE(31,*) ' # OF NODES(M) = ', M
WRITE(31,*)
WRITE(31,*) ' # OF BOUNDARY NODES(NBO) = ',NBO
WRITE(31,*)
WRITE(31,*) ' # OF INTIRIOR NODES(NIN) = ',NIN
WRITE(31,*)
WRITE(31,*) ' # OF BASIS FUNCTIONS(NBF) = ',NBF
WRITE(31,*)
WRITE(31,*) ' # OF QUADRATURE POINTS(NQPT) = ',NQPT
WRITE(31,*)
WRITE(31,*) ' # OF MAXIMUM NEWTON ITERATIONS(MAX) = ',MAX
WRITE(31,*)
WRITE(31,*) ' # OF NONZERO ENTRY IN THE JACOBIAN(NNZ) = ',NNZ
WRITE(31,*)
WRITE(31,*) ' THE TOLERANCE = ',TOL
WRITE(31,*)
WRITE(31,*) ' TOTAL CPU TIME IN SECONDS = ', CPU
WRITE(31,*)
WRITE(31,*) ' ------------------------------------'
WRITE(31,*)
WRITE(31,*) ' #BCGS = ',KK
WRITE(31,*)
WRITE(31,*) ' ------------------------------------'

DO I=1,M
  WRITE(31,*) SOL(I)
  WRITE(32,*) SOL(I)
END DO
!
END SUBROUTINE PRINTOUT1
!
!
SUBROUTINE COARSE_LEVEL_SOLVE( &
  SOL,SOLNODE,ER,ITER,CPU, &
  LX,RE,TOL,MAX,M,IN,IBO,NIN, &
  NBO,NBF,NELEM,XX,YY,PHI,PHIX, &
  PHIY,XI,ETA,W,NQPT,EA,NNZ,B &
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION SOL(M),IELNODE(NELEM,NBF), &
    ITER(MAX),IBO(NBO),IN(NIN),XX(M), &
    YY(M),PHI(NBF,NQPT),PHIX(NBF,NQPT), &
    PHIY(NBF,NQPT),XI(NQPT),ETA(NQPT), &
    W(NQPT),EA(NBF,NBF),ER(MAX,2), &
    B(NBF,NBF,NBF)
INTENT(INOUT) :: SOL
INTENT(OUT) :: ER,ITER,CPU
INTENT(IN) :: IELNODE,IN,LX,RE,TOL,MAX,M, &
    IBO,NIN,NBO,NBF,NELEM,XX,YY, &
    PHI,PHIX,PHIY,XI,ETA,W,NQPT, &
    EA,B
! LOCAL
DIMENSION UP(M),A(NNZ),IA(NNZ),JA(NNZ),RHS(M)
UP = SOL
ER1 = 1.0D0 ; XNORM = 0.0D0 ; YNORM = 0.0D0
DO I=1,MAX
    CALL JACOBS(LX,M,NNZ,NELEM,NIN,NBO, &
        NBF,A,IA,JA,RE,SOL,IELNODE, &
        IN,IBO,EA,B)
    CALL FORCE(RHS,LX,RE,SOL,IELNODE, &
        IN,IBO,XX,YY,PHI,PHIX, &
        PHIY,XI,ETA,W,EA,NBF, &
        NELEM,M,NIN,NBO,NQPT,B)
    IF( RATIO >= 1) THEN
        PRINT*,’ NO IMPROVEMENT FOR F(SOL)’
        EXIT
    END IF
    CALL BOUNDARY_A(A,IA,JA,IBO,NNZ,NBO)
    CALL BOUNDARY_RHS(RHS,IBO,M,NBO)
    CALL NORM(XNORM,RHS,M)
    OLDER1=ER1
    ER1 = XNORM/DBLE(M)
    RATIO = ER1/OLDER1
    CALL BICGSTAB(UP,KK,RHS,TOL,A,IA,JA,NNZ,M,I)
    ITER(I)=KK
    SOL = SOL - UP
    CALL NORM(YNORM,UP,M)
ER2 = YNORM/DBLE(M)
ER(I,:)=(/ER1,ER2/)
PRINT*,I,ER(I,:)
IF(ER1<TOL .AND. ER2<TOL)THEN;EXIT;END IF
END DO
!
!
END SUBROUTINE COARSE_LEVEL_SOLVE
!
!
! STATUS = NOT OK
! FIX HOW TO STORE A , IA , JA
SUBROUTINE STIFF(A,IA,JA,NNZ,LX,NBF,M,RE,IELNODE,XX,YY,&
LXO,MO,IELNODEO,SOLO,EA,B,&
NELEM,NELEMO)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION IA(NNZ),JA(NNZ),IELNODE(NELEM,NBF),&
IELNODEO(NELEMO,NBF)
DIMENSION A(NNZ),XX(M),YY(M),SOLO(MO),EA(NBF,NBF)
DIMENSION B(NBF,NBF,NBF)
INTENT(IN) :: NNZ,LX,NBF,M,RE,IELNODE,XX,YY,&
LXO,MO,IELNODEO,SOLO,EA,B, &
NELEM,NELEMO
INTENT(OUT) :: IA,JA,A
! LOCAL
DIMENSION X(3),Y(3)
!
! --------------------------------------
! COMPUTE SOME PARAMETERS OF PROBLEMS
! DEPENDS ON LX
! --------------------------------------
CONT=0.0DO
H = 1.0DO / DBLE(LX)
RATIO=DBLE(LX)/DBLE(LXO)
!----------------ASSIMBLE STIFF ---------
INDEX =0
A = 0.0DO; IA = 0 ; JA=0
X=0.0DO; Y=0.0DO
DO K=1,NELEM
   DO IX=1,3
      IP= IABS( IELNODE(K,IX))
      X(IX) = XX(IP)
      Y(IX) = YY(IP)
   END DO
END DO
XM = (X(2) + X(3) + 2.0D0 * X(1)) / 4.0D0
YM = (Y(2) + Y(3) + 2.0D0 * Y(1)) / 4.0D0
DO IR = 1, NBF
   AIJ = 0.0D0
   II = IELNODE(K, IR)
   IROW = ABS(II)
   DO IS = 1, NBF
      JJ = IELNODE(K, IS)
      D = DBLE(ISIGN(1, II) * ISIGN(1, JJ))
      JCOL = IABS(JJ)
      CONT = (D/RE) * EA(IR, IS) / (H * H)
      ! AF(IROW, JCOL) = AF(IROW, JCOL) + CONT
      ! AIJ = AIJ + (D/RE) * EA(IR, IS) / (H * H)
      CALL STORE(A, IA, JA, NNZ, IROW, JCOL, CONT, INDEX)
   END DO
   SUM = 0.0D0
   DO IK = 1, NBF
      CALL FINDK(XM, YM, LXO, K) ! K = FINDK
      K3 = IELNODEO(K, IK)
      KK = IABS(K3)
      D = DBLE(SIGN(1, II) * SIGN(1, JJ) * SIGN(1, K3))
      SUM = SUM + D * SOLO(KK) * B(IK, IR, IS)
   END DO
   SUM = SUM / (RATIO * H**2)
   ! AF(IROW, JCOL) = AF(IROW, JCOL) + SUM
   ! AIJ = AIJ + SUM
   CALL STORE(A, IA, JA, NNZ, IROW, JCOL, SUM, INDEX)
   END DO
END DO
END SUBROUTINE STIFF
!
!
SUBROUTINE BOUNDARY_A(A, IA, JA, IBO, NNZ, NBO)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION A(NNZ), IA(NNZ), JA(NNZ), IBO(NBO)
INTENT(INOUT) :: A, IA, JA
INTENT(IN) :: NNZ, NBO, IBO
!
DO IZ = 1, NBO
   DO IH = 1, NNZ
      IF (IA(IH) == IBO(IZ)) A(IH) = 0.0D0
   END DO
END DO
END SUBROUTINE BOUNDARY_A
IF(JA(IH)==IBO(IZ)) A(IH)=0.0D0
END DO
END DO

DO IZ = 1,NBO
DO IH = 1,NNZ
IF(IA(IH)==IBO(IZ) .AND. JA(IH)==IBO(IZ)) THEN
A(IH)=1.0D0
END IF
END DO
END DO
!
END SUBROUTINE BOUNDARY_A
!
!
SUBROUTINE BOUNDARY_RHS(RHS,IBO,M,NBO)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),IBO(NBO)
INTENT(INOUT) :: RHS
INTENT(IN) :: M,NBO,IBO
!
DO IR = 1,NBO
RHS( IBO(IR) ) = 0.0D0
END DO
!
END SUBROUTINE BOUNDARY_RHS
!
!
!-----------------------------------------
SUBROUTINE BICGSTAB(X,K,B,TOL,C,IC,JC,NNZ,M,NWTN)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(M),B(M),C(NNZ),IC(NNZ),JC(NNZ)
INTENT(IN) :: B,TOL,C,IC,JC,NNZ,M,NWTN
INTENT(INOUT) :: X
INTENT(OUT) :: K
! LOCAL
DIMENSION R(M),RH(M),P(M),V(M),S(M),T(M),UP(M)
K=0
R=0.0D0;RH=0.0D0;P=0.0D0;V=0.0D0;S=0.0D0
T=0.0D0;UP=0.0D0;RA=0.0D0
CALL ATX(R,C,X,IC,JC,NNZ,M) ! R=C*X
R = B-R
RH = R
DO I=1,M*M
  PRINT*,NWTN ,'-TH NEWTON',I,'-TH BICG'
  CALL DOT(ROW,RH,R,M)
  IF(ROW == 0 ) EXIT
  IF(I == 1) THEN
    P = R
  ELSE
    BETA = (ROW/ROWOLD)*(ALPHA/W)
    P = R + BETA*(P-W*V)
  END IF
  P = P
  CALL ATX(V,C,P,IC,JC,NNZ,M) ! V=C*P
  CALL DOT(VAL,RH,V,M) ! VAL = RH'*V
  ALPHA = ROW/VAL
  S = R - ALPHA*V
  CALL NORM(XNORM,S,M)
  IF( XNORM < TOL) THEN
    K=I
    X=X+ALPHA*P
    PRINT*, 'I STOP BECAUSE NORM(S)<TOL'
    EXIT
  END IF
  S = S
  CALL ATX(T,C,S,IC,JC,NNZ,M) ! T = C*S
  CALL DOT(VAL2,T,S,M)
  CALL DOT(VAL3,T,T,M)
  W = VAL2/VAL3
  UP = ALPHA*P +W*S
  X = X + UP
  R = S - W*T
  IF(W == 0 ) THEN
    PRINT*, 'I CAN NOT CONTINUE ITERATION BECAUSE W = 0'
    EXIT
  END IF
  CALL NORM(XUP,UP,M)
  CALL NORM(XR,R,M)
  IF( XUP < TOL .AND. XR < TOL ) THEN
    K = I
    PRINT*, 'I STOP BECAUSE NORM(R,UP)<TOL'
    EXIT
  END IF
  ROWOLD=ROW
END DO
END SUBROUTINE BICGSTAB
SUBROUTINE DOT(VAL, X, Y, M)
INTEGER I, M
DOUBLE PRECISION X(M), Y(M), VAL
INTENT(IN) :: X, Y, M
INTENT(OUT) :: VAL
VAL = 0.0D0
DO I = 1, M
   VAL = VAL + X(I) * Y(I)
END DO
END SUBROUTINE DOT

SUBROUTINE NORM(VAL, X, M)
INTEGER I, M
DOUBLE PRECISION VAL, X(M)
INTENT(IN) :: X, M
INTENT(OUT) :: VAL
VAL = 0.0D0
DO I = 1, M
   VAL = VAL + X(I) * X(I)
END DO
VAL = DSQRT(VAL)
END SUBROUTINE NORM

SUBROUTINE ATX(Y, A, X, IA, JA, NNZ, M)
INTEGER I, J, K, M, NNZ
INTEGER IA(NNZ), JA(NNZ)
DOUBLE PRECISION X(M), Y(M), A(NNZ), AIJ
INTENT(IN) :: A, X, IA, JA, NNZ, M
INTENT(OUT) :: Y
Y = 0.0D0
DO K = 1, NNZ
   I = IA(K)
   J = JA(K)
   AIJ = A(K)
   Y(I) = Y(I) + AIJ * X(J)
END DO
END SUBROUTINE ATX

SUBROUTINE ATTX(Y, A, X, IA, JA, NNZ, M)
INTEGER I,J,K,M,NNZ
INTEGER IA(NNZ),JA(NNZ)
DOUBLE PRECISION X(M),Y(M),A(NNZ),AJI
INTENT(IN) :: A,X,IA,JA,NNZ,M
INTENT(OUT) :: Y
Y=0.0D0
DO K=1,NNZ
  J=IA(K)
  I=JA(K)
  AJI=A(K)
  Y(I)= Y(I) + AJI*X(J)
END DO
END SUBROUTINE ATTX

STATUS( STORE ) =OK
SUBROUTINE STORE(A,IA,JA,NNZ,IROW,JCOL,AIJ,INDEX)
DOUBLE PRECISION A(NNZ),AIJ
INTEGER IA(NNZ),JA(NNZ),IROW,JCOL,IFLAG
INTEGER INDEX
INTENT(INOUT) :: A,IA,JA,INDEX
INTENT(IN) :: IROW,JCOL,NNZ,AIJ
CALL CHK(NNZ,IA,JA,IROW,JCOL,IFLAG)
IF( IFLAG == 0) THEN
  INDEX = INDEX + 1
  A(INDEX) = AIJ
  IA(INDEX) = IROW
  JA(INDEX) = JCOL
ELSE
  A(IFLAG) = A(IFLAG) + AIJ
END IF
END SUBROUTINE STORE

STATUS( CHK ) = OK
SUBROUTINE CHK(NNZ,IA,JA,NNZ,IROW,JCOL,IFLAG)
INTEGER NNZ,IA(NNZ),JA(NNZ),JCOL,IROW
INTENT(IN) :: NNZ,IA,JA,IROW,JCOL
INTENT(OUT) :: IFLAG
INTEGER IFLAG,I
IFLAG = 0
DO I=1,NNZ
IF( IA(I) == IROW .AND. JA(I) == JCOL ) THEN
  IFLAG = I
EXIT
END IF
END DO
END SUBROUTINE CHK

! STATUS( FINDK ) = OK
!
SUBROUTINE FINDK(X,Y,LX,K)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
INTENT(IN) :: X,Y,LX
INTENT(OUT) :: K
H = 1.0D0/DBLE(LX)
XK = 0.0D0 ; YK = 0.0D0 ;
I =LX; J=LX;
DO IK= 1,LX
  XK=XK+H
  IF( XK > X ) THEN
    I=IK
  EXIT
  END IF
END DO
DO JK =1,LX
  YK=YK +H
  IF( YK > Y ) THEN
    J=JK
  EXIT
  END IF
END DO
SW = Y + X - H*DBLE(J+I-1)
K=2*( (J-1)*LX +I ) - 1
IF( SW > 0 ) THEN
  K=K+1
END IF
END SUBROUTINE FINDK

! STATUS( RHS2L ) = OK
!
SUBROUTINE RHS2L(RHS,LX,M,NIN,NBO,RE, &
                 NBF,IENODE,IN,IBO,&
                 XX,YY,PHI,PHIX,PHIY, &
                 XI,ETA,W,NQPT,NELEM )
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),IELNODE(NELEM,NBF),IN(NIN), &
   IBO(NBO),XX(M),YY(M),PHI(NBF,NQPT), &
   PHIX(NBF,NQPT),PHIY(NBF,NQPT), &
   XI(NQPT),ETA(NQPT),W(NQPT)
INTENT(IN) :: IELNODE,IN,IBO,XX,YY,PHI,PHIX,PHIY, &
   XI,ETA,W,LX,M,NIN,NBO,RE,NBF, &
   NQPT,NELEM
INTENT(OUT) :: RHS
!
! LOCAL
DIMENSION X(3),Y(3)
H=1.0D0/DBLE(LX)
RHS = 0.0D0
DO NK=1,NELEM
   DO II=1,3
      X(II) = XX( IABS(IELNODE(NK,II)) )
   Y(II) = YY( IABS(IELNODE(NK,II)) )
   END DO
   ! --------------------------------------
   DO IR=1,6
      II=IELNODE(NK,IR)
      I =IABS(II)
      D= DBLE(SIGN(1,II))
      SUM1=0 ; SUM2= 0
      DO IZ=1,7
         S=(X(2)-X(1))*XI(IZ) + X(1)
         T=(Y(3)-Y(1))*ETA(IZ) + Y(1)
         CALL F1(S,T,RE,F1ST)
         CALL F2(S,T,RE,F2ST)
         SUM1=SUM1 + W(IZ)*PHIY(IR,IZ)*F1ST
         SUM2=SUM2 + W(IZ)*PHIX(IR,IZ)*F2ST
      END DO
      VALUE =-SUM1*( X(2)-X(1) ) + SUM2*( Y(3)-Y(1) )
      RHS(I) = RHS(I) + D*VALUE
   END DO
END DO
!
!
END SUBROUTINE RHS2L
!
! STATUS = OK
!
SUBROUTINE F1(X,Y,RE,F1XY)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
INTENT(IN) X,Y,RE
INTENT(OUT) F1XY

\[ D_1 = 12.0D0 \cdot (-1.0D0 + X)^2 \cdot X^2 \cdot (-1.0D0 + Y) \]
\[ D_2 = 12.0D0 \cdot (-1.0D0 + X)^2 \cdot X^2 \cdot Y \]
\[ D_3 = 4.0D0 \cdot (-1.0D0 + X)^2 \cdot (-1.0D0 + Y)^2 \]
\[ D_4 = 16.0D0 \cdot (-1.0D0 + X) \cdot X \cdot (-1.0D0 + Y)^2 \cdot Y \]
\[ D_5 = 4.0D0 \cdot X^2 \cdot (-1.0D0 + Y)^2 \]
\[ D_6 = 4.0D0 \cdot (-1.0D0 + X)^2 \cdot (-1.0D0 + Y) \cdot Y^2 \]
\[ D_7 = 16.0D0 \cdot (-1.0D0 + X) \cdot X \cdot (-1.0D0 + Y) \cdot Y^2 \]
\[ D_8 = 4.0D0 \cdot X^2 \cdot (-1.0D0 + Y) \cdot Y^2 \]
\[ D = -(D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + D_7 + D_8) / RE \]

\[ E_1 = 3.0D0 + 4.0D0 \cdot X \cdot Y^2 - 20.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_2 = 36.0D0 \cdot X \cdot Y^2 - 8.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_3 = 8.0D0 \cdot X^2 \cdot Y^2 - 16.0D0 \cdot X \cdot Y^2 \]
\[ E_4 = 80.0D0 \cdot X^2 \cdot Y^2 - 144.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_5 = 112.0D0 \cdot X^2 \cdot Y^2 - 32.0D0 \cdot X \cdot Y^2 \]
\[ E_6 = 28.0D0 \cdot X \cdot Y^2 - 140.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_7 = 252.0D0 \cdot X^2 \cdot Y^2 - 196.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_8 = 56.0D0 \cdot X \cdot Y^2 - 40.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_9 = 72.0D0 \cdot X \cdot Y^2 - 56.0D0 \cdot X^2 \cdot Y^2 \]
\[ E_10 = 16.0D0 \cdot X \cdot Y^2 - 140.0D0 \cdot X^2 \cdot Y^2 \]
\[ E = X^2 \cdot (E_1 + E_2 + E_3 + E_4 + E_5 + E_6 + E_7 + E_8 + E_9 + E_10 + E_11 + E_12 + E_13) \]
\[ F1XY = D + E \]

END SUBROUTINE F1

SUBROUTINE F2(X,Y,RE,F2XY)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
INTENT(IN) :: X,Y,RE
INTENT(OUT) :: F2XY

\[ D_1 = 4.0D0 \cdot (-1.0D0 + X)^2 \cdot X \cdot (-1.0D0 + Y)^2 \]
\[ D_2 = 4.0D0 \cdot (-1.0D0 + X)^2 \cdot X \cdot (-1.0D0 + Y)^2 \]
\[ D_3 = 16.0D0 \cdot (-1.0D0 + X)^2 \cdot X \cdot (-1.0D0 + Y) \]
\[ D_4 = 16.0D0 \cdot (-1.0D0 + X)^2 \cdot (-1.0D0 + Y) \]
\[ D_5 = 4.0D0 \cdot (-1.0D0 + X)^2 \cdot Y \cdot (-1.0D0 + Y) \]
\[ D_6 = 4.0D0 \cdot (-1.0D0 + X)^2 \cdot Y \cdot (-1.0D0 + Y) \]
\[ D_7 = 12.0D0 \cdot (-1.0D0 + X) \cdot (-1.0D0 + Y) \]

END SUBROUTINE F2
\[ D8 = 12.0 \times 10^{-1} \times (-1.0 + Y)^2 \times Y^2 \]

\[ D = -(-D1 - D2 - D3 - D4 - D5 - D6 - D7 - D8) / RE \]

\[ E1 = 3.0 \times 10^1 + 4.0 \times 10^1 \times X^2 \times Y - 16.0 \times 10^3 \times X^3 \times Y \]

\[ E2 = 28.0 \times 10^1 \times X^4 \times Y - 24.0 \times 10^1 \times X^5 \times Y + 8.0 \times 10^1 \times X^6 \times Y \]

\[ E3 = -20.0 \times 10^1 \times X^2 \times Y^2 - 80.0 \times 10^1 \times X^3 \times Y^2 \]

\[ E4 = -140.0 \times 10^1 \times X^4 \times Y^2 + 120.0 \times 10^1 \times X^5 \times Y^2 \]

\[ E5 = -40.0 \times 10^1 \times X^6 \times Y^2 + 36.0 \times 10^1 \times X^2 \times Y^3 \]

\[ E6 = -144.0 \times 10^1 \times X^3 \times Y^3 + 252.0 \times 10^1 \times X^4 \times Y^3 \]

\[ E7 = -216.0 \times 10^1 \times X^5 \times Y^3 + 72.0 \times 10^1 \times X^6 \times Y^3 \]

\[ E8 = -28.0 \times 10^1 \times X^2 \times Y^4 - 112.0 \times 10^1 \times X^3 \times Y^4 \]

\[ E9 = -196.0 \times 10^1 \times X^4 \times Y^4 + 168.0 \times 10^1 \times X^5 \times Y^4 \]

\[ E10 = -56.0 \times 10^1 \times X^6 \times Y^4 + 8.0 \times 10^1 \times X^2 \times Y^5 \]

\[ E11 = -32.0 \times 10^1 \times X^3 \times Y^5 + 56.0 \times 10^1 \times X^4 \times Y^5 \]

\[ E12 = -48.0 \times 10^1 \times X^5 \times Y^5 + 16.0 \times 10^1 \times X^6 \times Y^5 \]

\[ E = Y^2 \times (E1 + E2 + E3 + E4 + E5 + E6 + E7 + E8 + E9 + E10 + E11 + E12) \]

\[ F2XY = D + E \]

END SUBROUTINE F2

! STATUS = OK

SUBROUTINE MAELNODE(IELNODE, IN, IBO, LX, NBO, NIN, M, NELEM, NBF, NPT)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
INTENT(IN) :: LX, NBO, NIN, NBF, NPT
INTENT(OUT) :: IELNODE, IN, IBO
DIMENSION IELNODE(NELEM, NBF), IN(NIN), IBO(NBO)
IELNODE=0; IN=0; IBO=0
IELEMN=0
!
! ODD TRAIANGLES
DO I=1, LX
  DO J=1, LX
    IELEMN=2*J-1 + 2*LX*(I-1)
    K = 2*J-1 + 2*NPT*(I-1)
    IELNODE(IELEMN,1)= K
    IELNODE(IELEMN,2)= K+2
    IELNODE(IELEMN,3)= K+2*NPT
    IELNODE(IELEMN,4)= K+1
    IELNODE(IELEMN,5)= K+1+NPT
    IELNODE(IELEMN,6)= K+NPT
  END DO
END DO

! EVEN TRAIANGLES
!------------------
!------------------
DO I=1,LX
  DO J=1,LX
    IELEMN=2*J +2*LX*(I-1)
    K = 2*NPT*I +2*J+1
    IELNODE(IELEMN,1)= K
    IELNODE(IELEMN,2)= K-2
    IELNODE(IELEMN,3)= K-2*NPT
    IELNODE(IELEMN,4)= K-1
    IELNODE(IELEMN,5)= K-1-NPT
    IELNODE(IELEMN,6)= K-NPT
  END DO
END DO
DO I=1,LX*LX
  IELNODE(2*I,4) = - IELNODE(2*I,4)
  IELNODE(2*I,5) = - IELNODE(2*I,5)
  IELNODE(2*I,6) = - IELNODE(2*I,6)
END DO
!
! CHANGE NORMAL POINTS ON EVEN TRIANGLE
! ON X=1
DO I=1,LX
  IELEMN= 2*LX*I
  IELNODE(IELEMN,6) = - IELNODE(IELEMN,6)
END DO
!
! CHANGE NORMAL POINTS ON Y=1
IELEMN = 2*LX*LX
DO I=1,LX
  IELEMN= IELEMN - 2*(I-1)
  IELNODE(IELEMN,4) = - IELNODE(IELEMN,4)
END DO
!

DO I=1,NPT
  IBO(I) = I
  IBO(I+NPT) = M-I+1
END DO
DO I=2,NPT-1
  IBO(I+2*NPT-1) = I*NPT
  IBO(3*NPT-3+I) = (I-1)*NPT + 1
END DO
!
INDEX=1
DO I=2,NPT-1
  KK=(I-1)*NPT +1
  DO J=1,NPT-2
    IN(INDEX) = KK +J
  INDEX = INDEX + 1
END DO
END DO
END SUBROUTINE MAELNODE
!
!
! STATUS = OK
!
SUBROUTINE COORD(X,Y,LX,M)
DOUBLE PRECISION, INTENT(OUT) :: X(M),Y(M)
INTEGER, INTENT(IN) :: LX,M
DOUBLE PRECISION H,HH
INTEGER NPT,I,J,N
H=1.0D0/DBLE(LX)
HH=H/2.0D0
NPT = 2*LX+1
X=0.0D0
Y=0.0D0
DO I=1,NPT
   DO J=1,NPT
      N=J + (I-1)*NPT
      Y(N) = (I-1)*HH
      X(N) = (J-1)*HH
   END DO
END DO
END SUBROUTINE COORD
!
!
SUBROUTINE QUAD7(NQPT,XI,ETA,W)
DOUBLE PRECISION, INTENT(OUT) :: XI(NQPT),ETA(NQPT),W(NQPT)
DOUBLE PRECISION THD
INTEGER, INTENT(IN) :: NQPT
! FOR GAUSS QUADRATUAL WITH 7-POINTS
! EXACT WITH 4 DEGREES POLY
THD = 1.0D0/3.0D0
XI =(/0.0D0,.50D0,1.0D0,.50D0,0.0D0,0.0D0,THD/)
ETA=(/0.0D0,0.0D0,0.0D0,.50D0,1.0D0,.50D0,THD/)
W =(/1.0D0/40.0D0,1.0D0/15.0D0,1.0D0/40.0D0, &
   1.0D0/15.0D0,1.0D0/40.0D0,1.0D0/15.0D0,9.0D0/40.0D0/)
! FOR GAUSS-LEGENDER INTEGRALS WITH 7-POINTS
! EXACT WITH 13 DEGREES POLY
END SUBROUTINE QUAD7
SUBROUTINE PHIQ(PHI,PHIX,PHIY,NBF,NQPT,XI,ETA)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION PHI(NBF,NQPT),PHIX(NBF,NQPT),PHIY(NBF,NQPT)
DIMENSION XI(NQPT),ETA(NQPT)
INTENT(OUT) :: PHI,PHIX,PHIY
INTENT(IN) :: NBF,NQPT,XI,ETA
PHI=0.0D0
PHIX=0.0D0
PHIY=0.0D0
DO I=1,NQPT
  X=XI(I)
  Y=ETA(I)
  PHI(1,I) = 1.0D0-X-Y+2.0D0*X*Y
  PHI(2,I) = X/2.0D0 + X**2/2.0D0 + Y/2.0D0 -X*Y-Y**2/2.0D0
  PHI(3,I) = X/2.0D0-X**2/2.0D0+Y/2.0D0-X*Y+Y**2/2.0D0
  PHI(4,I) = -Y+Y**2
  PHI(5,I) = (-X+X**2-Y+2.0D0*X*Y+Y**2)/DSQRT(2.0D0)
  PHI(6,I) = -X+X**2
  PHIX(1,I) = -1.0D0+2.0D0*Y
  PHIX(2,I) = 1.0D0/2.0D0 + X-Y
  PHIX(3,I) = 1.0D0/2.0D0-X-Y
  PHIX(4,I) = 0.0D0
  PHIX(5,I) = DSQRT(2.0D0)*(-0.50D0+X+Y)
  PHIX(6,I) = -1.0D0+2.0D0*X
  PHIY(1,I) = -1.0D0+2.0D0*Y
  PHIY(2,I) = 0.50D0 -X-Y
  PHIY(3,I) = 0.50D0+X+Y
  PHIY(4,I) = -1.0D0+2.0D0*Y
  PHIY(5,I) = DSQRT(2.0D0)*(-0.50D0+X+Y)
  PHIY(6,I) = 0.0D0
END DO
END SUBROUTINE PHIQ

SUBROUTINE MAEA(EA,NBF)
DOUBLE PRECISION, INTENT(OUT) :: EA(NBF,NBF)
INTEGER, INTENT(IN) :: NBF
EA= RESHAPE( &
(4.0D0,-2.0D0,-2.0D0,0.0D0,2.828430D0,0.0D0, &
-2.0D0,2.0D0,0.0D0,-1.0D0,-1.414210D0,1.0D0, &
-2.0D0, 0.0D0, 2.0D0, 1.0D0, -1.414210D0, -1.0D0, &
0.0D0, -1.0D0, 1.0D0, 2.0D0, 1.414210D0, 0.0D0, &
2.828430D0, -1.414210D0, -1.414210D0, 1.414210D0, &
4.0D0, 1.414210D0, 0.0D0, 1.0D0, -1.0D0, 0.0D0, &
1.414210D0, 2.0D0/), &
(/NBF,NBF/) }
END SUBROUTINE MAEA

! STATUS( JACOBS ) = OK
!
SUBROUTINE JACOBS(LX,M,NNZ,NELEM,NIN,NBO, 
&
NBF,A,IA,JA,RE,SOL,IELNODE, &
IN,IBO,EA,B)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
INTENT(IN) :: LX,M,NNZ,NELEM,NIN, &
NBO,NBF,RE,SOL, &
IELNODE,IN,IBO,EA,B
INTENT(OUT) :: A,IA,JA
DIMENSION A(NNZ),IA(NNZ),JA(NNZ), &
IELNODE(NELEM,NBF), &
IN(NIN),IBO(NBO), &
SOL(M),EA(NBF,NBF), &
B(NBF,NBF,NBF)
!
H = 1.0D0 / DBLE(LX)
!----------------ASSIMBLE STIFF ----------------
IA= 0 ; JA = 0 ; A = 0.0D0
INDEX = 0
DO K=1,NELEM
  DO IR =1,NBF
    II = IELNODE(K,IR)
    I = ABS(II)
    DO IS =1,NBF
      JJ=IELNODE(K,IS)
      D=DBLE(SIGN(1,II)*SIGN(1,JJ))
      J = ABS(JJ)
      !AF(I,J) = AF(I,J) + (D/RE)*EA(IR,IS)/H**2
      CONT = (D/RE)*EA(IR,IS)/H**2
      CALL STORE(A,IA,JA,NNZ,I,J,CONT,INDEX)
      SUM = 0.0D0
      DO IK=1,NBF
        K3=IELNODE(K,IK)
        KK= ABS(K3)
D = DBLE(SIGN(1,II)*SIGN(1,JJ)*SIGN(1,K3))
BRACK = B(IR,IS,IK) + B(IR,IK,IS)
SUM=SUM + D*SOL(KK)*BRACK/H**2
IF(K==2 .AND. IR==4 .AND. IS==2 .AND. IK==4)THEN
   END IF
END DO
!AF(I,J) = AF(I,J) + SUM
   CALL STORE(A,IA,JA,NNZ,I,J,SUM,INDEX)
END DO
END DO
END DO
!
END SUBROUTINE JACOBS
!
! STATUS( FORCE ) = OK
!
SUBROUTINE FORCE(RHS,LX,RE,SOL,IELNODE, &
     IN,IBO,XX,YY,PHI,PHIX, &
     PHIY,XI,ETA,W,EA,NBF, &
     NELEM,M,NIN,NBO,NQPT,B)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),SOL(M),IN(NIN), &
     IELNODE(NELEM,NBF),IBO(NBO), &
     XX(M),YY(M),PHI(NBF,NQPT), &
     PHIX(NBF,NQPT),PHIY(NBF,NQPT),&
     XI(NQPT),ETA(NQPT),W(NQPT), &
     EA(NBF,NBF),B(NBF,NBF,NBF)
! LOCA
DIMENSION X(3),Y(3)
INTENT(IN) :: LX,RE,SOL,IELNODE,IN, &
     IBO,XX,YY,PHI,PHIX, &
     PHIY,XI,ETA,W,EA,B, &
     NBF,NELEM,M,NIN,NBO, &
     NQPT
H=1.0D0/DBLE(LX)
RHS = 0.0D0
DO NK=1,NELEM
   DO II=1,3
      X(II) = XX( ABS(IELNODE(NK,II)) )
   END DO
   Y(II) = YY( ABS(IELNODE(NK,II)) )
END DO
! --------------------------------------
   DO IR=1,6
      II=IELNODE(NK,IR)
I = ABS(II)
D = DBLE(SIGN(1,II))
    SUM1 = 0.0D0 ; SUM2 = 0.0D0
DO IZ=1,7
    S = (X(2) - X(1)) * XI(IZ) + X(1)
    T = (Y(3) - Y(1)) * ETA(IZ) + Y(1)
    CALL F1(S, T, RE, F1ST)
    CALL F2(S, T, RE, F2ST)
    SUM1 = SUM1 + W(IZ) * PHIY(IR, IZ) * F1ST
    SUM2 = SUM2 + W(IZ) * PHIX(IR, IZ) * F2ST
END DO
VALUE = -SUM1 * (X(2) - X(1)) + SUM2 * (Y(3) - Y(1))
RHS(I) = RHS(I) + D * VALUE
!
SUM = 0.0D0
DO IS=1,NBF
    JJ = IELNODE(NK, IS)
    J = ABS(JJ)
    D = DBLE(SIGN(1, II) * SIGN(1, JJ))
    SUM = SUM + (D/RE) * EA(IR, IS) * SOL(J)
END DO
RHS(I) = RHS(I) + SUM/H**2
!
END SUBROUTINE FORCE
!
!
SUBROUTINE MAB(B,NBF)
DOUBLE PRECISION, INTENT(OUT) :: B(NBF,NBF,NBF)
INTEGER, INTENT(IN) :: NBF
B = RESHAPE( (/ &
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  4.714045200000000E-001, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  1.666666700000000E-001, &,
-1.666666700000000E-001, &
  1.666666700000000E-001, &,
  0.000000000000000E+000, &,
  1.666666700000000E-001, &,
  0.000000000000000E+000, &,
-1.666666700000000E-001, &,
  1.666666700000000E-001, &,
-1.666666700000000E-001, &
-4.714045200000000E-001, &
-1.666666700000000E-001, &
  0.000000000000000E+000, &,
-3.333333300000000E-001, &,
  3.333333300000000E-001, &,
  0.000000000000000E+000, &,
-4.714045200000000E-001, &,
-3.333333300000000E-001, &,
  0.000000000000000E+000, &,
-2.357022600000000E-001, &,
  2.357022600000000E-001, &,
  2.357022600000000E-001, &,
  0.000000000000000E+000, &,
-2.357022600000000E-001, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  3.333333300000000E-001, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
  0.000000000000000E+000, &,
-3.333333300000000E-001, &,
  3.333333300000000E-001, &,
  0.000000000000000E+000, &,
-1.178511300000000, &,
-3.333333300000000E-001, &,
  1.666666700000000E-001, &
-2.500000000000000E-001, &
8.333333300000000E-002, &
-1.666667000000000E-001, &
5.892556500000000E-001, &
0.000000000000000E+000, &
-1.666667000000000E-001, &
5.833333300000000E-001, &
-4.166667000000000E-001, &
1.666667000000000E-001, &
5.892556500000000E-001, &
3.333333300000000E-001, &
0.000000000000000E+000, &
4.166667000000000E-001, &
-4.166667000000000E-001, &
0.000000000000000E+000, &
5.892556500000000E-001, &
3.333333300000000E-001, &
2.357026000000000E-001, &
-2.357026000000000E-001, &
0.000000000000000E+000, &
-2.357026000000000E-001, &
0.000000000000000E+000, &
3.333333300000000E-001, &
-4.166667000000000E-001, &
8.333333300000000E-002, &
-3.333333300000000E-001, &
5.892556500000000E-001, &
0.000000000000000E+000, &
0.000000000000000E+000, &
3.333333300000000E-001, &
-3.333333300000000E-001, &
0.000000000000000E+000, &
7.071067800000001E-001, &
3.333333300000000E-001, &
-1.666667000000000E-001, &
8.333333300000000E-002, &
8.333333300000000E-002, &
0.000000000000000E+000, &
-5.892556500000000E-001, &
-1.666667000000000E-001, &
1.666667000000000E-001, &
-4.166667000000000E-001, &
2.500000000000000E-001, &
0.000000000000000E+000, &
-1.17851300000000E-001, &
-1.66666700000000E-001, &
0.00000000000000E+000, &
-8.33333300000000E-002, &
8.33333300000000E-002, &
0.00000000000000E+000, &
-1.17851130000000E-001, &
0.00000000000000E+000, &
-2.35702260000000E-001, &
4.71405200000000E-001, &
-2.35702260000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
2.35702260000000E-001, &
-3.33333300000000E-001, &
4.16666700000000E-001, &
-8.33333300000000E-002, &
0.00000000000000E+000, &
-5.89255650000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
0.00000000000000E+000, &
3.33333300000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
1.66666700000000E-001, &
-1.66666700000000E-001, &
0.00000000000000E+000, &
-1.66666700000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
1.66666700000000E-001, &
-1.66666700000000E-001, &
0.00000000000000E+000, &
-1.66666700000000E-001, &
1.66666700000000E-001, &
0.00000000000000E+000, &
-1.66666700000000E-001, &
0.00000000000000E+000, &
-3.33333300000000E-001, &
3.33333300000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
0.00000000000000E+000, &
0.00000000000000E+000, &
-2.35702260000000E-001, &
2.357022600000000E-001, &
0.000000000000000E+000, &
2.357022600000000E-001, &
0.000000000000000E+000, &
0.000000000000000E+000, &
0.000000000000000E+000, &
0.000000000000000E+000, &
0.000000000000000E+000, &
0.000000000000000E+000, &
4.714045200000000E-001, &
-1.178511300000000, &
7.071067800000001E-001, &
0.000000000000000E+000, &
0.000000000000000E+000, &
-4.714045200000000E-001, &
0.000000000000000E+000, &
5.892556500000000E-001, &
-5.892556500000000E-001, &
0.000000000000000E+000, &
8.333333300000000E-001, &
4.714045200000000E-001, &
-4.714045200000000E-001, &
5.892556500000000E-001, &
-1.178511300000000, &
0.000000000000000E+000, &
-8.333333300000000E-001, &
0.000000000000000E+000, &
0.000000000000000E+000, &
-5.892556500000000E-001, &
5.892556500000000E-001, &
0.000000000000000E+000, &
-8.333333300000000E-001, &
-4.714045200000000E-001, &
6.666667000000000E-001, &
-1.666667000000000, &
1.000000000000000, &
0.000000000000000E+000, &
0.000000000000000E+000, &
-6.666667000000000E-001, &
4.714045200000000E-001, &
-5.892556500000000E-001, &
1.178511300000000E-001, &
0.000000000000000E+000, &
END SUBROUTINE MAB

8.33333330000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
-3.33333330000000E-001, &
3.33333330000000E-001, &
0.00000000000000E+000, &
-4.71404520000000E-001, &
-3.33333330000000E-001, &
1.66666670000000E-001, &
0.00000000000000E+000, &
-1.66666670000000E-001, &
0.00000000000000E+000, &
4.71404520000000E-001, &
1.66666670000000E-001, &
-1.66666670000000E-001, &
3.33333330000000E-001, &
-1.66666670000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
1.66666670000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
0.00000000000000E+000, &
0.00000000000000E+000, &
2.35702260000000E-001, &
-4.71404520000000E-001, &
2.35702260000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
-2.35702260000000E-001, &
3.33333330000000E-001, &
-3.33333330000000E-001, &
0.00000000000000E+000, &
0.00000000000000E+000, &
4.71404520000000E-001, &
0.00000000000000E+000 &
/), &
(/NBF , NBF , NBF /), &
ORDER = (/3,2,1/) &

END SUBROUTINE MAB
Appendix B

FORTRAN 77 Program Using Bogner-Fox-Schmit Element
THIS PROGRAM IS FOR SOLVING THE NAVIER-STOKES EQUATIONS USING THE TWO LEVEL FINITE ELEMENT METHOD ON THE STREAM FUNCTION FORMULATION FOR A FULL DESCRIPTION OF THE METHOD SEE F. FAIRAG, TWO-LEVEL FINITE ELEMENT METHOD FOR THE STREAM FUNCTION FORMULATION OF THE NAVIER-STOKES EQUATIONS, TO BE APPEAR IN INTERNATIONAL JOURNAL COMPUTERS AND MATHEMATICS WITH APPLICATIONS.


CONSIDER THE FOLLOWING EQUATION:

\begin{equation}
\begin{split}
R^{-1} \nabla^2 \psi - \psi_{y} \nabla \psi_{x} + \psi_{x} \nabla \psi_{y} &= \text{curl}\, \text{F}, \text{ in } \Omega, \\
\psi &= 0, \text{ on } \partial \Omega, \\
\frac{\partial \psi}{\partial \hat{n}} &= 0, \text{ on } \partial \Omega.
\end{split}
\end{equation}

WHERE $\hat{n}$ REPRESENTS THE OUTWARD UNIT NORMAL TO $\Omega$.

\[ \text{LAP } (F) = \text{Laplacian of } F = F_{xx} + F_{yy} \]
\[ R(F,G) = F \times \text{LAP}(G) - G \times \text{LAP}(F) \]

**BOGNER-FOX-SCHMEDIT RECTANGLE:**

For the BFS rectangular element, we have that the functions are bicubic polynomials within each element. The degrees of freedom are chosen to be the function value, the first derivatives, and mixed second derivatives at the vertices. So the dimension is 16.

**SYMBOLS:**

- \( L_X \) = # of elements along \( X \)-axis
- \( L_{X0} \) = # of elements along \( X \)-axis in the coarse mesh
- \( \text{RE} \) = Reynolds number
- \( \text{NELEM} \) = # of elements = \( L_X \times L_X \)
- \( \text{NPTS} \) = # of points = \( (L_X+1) \times (L_X+1) \)
- \( \text{NBO} \) = # of boundary points on the all boundary except the top which relates to \( y \)-derivative = \( 15 \times L_X - 1 \)
- \( \text{NBOY} \) = # of boundary points on the top which relates to \( y \)-derivative = \( L_X + 1 \)
- \( \text{NBF} \) = # of basis functions = 16
- \( M \) = # of degrees of freedom = \( 4 \times \text{NPTS} \)
- \( \text{NNZ} \) = # of nonzero entries in the Jacobian matrix = \( 36 \times M \)
- \( \text{MAX} \) = maximum number of Newton iterations to solve the nonlinear system resulting from coarse mesh
ARRAYS:

IELNODE(NELEM,NBF) = GLOBAL NODE NUMBERS FOR ALL ELEMENTS

IBO(NBO) = THE SET OF NODE NUMBERS OF NODES ON BOUNDARY EXCEPT THE TOP

IBOY(NNBO) = THE SET OF NODE NUMBERS OF NODES ON THE TOP BOUNDARY WHERE NON-HOMOGENOUS CONDITION IS SATISFIED

XX(M) = X-COORDINATES OF ALL NODES

YY(M) = Y-COORDINATES OF ALL NODES

XI(4) = X-COORDINATES OF INTEGRATION POINTS ON THE SQUARE [-1,1]x[-1,1]

ETA(4) = Y-COORDINATES OF INTEGRATION POINTS ON THE SQUARE [-1,1]x[-1,1]

W(4) = THE CORRESPONDING INTEGRATION WEIGHTS

PHI(16,6,4) = VALUES OF BASIS FUNCTIONS, X-DERIVATIVE, Y-DERIVATIVE, XY-DERIVATIVE, XX-DERIVATIVE, YY-DERIVATIVE AT THE INTEGRATION POINTS.

EXAMPLE: PHI(3,2,3) = PHI3_Y(XI(3),ETA(3))

* THIS PROGRAM CAN SOLVE TWO PROBLEMS:

1- NSE IN [0,1]^2 WITH KNOWN STREAMFUNCTION

   PSI(X,Y) = X^2*(X-1)^2*Y*(Y-1)^2

2- DRIVEN CAVITY PROBLEM IN [0,1]^2

   WITH U=V=0 ON THE BOUNDARY AND ZERO NORMAL DERIVATIVE ON ALL BOUNDARY EXCEPT THAT U=1 ON THE TOP

* TO COMPILE THIS PROGRAM SEE LINE 1389

* WITH THIS CODE WE CAN SOLVE ANY NSE IN A SQUARE OR ANY DOMAIN SEE LINE 1389.
PARAMETER(LX =16,LXO=LX/2,
- NPTS = (LX+1)**2,NPTSO=(LXO+1)**2,
- M = 4*NPTS,M0=4*NPTSO,
- NELEM= LX**2,NELEMO=LXO**2,
- NBF = 16,
- NBO = 15*LX-1,NBOO=15*LXO-1,
- NBOY = LX+1,NBOYO=LXO+1,
- NNZ = 36*M,NNZO=36*M0,
- MAX =20)

IMPLICIT DOUBLE PRECISION (A-H,O-Y)
IMPLICIT INTEGER (I-N)
DIMENSION IELNODE(NELEM,NBF),IBO(NBO),IBOY(NBOY),
- RHS(M),A(NNZ),IA(NNZ),JA(NNZ),SOL(M),X(M),
- ER(0:MAX,2),ITER(0:MAX),SOLO(MO),X0(M),
- XRE(18)
CHARACTER*7 ZO(4)
CHARACTER*6 Z(4)
DATA XRE/ 1.0 , 10.0, 20.0 , 30.0 , 240.0 , 250.0,140.0 
- , 160.0 , 180.0 , 200.0 , 250.0, 300.0, 
- 350.0 , 370.0 , 380.0 , 390.0 , 390.0 /
DATA Z/ 'CN001' , 'CN010' , 'CN050', 'CN100'/
DATA ZO/'CO001' , 'CO010' , 'CO050', 'CO100'/
TOL=0.0000010D0
CALL GUISS (SOLO,LXO,MO,NBF,NBOO,NBOYO,NELEMO,NPTSO)
C OPEN(UNIT=15,FILE='SOLO',STATUS='OLD')
C READ (15,*) SOLO
C CLOSE(UNIT=15)
DO 66 IE=1,1
RE = XRE(IE)
C -------------- STEP1 OR READ ----------------
C OPEN(UNIT=15,FILE='SOLO',STATUS='OLD')
C OPEN(UNIT=25,FILE=Z(IE),STATUS='OLD')
C READ (15,*) SOLO
C CLOSE(UNIT=15)
C CALL GUISS (SOLO,LXO,MO,NBF,NBOO,NBOYO,NELEMO,NPTSO)
CALL CSOLVE(SOLO,ER,ITER,LXO,RE,MO,NPTSO,
- NELEMO,NBOO,NBOYO,MAX,TOL,NNZO)
C ------- GENERATE INITIAL GUISS FOR LINEAR SOLVE -------
C IN THE FINE MESH OR READ -------------------
C YOU CAN USE THE OUTPUT FROM INTER.M --------
C MATLAB FUNCTION ------------------------
C OPEN(UNIT=25,FILE='SOL',STATUS='OLD')
C READ (25,*) SOL
C CLOSE(UNIT=25)
C CALL GUISS (SOL,LX,M,NBF,NBO,NBOY,NELEM,NPTS)
   CALL GUISS (X0 ,LX,M,NBF,NBO,NBOY,NELEM,NPTS)
C ----------------- STEP2 -------------------
   CALL FSOLVE(SOL,RE,SOLO,X0,
   - M,MO,LX,LXO,TOL,NPTS,NPTSO,NBF,
   - NBO,NBOY,NELEM,NELEMO,NBOO,
   - NBOYO,NNZ)
66 CONTINUE
END
C ----------------------------------------
C THIS SUBROTINE WILL TAKE SOLO VECTOR AS THE
C THE SOLUTION IN THE COARSE MESH AND COMPUTE
C THE MATRIX A, IA, JA AND RHS AND SOLVE THE
C LINEAR SYSTEM TO GENERATE THE VECTOR SOL AS
C THE SOLUTION OF THE PROBLEM IN THE FINE MESH
C ----------------------------------------
SUBROUTINE FSOLVE(SOL,RE,SOLO,X0,
   - M,MO,LX,LXO,TOL,NPTS,NPTSO,NBF,
   - NBO,NBOY,NELEM,NELEMO,NBOO,
   - NBOYO,NNZ)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION SOL(M),SOLO(MO),IBO(NBO),IBOO(NBOO),
   - IBOY(NBOY),IBOYO(NBOYO),X0(M),Y0(M),
   - IELNODE(NELEM,NBF),IELNODEO(NELEMO,NBF),
   - EA(NBF,NBF),B(NBF,NBF,NBF),RHS(M),
   - A(NNZ),IA(NNZ),JA(NNZ),XX(M),YY(M),
   - PHI(NBF,6,4),XI(4),ETA(4),W(4)
H=1.0D0/DBLE(FLOAT(LX))
C
C HERE WE COMPUTE THE GLOBAL LABELING ARRAY
C IELNODE FOR THE FINE MESH AND IELNODEO ARRAY
C FOR THE COARSE MESH AND THE X-Y-COORDINATES
C OF THE NODES
C
   CALL MAELNODE(IELNODE,IBO,IBOY,LX,
   - NBO,NBOY,NBF,M,NELEM,NPTS)
   CALL MAELNODE(IELNODEO,IBOO,IBOYO,LXO,
   - NBOO,NBOYO,NBF,MO,NELEMO,NPTSO)
   CALL COORD(XX,YY,LX,M)
C
C HERE WE READ THE INTEGRATION POINTS XI(4)
C AND ETA(4) AND W(4). ALSO WE READ THE ELEMENT
STIFFNESS MATRIX A, IA, JA RESULTING FROM THE BIHARMONIC TERM AND THE ARRAY B(16,16,16) WHICH COMES FROM THE TRILINEAR TERM IN THE STREAM FUNCTION FORM

CALL SQUAD(XI, ETA, W)
NBF2 = NBF**2
CALL MAEA(NBF, NBF2, EA)
NBF3 = NBF**3
CALL MAB(NBF, NBF3, B)
NBF4 = 6*4*NBF
CALL MAQ(NBF, NBF4, PHI)
CALL RZEROS(M, RHS)

HERE WE COMPUTE THE RHS VECTOR OF THE LINEAR SYSTEM

CALL RHS2L(RHS, LX, M, RE, NBF, IELNODE,
- XX, YY, PHI, XI, ETA, W, NELEM, H)

HERE WE COMPUTE THE RESULTING MATRIX

CALL FSTIFF(A, IA, JA, LX, RE, IELNODE,
- XX, YY, LXO, IELNODEO, SOLO, EA, B,
- NNZ, NBF, NELEM, NELEMO, M, MO, H, RHS)

HERE WE TAKE CARE THE BOUNDARY CONDITION SO CHANGE THE MATRIX A AND RHS

CALL ATX(Y0, A, X0, IA, JA, NNZ, M)
CALL AXPBY(RHS, 1.0D0, RHS, -1.0D0, Y0, M)
CALL FBOUNDRYA(A, IA, JA, IBO, IBOY,
- NNZ, NBO, NBOY)
CALL FBORHS(RHS, IBO, IBOY, M, NBO, NBOY, X0)

HERE WE SEND THE MATRIX AND RHS TO THE LINEAR SOLVER BICGSTAB

CALL BICGSTAB(SOL, K, RHS, TOL, A, IA, JA, NNZ, M, 1)

HERE WE COMPUTE THE RESIDUAL = F(SOL)

CALL ATX(X0, A, SOL, IA, JA, NNZ, M)
CALL AXPBY(X0, 1.0D0, RHS, -1.0D0, X0, M)
CALL NORM(M, X0, RESDA)
CALL FORCE(X0,LX,RE,SOL,
   - IELNODE,IBO,XX,YY,EA,B,PHI,W,XI,ETA,
   - M,NBF,NELEM,NBO)
CALL FBORHS(X0,IBO,IBOY,M,NBO,NBOY,X0)
CALL NORM(M,X0,RESDF)

-----------
CALL BACKUP(SOL,M,LX,LXO,RE,RESDA,RESDF,K)
END

----------------------------------------
SUBROUTINE RHS2L(RHS,LX,M,RE,NBF,IELNODE,
   - XX,YY,PHI,XI,ETA,W,NELEM,H)

C
C THIS SUBROUTINE WILL COMPUTE THE RHS VECTOR
C OF THE LINEAR SYSTEM ON THE FINE MESH
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),IELNODE(NELEM,NBF),
   - XX(M),YY(M),PHI(NBF,6,4),
   - XI(4),ETA(4),W(4)
DIMENSION X(4),Y(4)
CALL RZEROS(M,RHS)
DO 10 NK=1,NELEM
   DO 20 II=1,4
      X(II) = XX( IELNODE(NK,II) )
      Y(II) = YY( IELNODE(NK,II) )
   20 CONTINUE
   DO 30 IR=1,NBF
      I=IELNODE(NK,IR)
      SUM1=0.0D0
      SUM2=0.0D0
      DO 40 IZ=1,4
         S=H*(XI(IZ)+1.0D0)/2.0D0+X(1)
         T=H*(ETA(IZ)+1.0D0)/2.0D0+Y(1)
         CALL F1(VF1,S,T,RE)
         CALL F2(VF2,S,T,RE)
         SUM1=SUM1 + W(IZ)*PHI(IR,3,IZ)*VF1
         SUM2=SUM2 + W(IZ)*PHI(IR,2,IZ)*VF2
      40 CONTINUE
      VALUE =SUM1 - SUM2
      RHS(I) = RHS(I) + VALUE*H/2.0D0
   30 CONTINUE
10 CONTINUE
END

C

----------------------------------------
SUBROUTINE FSTIFF(A,IA,JA,LX,RE,IELNODE,
-       XX,YY,LXO,IELNODEO,SOLO,EA,B,
-       NNZ,NBF,NELEM,NELEMO,M,MO,H,RHS)

C
C THIS SUBROTINE WILL TAKE THE INPUT VECTOR SOLO
C TO COMPUTE THE MATRIX A,IA,J A IN A SPARSE FORM
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION A(NNZ),IA(NNZ),JA(NNZ),XX(M),YY(M),
-       IELNODE(NELEM,NBF),IELNODEO(NELEMO,NBF),
-       SOLO(MO),EA(NBF,NBF),B(NBF,NBF,NBF),
-       X(4),Y(4),RHS(M)
RATIO=DBLE(FLOAT(LX/LXO))
CALL RZEROS(M,A)
CALL IZEROS(M,IA)
CALL IZEROS(M,JA)
INDEX=0
DO 10 K=1,NELEM
   DO 20 IX=1,4
      IP=IELNODE(K,IX)
      X(IX) = XX(IP)
      Y(IX) = YY(IP)
20 CONTINUE
   XM=( X(1)+X(2) )/2.0D0
   YM=( Y(1)+Y(3) )/2.0D0
   DO 30 IR =1,NBF
      I = IELNODE(K,IR)
      DO 40 IS =1,NBF
         J=IELNODE(K,IS)
         CONT1= (4.0D0/RE)*EA(IR,IS)/H**2
         CALL STORE(A,IA,JA,NNZ,I,J,CONT1,INDEX)
         SUM = 0.0D0
         DO 50 IK=1,NBF
            CALL FINDK(XM,YM,LXO,KO)
            KK=IELNODEO(KO,IK)
            B1 = 4.0D0*B(IR,IS,IK)/(RATIO*H**2)
            SUM=SUM + SOLO(KK)*(B1)
50 CONTINUE
         CALL STORE(A,IA,JA,NNZ,I,J,SUM,INDEX)
40 CONTINUE
30 CONTINUE
10 CONTINUE
END
C----------------------------------------
SUBROUTINE FINDK(XM,YM,LX,K)
C
GIVEN A POINT (XM,YM)
K WILL THE ELEMENT NUMBER WHERE THAT POINT
LIVE IN (LX+1) X (LX+1) GRID POINTS
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
H = 1.0D0/DBLE(FLOAT(LX))
X = 0.0D0+H
Y = 0.0D0+H
KX =1
KY=1
DO 10 I= 2,LX+1
   IF( X .GT. XM ) THEN
      KX = I-1
      EXIT
   ENDIF
   X = X + H
10 CONTINUE
DO 20 J=2,LX+1
   IF( Y .GT. YM) THEN
      KY = J -1
      EXIT
   ENDIF
   Y = Y + H
20 CONTINUE
K = (KY -1)*LX + KX
END
C ----------------------------------------
SUBROUTINE FBORHS(RHS,IBO,IBOY,M,NBO,NBOY,XO)
C
THIS SUBROTINE WILL ZEROS THE ENTRIES IN THE
VECTOR RHS WHICH RELATES THE BOUNDARY NODES OR
CHANGE THEM SO THEY WILL SATISFY THE BOUNDARY
CONDITION
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),IBO(NBO),IBOY(NBOY),XO(M)
DO 10 II=1,NBO
   I=IBO(II)
   RHS(I)=XO(I)
10 CONTINUE
DO 20 II=1,NBOY
I=IBOY(II)
RHS(I)=X0(I)
20 CONTINUE
END

C ----------------------------------------
SUBROUTINE FBOUNDRYA(A,IA,JA,IBO,IBOY,
-   NNZ,NBO,NBOY)
C
C THIS SUBROUTINE WILL CHANGE THE MATRIX A IN
C THE FOLLOWING MANNER: ( WHERE THE RESULTING MATRIX
C FROM FINE LEVEL SOLVE )
C
C A(:,I)=0 AND A(I,:) AND A(I,I)=1
C FOR ALL I IN THE VECTOR IBO AND IBOY
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION A(NNZ),IA(NNZ),JA(NNZ),
-   IBO(NBO),IBOY(NBOY)
DO 10 II=1,NBO
   I=IBO(II)
   DO 20 K=1,NNZ
      IF(IA(K) .EQ.I.AND.JA(K).EQ.I) THEN
         A(K)=1.0D0
      ELSEIF(IA(K).EQ.I.OR.JA(K).EQ.I) THEN
         A(K)=0.0D0
      ELSE
         ENDIF
   20 CONTINUE
10 CONTINUE
DO 30 II=1,NBOY
   I=IBOY(II)
   DO 40 K=1,NNZ
      IF(IA(K) .EQ.I.AND.JA(K).EQ.I) THEN
         A(K)=1.0D0
      ELSEIF(IA(K).EQ.I.OR.JA(K).EQ.I) THEN
         A(K)=0.0D0
      ELSE
         ENDIF
   40 CONTINUE
SUBROUTINE CSOLVE(SOL,ER,ITER,LX,RE,M,NPTS,
- NELEM,NBF,NBO,NBOY,M,N,MAX,TOL,NNZ)

C THIS SUBROUTINE WILL SOLVE THE NAVIER-STOKES
C EQUATION USING ONE LEVEL FINITE ELEMENT METHOD
C GIVING THE FOLLOWING :
C LX = # OF SQUARES ALONG X-AXIS
C RE = REYNOLDS NUMBER
C SOL = SOLUTION VECTOR (OUTPUT)
C = INITIAL GUSS (INPUT)
C ER(I,1) = I-TH RESIDUAL ERROR
C ER(I,2) = NORM( SOL_I+1 - SOL_I )
C ITER(I) = BICGSTAB ITERATION # NEEDED AT I-TH NEWTON STEP
C THE OUTPUT SOLUTION IS STORED IN SOL AS BELOW :
C SOL(1:NPTS) = PSI VALUES AT (LX+1)^2 POINTS
C SOL(NPTS+1:2*NPTS) = PSI_X VALUES AT (LX+1)^2 POINTS
C SOL(2*NPTS+1:3*NPTS) = PSI_Y VALUES AT (LX+1)^2 POINTS
C SOL(3*NPTS+1:4*NPTS) = PSI_XY VALUES AT (LX+1)^2 POINTS
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION SOL(M),UP(M)
DIMENSION IELNODE(NELEM,NBF),IBO(NBO),IBOY(NBOY),
- XX(M),YY(M),PHI(NBF,6,4),B(NBF,NBF,NBF),
- XI(4),ETA(4),W(4),EA(NBF,NBF),ER(0:MAX,2),
- ITER(0:MAX),TSOL(M),TRHS(M),RHS(M),
- A(NNZ),IA(NNZ),JA(NNZ)
H=1.0D0/DBLE(FLOAT(LX))
CALL MAELNODE(IELNODE,IBO,IBOY,LX,
- NBO,NBOY,NBF,NELEM,M)
CALL COORD(XX,YY,LX,M)
NBF4=6*4*NBF
CALL MAQ(NBF,NBF4,PHI)
CALL MAB(NBF,NBF**3,B)
CALL SQUAD(XI,ETA,W)
CALL MAEA(NBF,NBF**2,EA)
CALL RZEROS(M,UP)
CALL FORCE(RHS,LX,RE,SOL,
- IELNODE,IBO,XX,YY,EA,B,PHI,W,XI,ETA,
- M,NBF,NELEM,NBO)
CALL BOUNDRYRHS(RHS,IBO,IBOY,
- LX,NBF,NELEM,NBO,NBOY,NPTS,M)
CALL PNORM(M,RHS,XNORM,LX)

OLD = XNORM/DBLE(FLOAT(LX+1))
ER(0,1) = OLD
DO 10 I=1,MAX
   CALL JACOBS(A,IA,JA,LX,RE,SOL,IELNODE,IBO,EA,B,
       - NNZ,M,NBF,NELEM,NBO,H)
   CALL BOUNDRYA(A,IA,JA,IBO,I Boy,
       - NBO,NBOY,M,NNZ)
   NWTN=I
   TOLB=0.00010DO
   CALL BICGSTAB(UP,K,RHS,TOL,A,IA,JA,NNZ,M,NWTN)
   I TER(I)=K;

   CALL AXPBY(TSOL,1.0DO,SOL,-1.0DO,UP,M)
   CALL FORCE(TRHS,LX,RE,T SOL,
       - IELNODE,IBO,XX,YY,EA,B,PHI,W,XI,ETA,
       - M,NBF,NELEM,NBO)
   CALL BOUNDRYRHS(TRHS,IBO,I Boy,
       - LX,NBF,NELEM,NBO,NBOY,NPTS,M)
   CALL PNORM(M,TRHS,XNORM,LX)
   XNEW = XNORM/DBLE(FLOAT(LX+1))
   CALL PNORM(M,UP,XNORM,LX)
   ER2 = XNORM/DBLE(FLOAT(LX+1))

   IF( XNEW .GT. OLD ) THEN
      CALL AXPBY(SOL,1.0DO,SOL,-0.50DO,UP,M)
      CALL FORCE(RHS,LX,RE,SOL,
          - IELNODE,IBO,XX,YY,EA,B,PHI,W, XI,ETA,
          - M,NBF,NELEM,NBO)
      CALL BOUNDRYRHS(RHS,IBO,I Boy,
          - LX,NBF,NELEM,NBO,NBOY,NPTS,M)
      CALL PNORM(M,RHS,XNORM,LX)
      XNEW = XNORM/DBLE(FLOAT(LX+1))
      ER2= 0.50DO*ER2
      PRINT*, ' I USE HALF STEP '
   ELSE
      CALL EQUAL(SOL,TSOL,M)
      CALL EQUAL(RHS,TRHS,M)
   ENDIF

   ER(I,1) = XNEW
ER(I,2) = ER2
OLD = XNEW
C IP=1 FOR TEST AND IP=2 FOR CAVEY
CALL BACKUPC(I,SOL,ER,ITER,MAX,M,LX,RE)
IF(ER(I,1) .LT. TOL .AND. ER(I,2) .LT. TOL) EXIT
10 CONTINUE
END
C -----------------------------------------
SUBROUTINE JACOBS(A,IA,JA,LX,RE,SOL,IELNODE,IBO,EA,B,
- NNZ,M,NBF,NELEM,NBO,H)
C
C THIS SUBROTINE WILL COMPUTE THE JACOBIAN MATRIX OF THE
C PROBLEM AND STORE IT IN THREE VECTORS A, IA, JA.
C A = CONTAINS THE VALUES OF NONZERO ENTREE IN JACOBIAN
C IA = CONTAINS THE ROW LOCATION OF NONZERO ENTRIES
C JA = CONTAINS THE COLUMN LOCATION OF NONZERO ENTRIES
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION A(NNZ),IA(NNZ),JA(NNZ),SOL(M),
- IELNODE(NELEM,NBF),IBO(NBO),EA(NBF,NBF),
- B(NBF,NBF,NBF)
CALL RZEROS(NNZ,A)
CALL IZEROS(NNZ,IA)
CALL IZEROS(NNZ,JA)
INDEX=0
DO 10 K=1,NELEM
  DO 20 IR =1,NBF
    I = IELNODE(K,IR)
    DO 30 IS =1,NBF
      J=IELNODE(K,IS)
      CONT1 = (4.0D0/RE)*EA(IR,IS)/H**2
      CALL STORE(A,IA,JA,NNZ,I,J,CONT1,INDEX)
      SUM = 0.0D0
      DO 40 IK=1,NBF
        KK=IELNODE(K,IK)
        BRACK = B(IR,IS,IK) + B(IR,IK,IS)
        SUM=SUM + 4.0D0*SOL(KK)*BRACK/H**2
40    CONTINUE
      CALL STORE(A,IA,JA,NNZ,I,J,SUM,INDEX)
  30 CONTINUE
20 CONTINUE
10 CONTINUE
END
SUBROUTINE FORCE(RHS,LX,RE,SOL,
   - IELNODE,IBO,XX,YY,EA,B,PHI,W,XI,ETA,
   - M,NBF,NELEM,NBO)

C
C THIS SUBROUTINE WILL TAKE THE INPUT VECTOR SOL
C TO COMPUTE F(SOL) AND STORE IT IN RHS VECTOR
C WHICH WILL BE THE RIGHT HAND SIDE OF THE RESULTING
C LINEAR SYSTEM FROM THE I-TH NEWTON ITERATION AND
C IN THE SAME TIME IT IS THE RESIDUAL.
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),SOL(M),IELNODE(NELEM,NBF),
   - IBO(NBO),XX(M),YY(M),EA(NBF,NBF),
   - B(NBF,NBF,NBF),PHI(NBF,6,4),
   - W(4),XI(4),ETA(4),X(4),Y(4)

H=1.0D0/DBLE(FLOAT(LX))
CALL RZEROS(M,RHS)
DO 10 NK=1,NELEM
   DO 20 II=1,4
      X(II) = XX( IELNODE(NK,II) )
      Y(II) = YY( IELNODE(NK,II) )
   20 CONTINUE
   DO 30 IR=1,NBF
      I=IELNODE(NK,IR)
      SUM1=0.0D0
      SUM2=0.0D0
      DO 40 IZ=1,4
         S=H*(XI(IZ)+1.0D0)/2.0D0+X(1)
         T=H*(ETA(IZ)+1.0D0)/2.0D0+Y(1)
         CALL F1(VF1,S,T,RE)
         CALL F2(VF2,S,T,RE)
         SUM1=SUM1 + W(IZ)*PHI(IR,3,IZ)*VF1
         SUM2=SUM2 + W(IZ)*PHI(IR,2,IZ)*VF2
      40 CONTINUE
   VALUE =-SUM1 + SUM2
   RHS(I) = RHS(I) + H*VALUE/2.0D0
10 CONTINUE

C
C ------------------------
DO 50 IS=1,NBF
   J=IELNODE(NK,IS)
   SUM = SUM + (1.0D0/RE)*EA(IR,IS)*SOL(J)
50 CONTINUE
RHS(I) = RHS(I) + 4.0D0*SUM/H**2

C -----------------------------
SUM=0.0D0
DO 60 IS=1,NBF
   J=IELNODE(NK,IS)
   DO 70 IK=1,NBF
      K=IELNODE(NK,IK)
      SUM=SUM+B(IR,IK,IS)*SOL(K)*SOL(J)
70    CONTINUE
60    CONTINUE
RHS(I) = RHS(I) + 4.0D0*SUM/H**2

C -----------------------------
30 CONTINUE
10 CONTINUE
END

C ------------------------------------
SUBROUTINE NORM(N,X,XNORM)
C
C THIS SUBROUTINE WILL COMPUTE THE EUCLIDEAN NORM
C OF THE VECTOR X AND STORE IT IN XNORM
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(N)
SUM = 0.0D0
DO 10 I=1,N
   SUM = SUM + X(I)**2
10 CONTINUE
XNORM = DSQRT( SUM )
END

C ------------------------------------
SUBROUTINE PNORM(N,X,XNORM,LX)
C
C THIS SUBROUTINE WILL COMPUTE THE EUCLIDEAN
C NORM OF THE VECTOR X(1:4:N)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(N)
SUM = 0.0D0
NPTS=(LX+1)**2
M=4*NPTS
DO 10 K=1,M,4
   SUM = SUM + X(K)**2
10 CONTINUE
XNORM = DSQRT( SUM )
END

SUBROUTINE RZEROS(N,X)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(N)
DO 10 I=1,N
   X(I) = 0.0D0
10 CONTINUE
END

SUBROUTINE IZEROS(N,IX)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION IX(N)
DO 10 I=1,N
   IX(I) = 0
10 CONTINUE
END

SUBROUTINE BOUNDRYA(A,IA,JA,IBO,IBOY,-
   NBO,NBOY,M,NNZ)

C
C THIS SUBROUTINE WILL CHANGE THE MATRIX A IN
C THE FOLLOWING MANNER: ( WHERE THE RESULTING MATRIX
C FROM COARSE LEVEL SOLVE )
C
C A(:,I)=0 AND A(I,:) AND A(I,I)=1
C FOR ALL I IN THE VECTOR IBO AND IBOY
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION A(NNZ),IA(NNZ),JA(NNZ),IBO(NBO),-
   IBOY(NBOY)
DO 10 II=1,NBO
   I=IBO(II)
   DO 20 K=1,NNZ
      IF(IA(K) .EQ.I.AND.JA(K) .EQ.I) THEN
         A(K)=1.0D0
      ELSEIF(IA(K) .EQ.I .OR. JA(K) .EQ.I) THEN
         A(K)=0.0D0
      ELSE
         ENDIF
20 CONTINUE
10 CONTINUE
DO 30 II=1,NBO
   I=IBO(II)
   DO 40 K=1,NNZ
      IF(IA(K).EQ.I.AND.JA(K).EQ.I) THEN
         A(K)=1.0D0
      ELSEIF(IA(K).EQ.I.OR.JA(K).EQ.I) THEN
         A(K)=0.0D0
      ELSE
         ENDIF
   40 CONTINUE
30 CONTINUE
END

SUBROUTINE BOUNDRYRHS(RHS,IBO,IBOY,
   - LX,NBF,NELEM,NBO,NBOY,NPTS,M)
C
C THIS SUBROTINE WILL ZEROS THE ENTRIES IN THE
C VECTOR RHS WHICH RELATES THE BOUNDARY NODES
C SO NO UPDATE IN THESE ENTRIES FOR EACH NEWTON
C ITERATION
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RHS(M),IBO(NBO),IBOY(NBOY)
DO 10 II=1,NBO
   I=IBO(II)
   RHS(I)=0.0D0
10 CONTINUE
DO 20 II=1,NBOY
   I=IBOY(II)
   RHS(I)=0.0D0
20 CONTINUE
END

SUBROUTINE MAELNODE(IELNODE,IBO,IBOY,LX,
   - NBO,NBOY,NBF,NELEM,NPTS)
C
C THIS SUBROTINE COMPUTES THE ELEMENT NODE MATRIX OF
C THE BOGNER-FOX ELEMENTS AND STORE THEM IN IELNODE
C THE BOUNDARY POINTS AND STORE THEM IN IBO AND IBOY.
C POINTS. NOTE THAT THE DOMAIN IS [-1,1]X[-1,1].
C LX = # OF SQUARES ALONG X-AXIS
C NELEM = # OF ELEMENTS = LX^2
C NPTS = # OF POINTS = (LX+1)^2
C H = 2.0 / LX
C NBF = # OF BASIS FUNCTIONS IN EACH ELEMENT = 16
C NBO = # OF BOUNDARY POINTS =
C NNBO = # OF NORMAL BOUNDARY
C ELNODE = EACH ROW CONTAINS THE GLOBAL NUMBERING OF LOCAL BASIS FNC.
C IBO = CONTAINS ALL GLOBAL BOUNDARY NUMBERS EXCEPT THE TOP
C WITH Y-DERIVATIVE NODE
C IBOY = CONTAINS GLOBAL BOUNDARY NUMBERS ON THE TOP WITH
C Y-DERIVATIVE NODE
C
C NOTE : THE GLOBAL NUMBERING IS GOING FROM LOWER LEFT CORNER
C TO THE TOP RIGHT CORNER STARTING BY LISTING THE FUNCTION
C NODE, X-DERIVATIVE NODE, Y-DERIVATIVE NODE AND
C XY-DERIVATIVE NODE FOR A POINT THEN LIST ALL NODES
C FOR THE NEXT POINT AND SO ON.
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C IMPLICIT INTEGER (I-N)
C DIMENSION IELNODE(NELEM,NBF),IBO(NBO),IBOY(NBOY)
C
C DO 10 I=1,LX
C DO 20 J=1,LX
C IE = (I-1) * LX + J
C IK = (I-1)*((LX+1)+J
C K = 4*(IK-1) + 1
C IELNODE(IE,1) = K
C IELNODE(IE,2) = K+4
C IELNODE(IE,3) = K+4*(LX+1)
C IELNODE(IE,4) = K+4*(LX+2)
C IELNODE(IE,5) = IELNODE(IE,1)+1
C IELNODE(IE,6) = IELNODE(IE,2)+1
C IELNODE(IE,7) = IELNODE(IE,3)+1
C IELNODE(IE,8) = IELNODE(IE,4)+1
C IELNODE(IE,9) = IELNODE(IE,1)+2
C IELNODE(IE,10) = IELNODE(IE,2)+2
C IELNODE(IE,11) = IELNODE(IE,3)+2
C IELNODE(IE,12) = IELNODE(IE,4)+2
C IELNODE(IE,13) = IELNODE(IE,1)+3
C IELNODE(IE,14) = IELNODE(IE,2)+3
C IELNODE(IE,15) = IELNODE(IE,3)+3
C IELNODE(IE,16) = IELNODE(IE,4)+3
C
C CONTINUE
CONTINUE

COMPUTE BOUNDARY NODE

NO = 4*(LX+1)

DO 30 I=1,NO
IBO(I) = I
30 CONTINUE

DO 40 J=1,LX+1
K=4*LX*(LX+1)+4*(J-1)+1
IBOY(J) = K+2
IBO(NO+J)=K
IBO(NO+LX+1+J)=K+1
IBO(NO+2*(LX+1)+J)=K+3
40 CONTINUE

N0=7*(LX+1)
N1=11*(LX+1)-8

DO 50 IKK=2,LX
IK1=IKK*(LX+1)
IK2=(IKK-1)*(LX+1)+1
K1=(IK1-1)*4+1
K2=(IK2-1)*4+1
IBO(NO+IKK-1)=K1
IBO(NO+LX-2+IKK)=K1+1
IBO(NO+2*(LX-1)+IKK-1)=K1+2
IBO(NO+3*(LX-1)+IKK-1)=K1+3
IBO(N1+IKK-1)=K2
IBO(N1+LX-2+IKK)=K2+1
IBO(N1+2*(LX-1)+IKK-1)=K2+2
IBO(N1+3*(LX-1)+IKK-1)=K2+3
50 CONTINUE

END

SUBROUTINE COORD(X,Y,LX,M)

THIS SUBROUTINE WILL COMPUTE THE X-COORDINATE
AND Y-COORDINATE OF ALL NODES AND STORE THEM
IN X AND Y ARRAY.

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(M),Y(M)
NPTS = M/4
H = 1.0D0/DBLE(FLOAT(LX))
Y0 = 0.0D0
DO 10 I=1,LX+1
  X0 = 0.0D0
DO 20 J=1,LX+1
  IK = (I-1)*(LX+1) + J
  K = (IK-1)*4+1
  X(K) = X0
  Y(K) = Y0
  X(K+1) = X0
  X(K+2) = X0
  X(K+3) = X0
  Y(K+1) = Y0
  Y(K+2) = Y0
  Y(K+3) = Y0
  X0 = X0 + H
20  CONTINUE
Y0 = Y0 + H
10  CONTINUE
END

SUBROUTINE SQUAD(XI,ETA,W)

C
C THIS SUBROUTINE WILL RETURN THE INTEGRATION
C POINTS ON THE SQUARE [-1,1]X[-1,1] THE
C X-COORDINATES OF THE POINTS ARE STORED IN
C ARRAY XI AND THE Y-COORDINATES STORED IN
C ARRAY ETA AND THE WIGHT ARE STORED IN W
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION XI(4),ETA(4),W(4)
P = 1.0D0/DSQRT(3.0D0)
PN = -P
XI(1)=PN
XI(2)=P
XI(3)=PN
XI(4)=P
ETA(1)=PN
ETA(2)=PN
ETA(3)=P
ETA(4)=P
W(1)=1.0D0
W(2)=1.0D0
W(3)=1.0D0
W(4)=1.0D0
SUBROUTINE MAQ(NBF,NBF4,Q)
C
C THIS SUBROUTINE WILL READ THE FILE Q1
C WHICH CONTAINS THE VALUES OF THE 16 BASIS
C FUNCTIONS, THE X-DERIVATIVE, Y-DERIVATIVE
C XY-DERIVATIVE, XX-DERIVATIVE AND YY-DERIVATIVE
C AND STORE THEM IN THE ARRAY Q(16,6,4). THE
C FIRST ARGUMENT OF Q REPRESENTS WHICH BASIS FUNCTIONS
C THE SECOND ARGUMENT MEANS WHICH DERIVATIVE AND
C THE THIRD ARGUMENT REPRESENTS WHICH INTEGRATION
C POINTS
C
IMPLICIT DOUBLE PRECISION (A-H,O-Y)
IMPLICIT INTEGER (I-N)
DIMENSION Q(NBF,6,4),Q1(NBF4)
OPEN(UNIT=10,FILE='Q1.DAT',STATUS='OLD')
DO 88 I=1,NBF4
  READ (10,*) Q1(I)
88 CONTINUE
DO 20 I=1,NBF
  DO 30 J=1,6
    DO 40 K=1,4
      N = (I-1)*24+(J-1)*4+K
      Q(I,J,K)=Q1(N)
  40 CONTINUE
30 CONTINUE
20 CONTINUE
CLOSE(UNIT=10)
END

SUBROUTINE MAEA(NBF,NBF2,EA)
C
C THIS SUBROUTINE WILL READ THE FILE
C EA WHICH CONTAINS ALL VALUES OF POSSIBLE
C INTEGRAL COMES FROM THE BIHARMONIC TERM
C BETWEEN THE 16 BASIS FUNCTIONS
C EA(I,J) = THE DOUBLE INTEGRAL OVER THE
C THE SQUARE [-1,1]X[-1,1] OF THE PRODUCT OF
C LAPLACE OF THE I-TH BASIS FUNCTION AND
C THE LAPLACE OF THE J-THE BASIS FUNCTION
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION EA(NBF,NBF),IEA(NBF2)
OPEN(UNIT=10,FILE='EA.DAT',STATUS='OLD')
DO 99 I=1,NBF2
   READ (10,*) IEA(I)
99 CONTINUE
DO 20 I=1,NBF
   DO 30 J=1,NBF
      IP = (I-1)*16+J
      EA(I,J)=DBLE(FLOAT(IEA(IP)))/3150.0D0
30 CONTINUE
20 CONTINUE
CLOSE(UNIT=10)
END

SUBROUTINE MAB(NBF,NBF3,B)

C THIS SUBROUTINE WILL READ THE FILE
C BB WHICH CONTAINS ALL VALUES OF POSSIBLE
C INTEGRAL COMES FROM THE TRILINEAR TERM
C BETWEEN THE 16 BASIS FUNCTIONS AND STORE
C THEM IN THE ARRAY B(16,16,16).
C B(I,J,K) = B( PHI_I ,PHI_J ,PHI_K)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION B(NBF,NBF,NBF),IBB(NBF3)
OPEN(UNIT=10,FILE='BB.DAT',STATUS='OLD')
DO 99 I=1,NBF3
   READ (10,*) IBB(I)
99 CONTINUE
DO 20 I=1,NBF
   DO 30 J=1,NBF
      DO 40 K=1,NBF
         IP = (I-1)*256 + (J-1)*16 + K
         B(I,J,K)=DBLE(FLOAT(IBB(IP)))/88200.0D0
40 CONTINUE
30 CONTINUE
20 CONTINUE
CLOSE(UNIT=10)
END

C -----------------------------------------
SUBROUTINE BICGSTAB(X,K,B,TOL,C,IC,JC,NNZ,M,NWTN)
C
THIS SUBROUTINE WILL SOLVE ANY LINEAR SYSTEM OF EQUATION WITH THE RHS STORED IN B AND THE MATRIX ARE STORED IN C, IC, JC WHERE C CONTAINS ALL NONZERO ENTRIES AND IC CONTAINS THE ROW LOCATION OF THE NONZERO ENTRIES AND JC CONTAINS THE COLUMN LOCATION OF THE NONZERO ENTRIES. THIS SUBROUTINE SOLVES THE LINEAR SYSTEM C X = B USING A BICGSTAB ALGORITHM.

THIS ALGORITHM IS DESCRIBED IN THE FOLLOWING
REFERENCE:
T. CHAN J. DEMMEL J. DONATOR J. DONCARRA V. EIJKHOUT
R. POZO C. ROMINE R. BARRETT M. BERRY AND H. VAN DEV VORST.
TEMPLATES: FOR THE SOLUTION OF LINEAR SYSTEMS:
BUILDING BLOCKS FOR ITERATIVE METHODS.
THE APPROXIMATION SOLUTION IS STORED IN THE VECTOR X(M) AND USE THIS X AS THE INITIAL GUESS.

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(M), B(M), C(NNZ), IC(NNZ), JC(NNZ)

LOCAL
DIMENSION R(M), RH(M), P(M), V(M), S(M), T(M), UP(M)
K=0
CALL RZEROS(M, R)
CALL RZEROS(M, RH)
CALL RZEROS(M, P)
CALL RZEROS(M, V)
CALL RZEROS(M, S)
CALL RZEROS(M, T)
CALL RZEROS(M, UP)
CALL ATX(R, C, X, IC, JC, NNZ, M)
CALL AXPBY(R, 1.0D0, B, -1.0D0, R, M)
CALL EQUAL(RH, R, M)
DO 10 I=1, 10*M*M
   CALL DOT(ROW, RH, R, M)
   IF(ROW .EQ. 0.0D0 ) EXIT
   IF(I .EQ. 1) THEN
      CALL EQUAL(P, R, M)
   ELSE
      BETA = (ROW/ROWOLD)*(ALPHA/W)
      CALL AXPBY(P, BETA, P, -BETA*W, V, M)
      CALL AXPBY(P, 1.0D0, R, 1.0D0, P, M)
   ENDIF
   CALL ATX(V, C, P, IC, JC, NNZ, M)
   CALL DOT(VAL, RH, V, M)
   ALPHA = ROW/VAL
CALL AXPBY(S, 1.0D0, R, -ALPHA, V, M)
CALL NORM(M, S, XNORM)
IF( XNORM .LT. TOL ) THEN
  K = I
  CALL AXPBY(X, 1.0D0, X, ALPHA, P, M)
  PRINT*, 'I STOP BECAUSE NORM(S)<TOL', NWTN, K
  EXIT
ENDIF
CALL ATX(T, C, S, IC, JC, NNZ, M)
CALL DOT(VAL2, T, S, M)
CALL DOT(VAL3, T, T, M)
W = VAL2 / VAL3
CALL AXPBY(UP, ALPHA, P, W, S, M)
CALL AXPBY(X, 1.0D0, X, 1.0D0, UP, M)
CALL AXPBY(R, 1.0D0, S, -W, T, M)
IF(W .EQ. 0.0D0 ) THEN
  PRINT*, 'I CAN NOT CONTINUE ITERATION BECAUSE W = 0'
  EXIT
ENDIF
CALL NORM(M, UP, XUP)
CALL NORM(M, R, XR)
IF( XUP .LT. TOL .AND. XR .LT. TOL ) THEN
  K = I
  PRINT*, 'I STOP BECAUSE NORM(R,UP)<TOL', NWTN, K
  EXIT
ENDIF
ROWOLD = ROW
10 CONTINUE
END
C-----------------------------
SUBROUTINE DOT(VAL, X, Y, M)
C
C THIS SUBROUTINE COMPUTES THE DOT PRODUCT
C BETWEEN THE TWO VECTORS X AND Y
C
INTEGER I, M
DOUBLE PRECISION X(M), Y(M), VAL
VAL = 0.0D0
DO 10 I = 1, M
  VAL = VAL + X(I) * Y(I)
10 CONTINUE
END
C
SUBROUTINE EQUAL(X, Y, M)
DOUBLE PRECISION X(M),Y(M)
INTEGER M,I
DO 10 I=1,M
  X(I) = Y(I)
10 CONTINUE
END

C
SUBROUTINE AXPBY(Z,ALPHA,X,BETA,Y,M)
C
C THIS SUBROUTINE WILL DO THE FOLLOWING ;
C  \[ Z = \alpha X + \beta Y \]
C WHERE \( \alpha \) AND \( \beta \) ARE CONSTANTS
C AND \( X, Y, Z \) ARE VECTORS
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION Z(M),X(M),Y(M)
DO 10 I=1,M
  Z(I) = ALPHA*X(I) + BETA*Y(I)
10 CONTINUE
END

C----------------------------------
SUBROUTINE ATX(Y,A,X,IA,JA,NNZ,M)
C
C THIS SUBROUTINE WILL DO MATRIX-VECTOR MULTIPLICATION
C BETWEEN THE MATRIX (A,IA,JA) AND THE VECTOR X AND
C STORE THE RESULT IN THE VECTOR Y WHERE A ARE STORED IN
C SPARSE FORM (I.E)
C A CONTAINS ALL NONZERO ENTRIES AND
C IA CONTAINS THE ROW LOCATION OF THE NONZERO ENTRIES AND
C JA CONTAINS THE COLUMN LOCATION OF THE NONZERO ENTRIES.
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION IA(NNZ),JA(NNZ)
DIMENSION X(M),Y(M),A(NNZ)
CALL RZEROS(M,Y)
DO 10 K=1,NNZ
  I=IA(K)
  J=JA(K)
  AIJ=A(K)
  Y(I)= Y(I) + AIJ*X(J)
SUBROUTINE STORE(A,IA,JA,NNZ,IROW,JCOL,AIJ,INDEX)

C
C THIS SUBROUTINE WILL STORE ANY ELEMENT CONTRIBUTION
C AJI TO THE JACOBIAN OR THE FINE LEVEL MATRIX IN A IA JA.
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION A(NNZ),IA(NNZ),JA(NNZ)
CALL CHK(NNZ,IA,JA,IROW,JCOL,IFLAG,INDEX)
IF( IFLAG .EQ. 0) THEN
    INDEX = INDEX + 1
    A(INDEX) = AIJ
    IA(INDEX) = IROW
    JA(INDEX) = JCOL
ELSE
    A(IFLAG) = A(IFLAG) + AIJ
ENDIF
END

SUBROUTINE CHK(NNZ,IA,JA,IROW,JCOL,IFLAG,INDEX)

C
C THIS SUBROUTINE WILL CHECK IF THIS A(IROW, JCOL)
C HAVE BEEN ENTERED IN A IA JA BEFORE OR NOT.
C
IMPLICIT INTEGER (I-N)
DIMENSION IA(NNZ),JA(NNZ)
IFLAG = 0
DO 10 I=1,INDEX
    IF( IA(I) .EQ. IROW .AND. JA(I) .EQ. JCOL ) THEN
        IFLAG = I
        EXIT
    ENDIF
10 CONTINUE
END

SUBROUTINE NAME(RE,LX,LXO,F)

C
C THIS SUBRoutine WILL GIVE A FILE NAME WITH
C THE HELP OF RE,LX,LXO AND STORES THE NAME
C IN F.
C
CHARACTER*10 NO
CHARACTER*11 F
DOUBLE PRECISION RE
NO(1:10)=’1234567890’
C --------------------------------
J=RE
K4=J/1000
J3=J-K4*1000
K3=J3/100
J2=J-K3*100
K2=J2/10
J1=J2-10*K2
K1=J1
IF(K1 .EQ. 0) K1=10
IF(K2 .EQ. 0) K2=10
IF(K3 .EQ. 0) K3=10
IF(K4 .EQ. 0) K4=10
C --------------------------------
LX2=LX/10
LX1=LX-LX2*10
IF(LX1 .EQ. 0) LX1=10
IF(LX2 .EQ. 0) LX2=10
C --------------------------------
LX02=LX0/10
LX01=LX0-LX02*10
IF(LX01 .EQ. 0) LX01=10
IF(LX02 .EQ. 0) LX02=10
C --------------------------------
F=’R’//NO(K4:K4)//NO(K3:K3)//NO(K2:K2)//NO(K1:K1)//
- ’H’//NO(LX2:LX2)//NO(LX1:LX1)//
- ’H’//NO(LX02:LX02)//NO(LX01:LX01)
END

SUBROUTINE BACKUP(SOL,M,LX,LXO,RE,RESDA,RESDF,K)
C
C THIS SUBROUTINE WILL PRINT THE SOLUTION AND SOME
C STATISTICS TO A FILE NAME GENERATED BY NAME SUBROUTINE
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Y)
IMPLICIT INTEGER (I-N)
DIMENSION SOL(M)
CHARACTER*11 Z
CALL NAME(RE,LX,LX0,Z)
OPEN(UNIT=10,FILE=Z,STATUS=’NEW’)
WRITE (10,*) ’CAVITY PROBLEM WITH 2-LEVEL’
WRITE (10,*) ’LX = ’,LX,’ LX0 = ’,LX0
WRITE (10,*) 'RE = ', RE
WRITE (10,*) '# OF ITERATIONS = ', K
WRITE (10,*) ' RESIDUAL(B-AX) = ', RESDA
WRITE (10,*) ' RESIDUAL F(X) = ', RESDF
WRITE (10,*) ' ---------------------------- '
DO 30 J=1,M
   WRITE(10,*) SOL(J)
30 CONTINUE
CLOSE(UNIT=10)

END

SUBROUTINE BACKUPC(I,SOL,ER,ITER,MAX,M,LX,RE)
C
C THIS SUBROUTINE WILL PRINT THE SOLUTION AND SOME
C STATISTICS TO A FILE NAME GENERATED BY NAME SUBROUTINE
C
IMPLICIT DOUBLE PRECISION (A-H,O-Y)
IMPLICIT INTEGER (I-N)
DIMENSION SOL(M),ER(0:MAX,2),ITER(0:MAX)
CHARACTER*10 ZF

CALL NAMEC(RE,LX,I,ZF)
OPEN(UNIT=22,FILE=ZF,STATUS='NEW')
WRITE(22,*) 'LX = ',LX,' RE = ',RE
DO 40 K=0,I
  WRITE(22,*) K,ITER(K),ER(K,1),ER(K,2)
40 CONTINUE
DO 30 J=1,M
  WRITE(22,*) SOL(J)
30 CONTINUE
CLOSE(UNIT=20)
END

C --------------------------------
SUBROUTINE NAMEC(RE,LX,ITER,F)
C
C THIS SUBROUTINE WILL GIVE A FILE NAME WITH
C THE HELP OF RE,LX,LXO AND STORES THE NAME
C IN F.
C
CHARACTER*10 NO
CHARACTER*10 F
DOUBLE PRECISION RE

NO(1:10)=’1234567890’
C --------------------------------
J=RE
K3=J/100
J2= J-K3*100
K2=J2/10
J1=J2-10*K2
K1=J1
IF(K1 .EQ. 0) K1=10
IF(K2 .EQ. 0) K2=10
IF(K3 .EQ. 0) K3=10
C --------------------------------
LX2=LX/10
LX1=LX-LX2*10
IF(LX1 .EQ. 0) LX1=10
IF(LX2 .EQ. 0) LX2=10
C --------------------------------
IT2=ITER/10
IT1=ITER-IT2*10
IF(IT1 .EQ. 0) IT1=10
IF(IT2 .EQ. 0) IT2=10
C --------------------------------
F='R'//NO(K3:K3)//NO(K2:K2)//NO(K1:K1)://
- 'X'//NO(LX2:LX2)//NO(LX1:LX1)//
- 'I'//NO(IT2:IT2)//NO(IT1:IT1)
END
C ----------------------------
C
C THE FOLLOWING SUBROUTINES AFTER THIS LINE
C ARE NOT GENERAL SUBROUTINES. SOME OF THEM
C ARE FOR THE TEST PROBLEM WITH A KNOWN SOLUTION
C AND THE OTHER ARE FOR CAVITY FLOW PROBLEM.
C THE REST IS FOR DIFFERENT WAYS OF GLOBAL ORDERING.
C
C NOTE:
C 1-FOR TEST PROBLEM:
C Compile the above lines with F1(TEST) and
C F2(TEST) and GUISS(TEST)
C 2-FOR CAVITY PROBLEM:
C Compile the above lines with F1(CAVITY) and
C F2(CAVITY) and GUISS(CAVITY)
C 3-FOR ANY OTHER PROBLEM WITH [0,1]^2 DOMAIN
C Change some lines in MAELNODE SUBROUTINE
C TO COMPUTE IBOY ARRAY. IBOY CONTAINS NONHOMOG.
C BOUNDARY CONDITIONS AND
C SUPPLY F1, F2 AND GUISS WHERE F1 AND F2
BODY FORCE FUNCTION OF NSE. SUPPLY GUISS SUBROUTINE
 WHICH GENERATE A VECTOR. THIS VECTOR SATISFY THE
 BOUNDARY CONDITIONS OF THE PROBLEM AND THE OTHER
 ENTRIES OF THE VECTOR ARE ZEROS. THEN COMPILE
 THE ABOVE AFTER CHANGE WITH THE NEW MELNODE, F1
 F2 AND GUISS
 4-FOR ANY GENERAL PROBLEM
 REWRITE MELNODE, COORD, F1, F2 AND GUISS
 THEN COMPILE THE ABOVE WITH THESE NEW ONES AFTER
 YOU DELETE COORD AND MELNODE FROM ABOVE

SUBROUTINE F1(FF,X,Y,RE)

THIS SUBROUTINE COMPUTE THE FIRST COMPONENT OF
THE BODY FORCE FUNCTION IN THE NAVIER-STOKES EQUATIONS
IN PARTICULAR, IT IS F1(X,Y) AFTER WE SUBSTITUTE
THE STREAM FUNCTION PSI(X,Y) TO BE
X^2 * (X - 1)^2 * Y * (Y - 1)^2
 (TEST)

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
FF=X^2*(3 + 64*(-1 + X)**7*X**5*(-1 + 2*X)*
 - (1 - 2*Y)**2*(-1 + Y)**6*Y**6 +
 - 16*(1 - 2*X)*(-1 + X)**7*X**5*(-1 + Y)**6*Y**6*
 - (3 - 14*Y + 14*Y**2) -
 - (8*(-1 + X)**2*Y*(1 - 3*Y + 2*Y**2)*
 - (6*(-1 + Y)**2*Y**2 -
 - 28*X*(-1 + Y)**2*Y**2 -
 - 6*X**3*(1 - 7*Y + 7*Y**2) +
 - 3*X**4*(1 - 7*Y + 7*Y**2) +
 - X**2*(3 - 21*Y + 49*Y**2 - 56*Y**3 +
 - 28*Y**4)))/RE)
END

SUBROUTINE F2(FF,X,Y,RE)

THIS SUBROUTINE COMPUTE THE SECOND COMPONENT OF
THE BODY FORCE FUNCTION IN THE NAVIER-STOKES EQUATIONS
IN PARTICULAR, IT IS F1(X,Y) AFTER WE SUBSTITUTE
THE STREAM FUNCTION $\psi(x,y)$ TO BE

$$x^2 \cdot (x - 1)^2 \cdot y \cdot (y - 1)^2$$

TEST

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)

$$ff = y^2 \cdot (3 + 64 \cdot (1 - 2 \cdot x)^2 \cdot (-1 + x)^6 \cdot x^6 \cdot$$
$$- (1 + y)^7 \cdot y^5 \cdot (-1 + 2 \cdot y) -$$
$$- 16 \cdot (-1 + x)^6 \cdot y^6 \cdot (-1 + y)^7 \cdot y^5 \cdot (-1 + 2 \cdot y) +$$
$$- (8 \cdot x \cdot (-1 + 3 \cdot x + 2 \cdot x^2))^2 \cdot (-1 + y)^2 \cdot$$
$$- (3 \cdot (-1 + y)^2 \cdot y^2 -$$
$$- 21 \cdot x \cdot (-1 + y)^2 \cdot y^2 -$$
$$- 4 \cdot x^3 \cdot (3 - 14 \cdot x + 14 \cdot y^2) +$$
$$- x^4 \cdot (6 - 28 \cdot y + 28 \cdot y^2) +$$
$$- x^2 \cdot (6 - 28 \cdot y + 49 \cdot y^2 - 42 \cdot y^3 +$$
$$- 21 \cdot y^4)) / re)$$

END

SUBROUTINE GUISS (X0,LX,M,NBF,NBO,NBOY,NELEM,NPTS)

C
C THIS SUBROUTINE WILL PICK AN INITIAL GUISS WHICH
C SATISFIES THE BOUNDARY CONDITION OF THE PROBLEMS
C THIS INITIAL GUISS IS SUITABLE TO BE THE INITIAL
C GUISS FOR THE NEWTON ITERATION
C (TEST)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X0(M),IBO(NBO),IBOY(NBOY),IELNODE(NELEM,NBF)
CALL RZEROS(M,X0)
END

SUBROUTINE F1(FF,X,Y,RE)

C
C THIS SUBROUTINE COMPUTE THE FIRST COMPONENT OF
C THE BODY FORCE FUNCTION IN THE NAVIER-STOKES EQUATIONS
C (CAVITY)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)

FF = 0.0D0

END
SUBROUTINE F2(FF,X,Y,RE)
   C
   C THIS SUBROUTINE COMPUTE THE SECOND COMPONENT OF
   C THE BODY FORCE FUNCTION IN THE NAVIER-STOKES EQUATIONS
   C (CAVITY)
   C
   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
   IMPLICIT INTEGER (I-N)
   FF=0.0D0
   END

SUBROUTINE GUISS (X0,LX,M,NBF,NBO,NBOY,NELEM,NPTS)
   C
   C THIS SUBROTINE WILL PICK AN INITIAL GUISS WHICH
   C SATISFIES THE BOUNDARY CONDITION OF THE PROBLEMS
   C THIS INITIAL GUISS IS SUITABLE TO BE THE INITIAL
   C GUISS FOR THE NEWTON ITERATION (CAVITY PROBLEM)
   C
   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
   IMPLICIT INTEGER (I-N)
   DIMENSION X0(M),IBO(NBO),IBOY(NBOY),IELNODE(NELEM,NBF)
   CALL MAELNODE(IELNODE,IBO,IBOY,LX,
     - NBO,NBOY,NBF,M,NELEM,NPTS)
   CALL RZEROS(M,X0)
   DO 10 II=1,NBOY
     I=IBOY(II)
     X0(I) = 1.0D0
   10 CONTINUE
   END

C
C BELOW THIS LINE FOR DIFFRENT GLOBAL ORDERING
C
SUBROUTINE PNORM(N,X,XNORM,K)
   C
   C THIS SUBROUTINE WILL COMPUTE THE EUCLIDEAN
   C NORM OF THE VECTOR X(1:K)
   C
   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
   IMPLICIT INTEGER (I-N)
   DIMENSION X(N)
   SUM = 0.0D0
   DO 10 I=1,K
     10 CONTINUE
SUM = SUM + X(I)**2
10 CONTINUE
XNORM = DSQRT( SUM )
END

SUBROUTINE MAELNODE(IELNODE,IBO,IBOY,LX,-
                     NBO,NBOY,NBF,M,NELEM,NPTS)
C -------------------------------
C THIS SUBROUTINE COMPUTES THE ELEMENT NODE MATRIX OF
C THE BOGNER-FOX ELEMENTS AND STORE THEM IN IELNODE
C THE BOUNDARY POINTS AND STORE THEM IN IBO AND IBOY.
C POINTS.NOTE THAT THE DOMAIN IS [-1,1]X[-1,1].
C LX = # OF SQUARES ALONG X-AXIS
C NELEM = # OF ELEMENTS = LX^2
C NPTS = # OF POINTS = (LX+1)^2
C H = 2.0 / LX
C NBF = # OF BASIS FUNCTIONS IN EACH ELEMENT = 16
C NBO = # OF BOUNDARY POINTS =
C NNBO = # OF NORMAL BOUNDARY
C ELNODE = EACH ROW CONTAINS THE GLOBAL NUMBERING OF LOCAL BASIS FNC.
C IBO = CONTAINS ALL GLOBAL BOUNDARY NUMBERS EXCEPT THE TOP
C WITH Y-DERIVATIVE NODE
C IBOY = CONTAINS GLOBAL BOUNDARY NUMBERS ON THE TOP WITH
C Y-DERIVATIVE NODE
C
C NOTE : THE GLOBAL NUMBERING IS GOING FROM LOWER LEFT CORNER
C TO THE TOP RIGHT CORNER STARTING BY LISTING THE FUNCTION
C NODE THEN X-DERIVATIVE NODE THEN Y-DERIVATIVE NODE THEN
C XY-DERIVATIVE NODE.
C -------------------------------
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION IELNODE(NELEM,NBF),IBO(NBO),IBOY(NBOY)
NO = 0
DO 10 I=1,LX
   DO 20 J=1,LX
      IE = (I-1) * LX + J
      K = IE + NO
      IELNODE(IE,1) = K
      IELNODE(IE,2) = K+1
      IELNODE(IE,3) = K+1+LX
      IELNODE(IE,4) = K+2+LX
      IELNODE(IE,5) = K+NPTS
      IELNODE(IE,6) = K+NPTS+1
      IELNODE(IE,7) = K+LX+1+NPTS
      IELNODE(IE,8) = K+LX+2+NPTS
10 CONTINUE
20 CONTINUE
IELNODE(IE,9) = K+2*NPTS
IELNODE(IE,10) = K+2*NPTS+1
IELNODE(IE,11) = K+LX+1+2*NPTS
IELNODE(IE,12) = K+LX+2+2*NPTS
IELNODE(IE,13) = K+3*NPTS
IELNODE(IE,14) = K+1+3*NPTS
IELNODE(IE,15) = K+LX+1+3*NPTS
IELNODE(IE,16) = K+LX+2+3*NPTS

20 CONTINUE
NO = NO + 1
10 CONTINUE

C ---------------------------------
C COMPUTE BOUNDARY NODE
C ---------------------------------
DO 30 I=1,LX-1
   IBO(I) = I+1
   IBO(I+LX-1) = (LX+1)*LX + I + 1
30 CONTINUE
DO 40 I=1,LX+1
   IBO(2*LX-2+I) = (I-1)*(LX+1) + 1
   IBO(3*LX-1+I) = I*(LX+1)
40 CONTINUE

C ---------------------------------
C COMPUTE NORMAL BOUNDARY NODE
C ---------------------------------
DO 50 I=1,4*LX
   IX=I+4*LX
   IXY=I+8*LX
   IBO(IX) = IBO(I) + NPTS
   IBO(IXY) = IBO(I) + 3*NPTS
50 CONTINUE
DO 60 I=1,LX+1
   IY=I+12*LX
   IBO(IY) = I + 2*NPTS
60 CONTINUE
DO 70 I=2,LX
   IY = I + 13*LX
   IBO(IY) = I*(LX+1) + 2*NPTS
70 CONTINUE
DO 80 I=2,LX
   IY = I - 1 + 14*LX
   IBO(IY) = (I-1)*(LX+1) + 1
80 CONTINUE
DO 90 I=0,LX
   IBOY(I+1)= 3*NPTS - I

90 CONTINUE
SUBROUTINE COORD(X,Y,LX,M)
C
C THIS SUBROUTINE WILL COMPUTE THE X-COORDINATE
C AND Y-COORDINATE OF ALL NODES AND STORE THEM
C IN X AND Y ARRAY.
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION X(M),Y(M)
NPTS = M/4
H = 2.0D0/DBLE(FLOAT(LX))
Y0 = -1.0D0
DO 10 I=1,LX+1
   X0 = -1.0D0
   DO 20 J=1,LX+1
      IP = (I-1)*(LX+1) + J
      X(IP) = X0
      Y(IP) = Y0
      X0 = X0 + H
  20 CONTINUE
   Y0 = Y0 + H
10 CONTINUE
DO 30 I=1,NPTS
   IX = I+NPTS
   IY = I+2*NPTS
   IXY = I+3*NPTS
   X(IX) = X(I)
   X(IY) = X(I)
   X(IXY) = X(I)
   Y(IX) = Y(I)
   Y(IY) = Y(I)
   Y(IXY) = Y(I)
30 CONTINUE
END

SUBROUTINE FINDK(XM,YM,LX,K)
C
C GIVEN A POINT (XM,YM)
C K WILL THE ELEMENT NUMBER WHERE THAT POINT
C LIVE IN (LX+1) X (LX+1) GRID POINTS
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
IMPLICIT INTEGER (I-N)
H = 2.0D0/DBLE(FLOAT(LX))
X = -1.0D0+H
Y = -1.0D0+H
KX =1
KY=1
DO 10 I= 2,LX+1
   IF( X .GT. XM ) THEN
      KX = I-1
      EXIT
   ENDIF
   X = X + H
10 CONTINUE
DO 20 J=2,LX+1
   IF( Y .GT. YM ) THEN
      KY=J -1
      EXIT
   ENDIF
   Y = Y + H
20 CONTINUE
K = (KY -1)*LX + KX
END
Appendix C

MATLAB and M-files
function [sol] = fsolve(lx,Re,solo)

% This function will take solo vector as the
% the solution in the coarse mesh and compute
% the matrix a and rhs and solve the
% linear system to generate the vector sol as
% the solution of the problem in the fine mesh
% lx = # of squares along the x-axis in the
% fine mesh
% solo = the computed solution from csolve
%       function i.e coarse solution with lxo
% Re  = Reynolds number
% sol = the solution in the fine mesh
% ----------------------------------------

tol= .001;
lxo = lx/2;
npts=(lx+1)^2;
m=4*(lx+1)^2;
[elnode,bo,boy]=maelnode(lx);
[elnodeo,boo,boyo]=maelnode(lxo);
[xx,yy]=coord(lx);
[ea] = maea;
[b] = mab;

x0 = zeros([m,1]);
x0(boy) = x0(boy) + 1;

[a] = fstiff(lx,Re,elnode,xx,
    yy,lxo,elnodeo,solo,ea,b);

rhs = zeros([m,1]);
rhs = rhs - a*x0;
[a,rhs] = fboundry(a,rhs,bo,boy);
[sol,k] = bicgstab(rhs,tol,a);
csol = sol(1:npts);
[aa]=mat(csol);
contour(aa,10)
----------------------------------

function [sol,er,iter] = csolve(lx,Re,x0,start)

% This function will solve the Navier-Stokes
% Equation using one level finite element method
% giving the following :
% lx = # of squares along x-axis
% Re = Reynolds number
% x0 = initial guiss
% start = Newton iteration number
% sol = solution vector
% er(i,1) = i-th residual error
% er(i,2) = norm( sol_i+1 - sol_i )
% iter(i) = BICGSTAB iteration # needed at i-th Newton step
% The output solution is stored in sol as below:
% sol(1:npts) = psi values at (lx+1)^2 points
% sol(npts+1:2*npts) = psi_x values at (lx+1)^2 points
% sol(2*npts+1:3*npts) = psi_y values at (lx+1)^2 points
% sol(3*npts+1:4*npts) = psi_xy values at (lx+1)^2 points
% 
h=2/lx;
tol= 1e-6;
max=20;
npts=(lx+1)^2;
m=4*(lx+1)^2;
[elnode,bo,boy]=maelnode(lx);
[xx,yy]=coord(lx);
[ea] = maea;
[b] = mab;
sol = x0;
%load sol
up=zeros([m,1]);
er1=1e10;
for i=start:max
    [a] = jacobs(lx,Re,sol,elnode,bo,ea,b);
    [rhs] = force(lx,Re,sol,elnode,bo,xx,yy,ea,b);
    [a,rhs] = boundry(a,rhs,bo,boy);
    older1=er1;
    er1 =norm(rhs)/m;
    disp(i);disp(er1)
ratio =er1/older1;
    if( er1 > older1)
        disp(’ no improvement for f(sol)’);
        break
    end
    [up,k] = bicgstab(rhs,tol,a);
    iter(i)=k;
    if( er1 > older1)
        sol = sol - 0.5*up;
        disp(’ I use half step ’);
    else
        break
    end
end
sol = sol - up;

end

er2 = norm(up)/m;
er(i,:) = [er1, er2];
backup(i, sol, er)
if(er1<tol & er2<tol)
    break;
end

end

csol = sol(1:npts);
[aa]=mat(csol);
contour(aa,10)

-------------------------------------

function [ea] = maea
%

% This function will return the array
% ea which contains all values of possible
% integral comes from the biharmonic term
% between the 16 basis functions. These values
% have been precomputed exactly using MATHEMATICA.
% ea(i,j) = the double integral over the
% the square [-1,1]x[-1,1] of the product of
% laplace of the i-th basis function and
% the laplace of the j-the basis function
%
aa = [-9288, -4563, -4563, -162, 4878, 3303, -153, 1422, 4878, ...
   -153, 3303, 1422, 2043, 468, 468, -1107, -4563, 9288, -162, ...
   -4563, -3303, -4878, -1422, 153, -153, 4878, 1422, 3303, ...
   -468, -2043, 1107, -468, -4563, -162, 9288, -4563, -153, ...
   1422, 4878, 3303, -3303, -1422, -4878, 153, -468, 1107, ...
   -2043, -468, -162, -4563, -4563, 9288, -1422, 153, -3303, ...
   -4878, -1422, -3303, 153, -4878, -1107, 468, 468, 2043, ...
   4878, -3303, -153, -1422, 6048, 1818, 252, 1332, 2043, ...
   -468, 468, 1107, 1848, 348, -252, -702, 3303, -4878, 1422, ...
   153, 1818, 6048, 1332, 252, 468, -2043, -1107, -468, 348, ...
   1848, -702, -252, -153, -1422, 4878, -3303, 252, 1332, 6048, ...
   1818, -468, -1107, -2043, 468, 252, 702, -1848, -348, 1422, ...
   153, 3303, -4878, 1332, 252, 1818, 6048, 1107, 468, -468, ...
   2043, 702, 252, -348, -1848, 4878, -153, -3303, -1422, 2043, ...
   468, -468, 1107, 6048, 252, 1818, 1332, 1848, -252, 348, ...
   -702, -153, 4878, -1422, -3303, -468, -2043, -1107, 468, ...
   252, 6048, 1332, 1818, 252, -1848, 702, -348, 3303, 1422, ...
   -4878, 153, 468, -1107, -2043, -468, 1818, 1332, 6048, 252, ...]
348, -702, 1848, -252, 1422, 3303, 153, -4878, 1107, -468, ...
468, 2043, 1332, 1818, 252, 6048, 702, -348, 252, -1848, ...
2043, -468, -468, -1107, 1848, 348, 252, 702, 1848, 252, ...
348, 702, 1408, -232, -232, -332, 468, -2043, 1107, 468, ...
348, 1848, 702, 252, -252, -1848, -702, -348, -232, 1408, ...
-332, -232, 468, 1107, -2043, 468, -252, -702, -1848, -348, ...
348, 702, 1848, 252, -232, -332, 1408, -232, -1107, -468, ...
-468, 2043, -702, -252, -348, -1848, -702, -348, -252, ...
-1848, -332, -232, -232, 1408] ;
for i=1:16
    for j=1:16
        ip = (i-1)*16+j;
        ea(i,j)=aa(ip)/3150;
    end
end

------------------------------
function shows(sol)

% This function will plot the 3-dimensional surface
% of the giving solution storeed in sol
% sol(1:npts) = psi values at (lx+1)^2 points
% sol(npts+1:2*npts) = psi_x values at (lx+1)^2 points
% sol(2*npts+1:3*npts) = psi_y values at (lx+1)^2 points
% sol(3*npts+1:4*npts) = psi_xy values at (lx+1)^2 points
%
m=length(sol);
npts=m/4;
csol = sol(1:npts);
[a]=mat(csol);
surfl(a);
shading interp;
colormap(pink);

------------------------------

function [a]=mat(v)

% This function change a vector v
% into a matrix a. This function is
% used by many other function
%

n=length(v);
k=sqrt(n);
for i=1:k
  for j=1:k
    m=(i-1)*k+j;
a(i,j)=v(m);
  end
end

-----------------------------------

function [b] = mab

% This function will read the file
% bb which contains all values of possible
% integral comes from the trilinear term
% between the 16 basis functions and store
% them in the array b(16,16,16).
% b(i,j,k) = b( phi_i ,phi_j ,phi_k)
% Note : This function will be Ok only
% in MATLAB version 5.0 since the
% older version does not support
% multi-dimensional array
%
load bb
for i=1:16
  for j=1:16
    for k=1:16
      ip=(i-1)*256+(j-1)*16+k;
b(i,j,k)=bb(ip)/88200;
    end
  end
end

-----------------------------------

function [a,rhs] = fboundry(a,rhs,bo,boy)

% In this function, the contributions arrising
% from the boundary conditions are added in.
% i.e
% a(:,i)=0 and a(i,:) and a(i,i)=1
% for all i in the vector bo and boy
% rhs(i) = 0 for all i in vector bo
% rhs(i) = 1 for all i in vector boy

nbo = length(bo);
m = length(rhs);
lx = sqrt(m/4) - 1;
nboy = length(boy);
for ii=1:nbo
  i=bo(ii);
a(i,:)=zeros([1,m]);
a(:,i)=zeros([m,1]);
a(i,i)=1;
rhs(i)=0;
end
for ii=1:nboy
  i=boy(ii);
a(i,:)=zeros([1,m]);
a(:,i)=zeros([m,1]);
a(i,i)=1;
rhs(i)=1;
end

function [a,rhs] = boundry(a,rhs,bo,boy)

% In this function, the contributions arising
% from the boundary conditions are added in.
% i.e
% a(i,:)=0 a(:,i) a(i,i)=1
% for all i in vectors bo and boy
% rhs(i) = 0 for all i in bo and boy
% in this way no update on the boundary nodes
% for all Newton steps.

nbo = length(bo);
m = length(rhs);
lx = sqrt(m/4) - 1;
nboy = length(boy);
for ii=1:nbo
i=bo(ii);
a(i,:) = zeros([1,m]);
a(:,i) = zeros([m,1]);
a(i,i) = 1;
rhs(i) = 0;
end
for ii=1:nboy
    i = boy(ii);
    a(i,:) = zeros([1,m]);
    a(:,i) = zeros([m,1]);
    a(i,i) = 1;
    rhs(i) = 0;
end

----------------------------
function [q] = maq
%
% This function will read the file q1
% which contains the values of the 16 basis
% functions, the x-derivative, y-derivative
% xy-derivative, xx-derivative and yy-derivative
% at the integration points and store them in
% the array q(16,6,4).
% The first argument of q represents which basis
% functions the second argument means which
% derivative and the third argument represents
% which integration points
%
% Example :
% q(5,3,2) = phi5_y( xi(2) , eta(2) )
%
load q1
for i=1:16
    for j=1:6
        for k=1:4
            n = (i-1)*24+(j-1)*4+k;
            q(i,j,k) = q1(n);
        end
    end
end

----------------------------

function [a] = fstiff(lx,Re,elnode,...
% This function will take the input vector solo to compute the matrix a in a sparse form. This a is the matrix coming from step 2 in the algorithm i.e the resulting matrix needed to solve the linear system in the fine mesh. % lx = # of squares along x-axis in fine mesh
% Re = Reynolds number
% elnode = global labeling matrix in fine mesh
% xx = x-coordinate of all nodes
% yy = y-coordinate of all nodes
% lxo = # of squares along x-axis in coarse mesh
% elnodeo = global labeling matrix in fine mesh
% solo = the solution on coarse mesh.
% ( output from csolve function )
% ea = element stiffness matrix from maea function
% b = element 3-dimensional matrix from mab function
% a = the resulting matrix from step2 i.e resulting matrix from fine mesh.
%
\textbf{nbf} = 16;
\textbf{nelem} = lx*lx;
\textbf{m} = 4*(lx+1)^2;
\textbf{h} = 2 / lx;
\textbf{ratio} = lx/lxo;
\textbf{------------assemble stiff -------------}
a = \textbf{sparse}(m,m);
for \textbf{k}=1:nelem
  \textbf{x} = elnode(k,ix);
  \textbf{x} = xx(p);
  \textbf{y} = yy(p);
  \textbf{xm} = ( x(1)+x(2) )/2;
  \textbf{ym} = ( y(1)+y(3) )/2;
  for \textbf{ir} = 1:nbf
    \textbf{i} = elnode(k,ir);
    \textbf{j} = elnode(k,is);
    \textbf{a}(i,j) = a(i,j) + (1/Re)*ea(ir,is)/h^2;
  end
end
\textbf{sum} = 0;
for \textbf{ik}=1:nbf
  \textbf{ko} = findk(xm,ym,lxo);
kk=elnodo(ko,ik);
sum=sum + solo(kk)*b(ik,ir,is) ;
end
sum = sum/(ratio*h^2);
a(i,j) = a(i,j) + sum;
end
end
end

----------------------------------
function k = findk(xm,ym,lx)
% % Given a point (xm,ym) in (lx+1)X(lx+1) grid
% points the function will return the element
% number k such that (xm,ym) inside the k-the
% square. This function is needed by many
% functions.
% h = 2/lx ;
x = -1+h ; y = -1+h ;
kx =1;ky=1;
for i= 2:lx+1
  if( x > xm )
    kx = i-1 ;
    break ;
  end
  x = x + h;
end
for j = 2:lx+1
  if( y > ym)
    ky=j -1 ;
    break;
  end
  y = y + h;
end
k = (ky -1)*lx + kx;

----------------------------------

function [x,k] = bicgstab(b,tol,c)
% % this function will solve any linear system of
% equation with the rhs stored in b and the matrix
% are stored in c. this function solves the linear
% system c x = b using a bicgstab algorithm.
% this algorithm is described in the following
% reference:
% t. chan j. demmel j. donator j. doncarra v. eijkhout
% r. pozo c. romine r. barrett m. berry and h. van dev vorst.
% , templates: for the solution of linear systems:
% building blocks for iterative methods.
% the approximation solution is stored in the vector
% x(m) and use this x as the initial guess. The matrix
% is stored in sparse form. k is the number of iteration
% needed to reach the required tol.
%  
k=0;
x = zeros(length(b),1) ;
r = c*x ;
r = b-r ;
rh= r ;
for i=1:100
    row = rh'*r ;
    if(row == 0 ) break; end;
    if(i == 1)
        p = r ;
    else
        beta = (row/rowold)*(alpha/w) ;
        p = r + beta*(p-w*v) ;
    end
    p = p ;
v = c*p ;
alpha = row/(rh'*v) ;
s = r - alpha*v ;
if( norm(s)<tol)
    k=i ;
    x=x+alpha*p ;
    disp('I stop because norm(s)<tol');
    break ;
end
s = s ;
t = c*s ;
w = (t'*s)/(t'*t) ;
up = alpha*p +w*s ;
x = x + up ;
r = s - w*t ;
if(w == 0 )
disp('I can not continue iteration because w = 0');
break ;
end
if( norm(up)<tol & norm(r)<tol )
k = i ;
disp('I stop because norm(r,up)<tol');
break
end
rowold=row ;
end

-------------------------------------
function [x0] = guiss(lx)
%
% This function will take the input lx
% and returns the vector x0 which is zeros
% every where and satisfy the boundry conditions
%
m=4*(lx+1)^2;
[elnode,bo,boy]=maelnode(lx);
x0 = zeros([m,1]);
x0(boy) = x0(boy) + 1;

-----------------------------------
function [a] = jacobs(lx,Re,sol,elnode,bo,ea,b)
%
% This function will compute the jacobian matrix of the
% problem and store it in the sparse matrix a.
% lx = number of squares along x-axis
% Re = Reynolds number
% sol = i-th iteration from i-th Newton step
% elnode = global labeling matrix
% bo = boundary nodes
% ea = element stiffness matrix
% b = element 3-dimensional array from mab function
% a = jacobian matrix at i-th Newton step
%
nbf = 16;
nelem = lx^2;
m = 4*(lx+1)^2;
h = 2 / lx;

%------------assemble Jacobian ----------------
a = sparse(m,m);

for k=1:nelem
    for ir =1:nbf
        i = elnode(k,ir);
        for is =1:nbf
            j=elnode(k,is);
            a(i,j) = a(i,j) + (1/Re)*ea(ir,is)/h^2;
        end
        sum = 0;
        for ik=1:nbf
            kk=elnode(k,ik);
            brack = b(ir,is,ik) + b(ir,ik,is);
            sum=sum + sol(kk)*brack/h^2;
        end
        a(i,j) = a(i,j) + sum;
    end
end

------------------------------------

function [elnode,bo,boy] = maelnode(lx)
%
% This function computes the element node matrix of
% the Bogner-Fox-Schemidt elements and store them in elnode
% the boundary points and store them in bo and boy.
% Note that the domain is [-1,1]x[-1,1].
% lx = # of squares along x-axis
% nelem = # of elements = lx^2
% npts = # of points = (lx+1)^2
% h = 2.0 / lx
% nbf = # of basis functions in each element = 16
% nbo = # of boundary points =
% elnode = the global labeling matrix
% bo = contains all global boundary numbers except the top
% with y-derivative node
% boy = contains global boundary numbers on the top with
% y-derivative node
%
% note : the global numbering is going from lower left corner
% to the top right corner starting by listing the function
% node then x-derivative node then y-derivative node then
% xy-derivative node.
%
npts=(lx+1)^2;
m=4*npts;
n0 = 0;
for i=1:lx
    for j=1:lx
        ie = (i-1) * lx + j;
        k = ie + n0;
        elnode(ie,1) = k;
        elnode(ie,2) = k+1;
        elnode(ie,3) = k+1+lx;
        elnode(ie,4) = k+2+lx;
        elnode(ie,5) = k+npts;
        elnode(ie,6) = k+npts+1;
        elnode(ie,7) = k+lx+1+npts;
        elnode(ie,8) = k+lx+2+npts;
        elnode(ie,9) = k+2*npts;
        elnode(ie,10) = k+2*npts+1;
        elnode(ie,11) = k+1+2*npts;
        elnode(ie,12) = k+lx+2+2*npts;
        elnode(ie,13) = k+3*npts;
        elnode(ie,14) = k+1+3*npts;
        elnode(ie,15) = k+lx+1+3*npts;
        elnode(ie,16) = k+lx+2+3*npts;
    end
    n0 = n0 + 1;
end

% Compute boundary node
% -------------------------------------
for i=1:lx-1
    bo(i) = i+1;
    bo(i+lx-1) = (lx+1)*lx + i + 1;
end
for i=1:lx+1
    bo(2*lx-2+i) = (i-1)*(lx + 1) + 1;
    bo(3*lx-1+i) = i*(lx + 1);
end

% Compute normal boundary node
% -------------------------------------
for i=1:4*lx
    ix = i+4*lx;
    ixy = i+8*lx;
    bo(ix) = bo(i) + npts;
    bo(ixy) = bo(i) + 3*npts;
end
for i=1:lx+1
    iy = i+12*lx;
bo(iy) = i + 2*npts;
end
for i=2:lx
    iy = i + 13*lx;
    bo(iy) = i*(lx+1) + 2*npts;
end
for i=2:lx
    iy = i - 1 + 14*lx;
    bo(iy) = (i-1)*(lx+1) + 1;
end
for i=0:lx
    boy(i+1)= 3*npts - i;
end

--------------------------------

function [x,y] = coord(lx)
%
% This function will compute the x-coordinate
% and y-coordinate of all nodes and store them
% in x and y array.
% lx = number of square along the x-axis
% x  = the x-coordinates of all nodes
% y  = the y-coordinates of all nodes
%
 npts=(lx+1)^2;
 h = 2/lx;
 y0 = -1.0d0;
 for i=1:lx+1
    x0 = -1.0d0;
    for j=1:lx+1
        ip = (i-1)*(lx+1) + j;
        x(ip)= x0;
        y(ip)= y0;
        x0 = x0 + h;
    end
    y0 = y0 + h;
 end
 for i=1:npts
    ix=i+npts;
    iy=i+2*npts;
    ixy=i+3*npts;
    x(ix) = x(i);
    x(iy) = x(i);
    x(ixy) = x(i);
\[
y(i) = y(i);
y(i) = y(i);
y(i) = y(i);
\]

function \([\text{rhs}] = \text{force}(lx, \text{Re}, \text{sol}, \text{eln}, \text{bo}, \text{xx}, \text{yy}, \text{ea}, \text{b})\)

\%
\%
This function will take the input vector \(\text{sol}\) to compute \(F(\text{sol})\) and store it in \(\text{rhs}\) vector
\%
which will be the right hand side of the resulting
\%
linear system from the \(i\)-th newton iteration and
\%
in the same time it is the residual.
\%
\%
\(lx = \# \) of square along the \(x\)-axis
\%
\(\text{Re} = \) Reynolds number
\%
\(\text{sol}= i\)-th iteration from Newton method
\%
\(\text{eln}(\text{nelem},16) = \) global labeling matrix
\%
\(\text{bo} = \) boundary nodes
\%
\(\text{xx} = x\)-coordinates of all nodes
\%
\(\text{yy} = y\)-coordinates of all nodes
\%
\(\text{ea} = \) element stiffness matrix from function maea
\%
\(\text{b} = \) element 3-dimensional matrix from function
\%
\(\text{mab}\)
\%
\(\text{rhs} = \) the rhs vector of the resulting matrix in
\%
the \(i\)-th iteration before the boundary
\%
condition requirement.
\%
\(\text{nbf}=16;\)
\(h=2/\text{lx};\)
\(\text{nelem}=\text{lx}^2;\)
\(m=4*(\text{lx}+1)^2;\)
\(\text{rhs} = \text{zeros}([\text{m},1]);\)
for \(\text{nk}=1:\text{nelem}\)
  for \(\text{ii}=1:4\)
    \(x(\text{ii}) = \text{xx}(\text{eln}(\text{nk},\text{ii}));\)
  end
  for \(\text{ir}=1:\text{nbf}\)
    \(i=\text{eln}(\text{nk},\text{ir});\)
    \(\text{sum}=0;\)
    for \(\text{is}=1:\text{nbf}\)
      \(j=\text{eln}(\text{nk},\text{is});\)
      \(\text{sum} = \text{sum} + (1/\text{Re})*\text{ea}(\text{ir},\text{is})*\text{sol}(\text{j});\)
    end
  end
end

\[
\]
end
rhs(i) = rhs(i) + sum/h^2 ;
    sum=0 ;
for is=1:nbf
    j=elnode(nk,is) ;
    for ik=1:nbf
        k= elnode(nk,ik) ;
        sum=sum+b(ir,ik,is)*sol(k)*sol(j) ;
    end
end
rhs(i) = rhs(i) + sum/h^2 ;
end
end

----------------------------------
function [xi,eta,w] = squad(n)
%
% This function will return the integration points on the square [-1,1]x[-1,1] the x-coordinates of the points are stored in array xi and the y-coordinates stored in array eta and the wight are stored in w
%
p = 1/sqrt(3);
m = -p;
xi = [m;p;m;p];
eta = [m;m;p;p];
w = [1;1;1;1];

----------------------------------
function backup(i,sol,er)
%
% This function will save the i-th solution from Newton step into a file.
%
save errrr er -ascii
if i == 1
    save sol1 sol -ascii
elseif i == 2
    save sol2 sol -ascii
else i == 3
elseif i == 4
    save sol4 sol -ascii
elseif i == 5
    save sol5 sol -ascii
elseif i == 6
    save sol6 sol -ascii
elseif i == 7
    save sol7 sol -ascii
elseif i == 8
    save sol8 sol -ascii
elseif i == 9
    save sol9 sol -ascii
elseif i == 10
    save sol10 sol -ascii
elseif i == 11
    save sol11 sol -ascii
elseif i == 12
    save sol12 sol -ascii
elseif i == 13
    save sol13 sol -ascii
elseif i == 14
    save sol14 sol -ascii
elseif i == 15
    save sol15 sol -ascii
elseif i == 16
    save sol16 sol -ascii
elseif i == 17
    save sol17 sol -ascii
elseif i == 18
    save sol18 sol -ascii
elseif i == 19
    save sol19 sol -ascii
else
    save sol20 sol -ascii
end

function showc(sol,c)
%
% This function will plot the streamlines
% of the giving solution stored in sol
% sol(1:npts) = psi values at (lx+1)^2 points
% sol(npts+1:2*npts) = psi_x values at (lx+1)^2 points

\% sol(2*npts+1:3*npts) = psi\_y values at (lx+1)^2 points
\% sol(3*npts+1:4*npts) = psi\_xy values at (lx+1)^2 points
\% c = \# of streamlines to be plotted
\% or you can enter some entries in the vector v below
\% to plot the streamlines where psi(x,y) = v(i).
\%
\% m=length(sol);
npts=m/4;
csol = sol(1:npts);
[a]=mat(csol);
v= [ ...
.1;.2;.3;.4;.5;.6;.7;.8;.9; ...
-1;-2;-3;-4;-5;-6;-7;-8;-9; ...
1.1;1.4;1.6;1.8; ...
-1.1;-1.4;-1.6;-1.8; ...
-2.1;-2.4;-2.6;-2.8; ...
1;2;3;4;5;6;7;8; ...
-1;-2;-3;-4;-5;-6;-7;-8; ...
1e-2;2e-2;3e-2;5e-2;7e-2;9e-2; ...
-1e-2;-2e-2;-3e-2;-5e-2;-7e-2;-9e-2; ...
5e-3;5e-4;5e-6; ...
-5e-3;-5e-4;-5e-6 ...
];
contour(a,c);
Appendix D

Data Files
This is the content of bb.dat file

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17010</td>
<td>-17010</td>
<td>0</td>
<td>-3780</td>
<td>-3780</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3780</td>
<td>0</td>
<td>3780</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8505</td>
</tr>
<tr>
<td>8505</td>
<td>8505</td>
<td>378</td>
<td>7182</td>
<td>3402</td>
<td>-3402</td>
</tr>
<tr>
<td>-1890</td>
<td>-1890</td>
<td>-1890</td>
<td>-1890</td>
<td>-756</td>
<td>756</td>
</tr>
<tr>
<td>-756</td>
<td>756</td>
<td>-8505</td>
<td>-8505</td>
<td>25515</td>
<td>-8505</td>
</tr>
<tr>
<td>1890</td>
<td>1890</td>
<td>1890</td>
<td>1890</td>
<td>-378</td>
<td>-3402</td>
</tr>
<tr>
<td>-7182</td>
<td>3402</td>
<td>756</td>
<td>756</td>
<td>-756</td>
<td>-756</td>
</tr>
<tr>
<td>0</td>
<td>17010</td>
<td>-17010</td>
<td>0</td>
<td>1512</td>
<td>-5292</td>
</tr>
<tr>
<td>-5292</td>
<td>1512</td>
<td>-1512</td>
<td>5292</td>
<td>5292</td>
<td>-1512</td>
</tr>
<tr>
<td>0</td>
<td>-1512</td>
<td>1512</td>
<td>0</td>
<td>-21615</td>
<td>111</td>
</tr>
<tr>
<td>17835</td>
<td>3669</td>
<td>-7935</td>
<td>2505</td>
<td>4155</td>
<td>-1875</td>
</tr>
<tr>
<td>7050</td>
<td>1938</td>
<td>-7062</td>
<td>-1590</td>
<td>1830</td>
<td>-960</td>
</tr>
<tr>
<td>-1698</td>
<td>804</td>
<td>3291</td>
<td>18213</td>
<td>-7071</td>
<td>-14433</td>
</tr>
<tr>
<td>3639</td>
<td>-7935</td>
<td>-3009</td>
<td>4155</td>
<td>-1182</td>
<td>-7806</td>
</tr>
<tr>
<td>2346</td>
<td>6306</td>
<td>-708</td>
<td>1830</td>
<td>1056</td>
<td>-1698</td>
</tr>
<tr>
<td>2085</td>
<td>-2631</td>
<td>1695</td>
<td>-1149</td>
<td>1635</td>
<td>15</td>
</tr>
<tr>
<td>2145</td>
<td>-645</td>
<td>762</td>
<td>-930</td>
<td>-750</td>
<td>582</td>
</tr>
<tr>
<td>690</td>
<td>-48</td>
<td>-822</td>
<td>204</td>
<td>-771</td>
<td>1317</td>
</tr>
<tr>
<td>4551</td>
<td>-5097</td>
<td>-1119</td>
<td>1635</td>
<td>489</td>
<td>2145</td>
</tr>
<tr>
<td>174</td>
<td>-6</td>
<td>-1338</td>
<td>1506</td>
<td>-300</td>
<td>690</td>
</tr>
<tr>
<td>-48</td>
<td>-822</td>
<td>21615</td>
<td>-17835</td>
<td>-111</td>
<td>-3669</td>
</tr>
<tr>
<td>-7050</td>
<td>7062</td>
<td>-1938</td>
<td>1590</td>
<td>7935</td>
<td>-4155</td>
</tr>
<tr>
<td>-2505</td>
<td>1875</td>
<td>-1830</td>
<td>1698</td>
<td>960</td>
<td>-804</td>
</tr>
<tr>
<td>-2085</td>
<td>-1695</td>
<td>2631</td>
<td>1149</td>
<td>-762</td>
<td>750</td>
</tr>
<tr>
<td>930</td>
<td>-582</td>
<td>-1635</td>
<td>-2145</td>
<td>-15</td>
<td>645</td>
</tr>
<tr>
<td>-690</td>
<td>822</td>
<td>48</td>
<td>-204</td>
<td>-3291</td>
<td>7071</td>
</tr>
<tr>
<td>-18213</td>
<td>14433</td>
<td>1182</td>
<td>-2346</td>
<td>7806</td>
<td>-6306</td>
</tr>
<tr>
<td>-3639</td>
<td>3009</td>
<td>7935</td>
<td>-4155</td>
<td>708</td>
<td>-1056</td>
</tr>
<tr>
<td>-1830</td>
<td>1698</td>
<td>771</td>
<td>-4551</td>
<td>-1317</td>
<td>5097</td>
</tr>
<tr>
<td>-174</td>
<td>1338</td>
<td>6</td>
<td>-1506</td>
<td>1119</td>
<td>-489</td>
</tr>
<tr>
<td>-1635</td>
<td>-2145</td>
<td>300</td>
<td>48</td>
<td>-690</td>
<td>822</td>
</tr>
<tr>
<td>0</td>
<td>-6762</td>
<td>6762</td>
<td>0</td>
<td>-810</td>
<td>2238</td>
</tr>
<tr>
<td>1272</td>
<td>-180</td>
<td>810</td>
<td>-1272</td>
<td>-2238</td>
<td>180</td>
</tr>
<tr>
<td>0</td>
<td>402</td>
<td>-402</td>
<td>0</td>
<td>462</td>
<td>6300</td>
</tr>
<tr>
<td>-2520</td>
<td>-4242</td>
<td>138</td>
<td>-810</td>
<td>-1020</td>
<td>1272</td>
</tr>
<tr>
<td>264</td>
<td>198</td>
<td>576</td>
<td>1482</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>252</td>
<td>-402</td>
<td>-462</td>
<td>2520</td>
<td>-6300</td>
<td>4242</td>
</tr>
<tr>
<td>-264</td>
<td>-576</td>
<td>-198</td>
<td>-1482</td>
<td>-138</td>
<td>1020</td>
</tr>
<tr>
<td>810</td>
<td>-1272</td>
<td>-66</td>
<td>-252</td>
<td>0</td>
<td>402</td>
</tr>
<tr>
<td>0</td>
<td>-2058</td>
<td>2058</td>
<td>0</td>
<td>264</td>
<td>-264</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>618</td>
<td>-198</td>
<td>-264</td>
<td>-618</td>
<td>264</td>
<td>198</td>
</tr>
<tr>
<td>0</td>
<td>-66</td>
<td>66</td>
<td>0</td>
<td>25515</td>
<td>-8505</td>
</tr>
<tr>
<td>-8505</td>
<td>-8505</td>
<td>7182</td>
<td>378</td>
<td>-3402</td>
<td>3402</td>
</tr>
<tr>
<td>1890</td>
<td>1890</td>
<td>1890</td>
<td>1890</td>
<td>756</td>
<td>-756</td>
</tr>
<tr>
<td>756</td>
<td>-756</td>
<td>-17010</td>
<td>0</td>
<td>0</td>
<td>17010</td>
</tr>
<tr>
<td>-3780</td>
<td>-3780</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3780</td>
</tr>
<tr>
<td>0</td>
<td>-3780</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-17010</td>
<td>0</td>
<td>0</td>
<td>17010</td>
<td>-5292</td>
<td>1512</td>
</tr>
<tr>
<td>1512</td>
<td>-5292</td>
<td>-5292</td>
<td>1512</td>
<td>1512</td>
<td>-5292</td>
</tr>
<tr>
<td>-1512</td>
<td>0</td>
<td>0</td>
<td>1512</td>
<td>8505</td>
<td>8505</td>
</tr>
<tr>
<td>8505</td>
<td>-25515</td>
<td>1890</td>
<td>1890</td>
<td>1890</td>
<td>1890</td>
</tr>
<tr>
<td>3402</td>
<td>378</td>
<td>-3402</td>
<td>7182</td>
<td>756</td>
<td>756</td>
</tr>
<tr>
<td>-756</td>
<td>-756</td>
<td>18213</td>
<td>3291</td>
<td>-14433</td>
<td>-7071</td>
</tr>
<tr>
<td>7935</td>
<td>-3639</td>
<td>-4155</td>
<td>3009</td>
<td>-7806</td>
<td>-1182</td>
</tr>
<tr>
<td>6306</td>
<td>2346</td>
<td>-1830</td>
<td>708</td>
<td>1698</td>
<td>-1056</td>
</tr>
<tr>
<td>111</td>
<td>-21615</td>
<td>3669</td>
<td>17835</td>
<td>-2505</td>
<td>7935</td>
</tr>
<tr>
<td>1875</td>
<td>-4155</td>
<td>1938</td>
<td>7050</td>
<td>-1590</td>
<td>-7062</td>
</tr>
<tr>
<td>960</td>
<td>-1830</td>
<td>-804</td>
<td>1698</td>
<td>1317</td>
<td>-771</td>
</tr>
<tr>
<td>-5097</td>
<td>4551</td>
<td>-1635</td>
<td>1119</td>
<td>-2145</td>
<td>-489</td>
</tr>
<tr>
<td>-6</td>
<td>174</td>
<td>1506</td>
<td>-1338</td>
<td>-690</td>
<td>300</td>
</tr>
<tr>
<td>822</td>
<td>48</td>
<td>-2631</td>
<td>2085</td>
<td>-1149</td>
<td>1695</td>
</tr>
<tr>
<td>-15</td>
<td>-1635</td>
<td>645</td>
<td>-2145</td>
<td>-930</td>
<td>762</td>
</tr>
<tr>
<td>582</td>
<td>-750</td>
<td>48</td>
<td>-690</td>
<td>-204</td>
<td>822</td>
</tr>
<tr>
<td>1695</td>
<td>2085</td>
<td>-1149</td>
<td>-2631</td>
<td>750</td>
<td>-762</td>
</tr>
<tr>
<td>-582</td>
<td>930</td>
<td>2145</td>
<td>1635</td>
<td>-645</td>
<td>15</td>
</tr>
<tr>
<td>822</td>
<td>-690</td>
<td>-204</td>
<td>48</td>
<td>17835</td>
<td>-21615</td>
</tr>
<tr>
<td>3669</td>
<td>111</td>
<td>7062</td>
<td>-7050</td>
<td>1590</td>
<td>-1938</td>
</tr>
<tr>
<td>4155</td>
<td>-7935</td>
<td>-1875</td>
<td>2505</td>
<td>1698</td>
<td>-1830</td>
</tr>
<tr>
<td>-804</td>
<td>960</td>
<td>4551</td>
<td>-771</td>
<td>-5097</td>
<td>1317</td>
</tr>
<tr>
<td>1338</td>
<td>-174</td>
<td>-1506</td>
<td>6</td>
<td>489</td>
<td>-1119</td>
</tr>
<tr>
<td>2145</td>
<td>1635</td>
<td>48</td>
<td>300</td>
<td>822</td>
<td>-690</td>
</tr>
<tr>
<td>-7071</td>
<td>3291</td>
<td>-14433</td>
<td>18213</td>
<td>-2346</td>
<td>1182</td>
</tr>
<tr>
<td>-6306</td>
<td>7806</td>
<td>-3009</td>
<td>3639</td>
<td>4155</td>
<td>-7935</td>
</tr>
<tr>
<td>-1056</td>
<td>708</td>
<td>1698</td>
<td>-1830</td>
<td>6300</td>
<td>462</td>
</tr>
<tr>
<td>-4242</td>
<td>-2520</td>
<td>810</td>
<td>-138</td>
<td>-1272</td>
<td>1020</td>
</tr>
<tr>
<td>198</td>
<td>264</td>
<td>1482</td>
<td>576</td>
<td>0</td>
<td>-66</td>
</tr>
<tr>
<td>402</td>
<td>-252</td>
<td>-6762</td>
<td>0</td>
<td>0</td>
<td>6762</td>
</tr>
<tr>
<td>-2238</td>
<td>810</td>
<td>180</td>
<td>-1272</td>
<td>-1272</td>
<td>810</td>
</tr>
<tr>
<td>180</td>
<td>-2238</td>
<td>-402</td>
<td>0</td>
<td>0</td>
<td>402</td>
</tr>
<tr>
<td>-2058</td>
<td>0</td>
<td>0</td>
<td>2058</td>
<td>264</td>
<td>-264</td>
</tr>
<tr>
<td>198</td>
<td>-618</td>
<td>-618</td>
<td>-264</td>
<td>198</td>
<td>264</td>
</tr>
<tr>
<td>66</td>
<td>0</td>
<td>0</td>
<td>-66</td>
<td>2520</td>
<td>-462</td>
</tr>
<tr>
<td></td>
<td>-6300</td>
<td>576</td>
<td>264</td>
<td>1482</td>
<td>198</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>1020</td>
<td>-138</td>
<td>-1272</td>
<td>810</td>
<td>252</td>
<td>66</td>
</tr>
<tr>
<td>-402</td>
<td></td>
<td>-25515</td>
<td>8505</td>
<td>8505</td>
<td>8505</td>
</tr>
<tr>
<td>-1890</td>
<td>-1890</td>
<td>-1890</td>
<td>-1890</td>
<td>-7182</td>
<td>3402</td>
</tr>
<tr>
<td>-378</td>
<td>-3402</td>
<td>-756</td>
<td>-756</td>
<td>756</td>
<td>756</td>
</tr>
<tr>
<td>17010</td>
<td></td>
<td></td>
<td>-17010</td>
<td>5292</td>
<td>-1512</td>
</tr>
<tr>
<td>-1512</td>
<td>5292</td>
<td>5292</td>
<td>-1512</td>
<td>-1512</td>
<td>5292</td>
</tr>
<tr>
<td>1512</td>
<td></td>
<td>-1512</td>
<td>17010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-17010</td>
<td>0</td>
<td>0</td>
<td>3780</td>
<td>3780</td>
</tr>
<tr>
<td>3780</td>
<td></td>
<td>3780</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-3402</td>
<td>3402</td>
<td></td>
<td>-1890</td>
<td>-1890</td>
<td>-1890</td>
</tr>
<tr>
<td>-1890</td>
<td></td>
<td>-756</td>
<td>756</td>
<td>756</td>
<td>756</td>
</tr>
<tr>
<td>-1695</td>
<td>1149</td>
<td>-2085</td>
<td>2631</td>
<td>-2145</td>
<td>645</td>
</tr>
<tr>
<td>-1635</td>
<td></td>
<td>-15</td>
<td>-750</td>
<td>582</td>
<td>762</td>
</tr>
<tr>
<td>-822</td>
<td>204</td>
<td>690</td>
<td>-48</td>
<td>-4551</td>
<td>5097</td>
</tr>
<tr>
<td>771</td>
<td>-1317</td>
<td>-489</td>
<td>-2145</td>
<td>1119</td>
<td>-1635</td>
</tr>
<tr>
<td>-1338</td>
<td>1506</td>
<td>174</td>
<td>-6</td>
<td>0</td>
<td>-822</td>
</tr>
<tr>
<td>-300</td>
<td>690</td>
<td>-17835</td>
<td>-3669</td>
<td>21615</td>
<td>-111</td>
</tr>
<tr>
<td>-4155</td>
<td>1875</td>
<td>7935</td>
<td>-2505</td>
<td>-7062</td>
<td>-1590</td>
</tr>
<tr>
<td>7050</td>
<td>1938</td>
<td>-1698</td>
<td>804</td>
<td>1830</td>
<td>-960</td>
</tr>
<tr>
<td>7071</td>
<td>14433</td>
<td>-3291</td>
<td>-18213</td>
<td>3009</td>
<td>-4155</td>
</tr>
<tr>
<td>-3639</td>
<td>7935</td>
<td>2346</td>
<td>6306</td>
<td>-1182</td>
<td>-7806</td>
</tr>
<tr>
<td>1056</td>
<td>-1698</td>
<td>708</td>
<td>1830</td>
<td>-18213</td>
<td>14433</td>
</tr>
<tr>
<td>-3291</td>
<td>7071</td>
<td>7806</td>
<td>-6306</td>
<td>1182</td>
<td>-2346</td>
</tr>
<tr>
<td>-7935</td>
<td>4155</td>
<td>3639</td>
<td>-3009</td>
<td>1830</td>
<td>-1698</td>
</tr>
<tr>
<td>-708</td>
<td>1056</td>
<td>-1317</td>
<td>5097</td>
<td>771</td>
<td>-4551</td>
</tr>
<tr>
<td>6</td>
<td>-1506</td>
<td>-174</td>
<td>1338</td>
<td>1635</td>
<td>2145</td>
</tr>
<tr>
<td>-1119</td>
<td>489</td>
<td>690</td>
<td>-822</td>
<td>-300</td>
<td>-48</td>
</tr>
<tr>
<td>-111</td>
<td>-3669</td>
<td>21615</td>
<td>-17835</td>
<td>-1938</td>
<td>1590</td>
</tr>
<tr>
<td>-7050</td>
<td>7062</td>
<td>2505</td>
<td>1875</td>
<td>-7935</td>
<td>4155</td>
</tr>
<tr>
<td>-960</td>
<td>804</td>
<td>1830</td>
<td>-1698</td>
<td>2631</td>
<td>1149</td>
</tr>
<tr>
<td>-2085</td>
<td>-1695</td>
<td>930</td>
<td>-582</td>
<td>762</td>
<td>750</td>
</tr>
<tr>
<td>15</td>
<td>-645</td>
<td>1635</td>
<td>2145</td>
<td>-48</td>
<td>204</td>
</tr>
<tr>
<td>690</td>
<td>-822</td>
<td>-6300</td>
<td>4242</td>
<td>-462</td>
<td>2520</td>
</tr>
<tr>
<td>-198</td>
<td>-1482</td>
<td>-264</td>
<td>576</td>
<td>-810</td>
<td>1272</td>
</tr>
<tr>
<td>138</td>
<td>1020</td>
<td>0</td>
<td>-402</td>
<td>66</td>
<td>252</td>
</tr>
<tr>
<td>2058</td>
<td></td>
<td>0</td>
<td>0</td>
<td>618</td>
<td>-198</td>
</tr>
<tr>
<td>264</td>
<td>264</td>
<td>264</td>
<td>-198</td>
<td>264</td>
<td>618</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td>0</td>
<td>66</td>
<td>6762</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-6762</td>
<td>1272</td>
<td>-180</td>
<td>-810</td>
<td>2238</td>
</tr>
<tr>
<td>2238</td>
<td>-180</td>
<td>-810</td>
<td>1272</td>
<td>402</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>0</td>
<td>-402</td>
<td>-2520</td>
<td>-4242</td>
<td>462</td>
<td>6300</td>
</tr>
<tr>
<td>-1020</td>
<td>1272</td>
<td>138</td>
<td>-810</td>
<td>-576</td>
<td>-1482</td>
</tr>
<tr>
<td>-264</td>
<td>-198</td>
<td>-252</td>
<td>402</td>
<td>-66</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-17010</td>
<td>17010</td>
<td>0</td>
<td>-1512</td>
<td>5292</td>
</tr>
<tr>
<td>5292</td>
<td>-1512</td>
<td>1512</td>
<td>-5292</td>
<td>-5292</td>
<td>1512</td>
</tr>
<tr>
<td>0</td>
<td>1512</td>
<td>-1512</td>
<td>0</td>
<td>-8505</td>
<td>25515</td>
</tr>
<tr>
<td>-8505</td>
<td>-8505</td>
<td>-1890</td>
<td>-1890</td>
<td>-1890</td>
<td>0</td>
</tr>
<tr>
<td>-3402</td>
<td>7182</td>
<td>3402</td>
<td>378</td>
<td>-756</td>
<td>-756</td>
</tr>
<tr>
<td>756</td>
<td>756</td>
<td>8505</td>
<td>8505</td>
<td>-25515</td>
<td>8505</td>
</tr>
<tr>
<td>3402</td>
<td>-3402</td>
<td>-7182</td>
<td>-378</td>
<td>1890</td>
<td>1890</td>
</tr>
<tr>
<td>1890</td>
<td>1890</td>
<td>756</td>
<td>-756</td>
<td>756</td>
<td>-756</td>
</tr>
<tr>
<td>0</td>
<td>-17010</td>
<td>17010</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3780</td>
<td>3780</td>
<td>0</td>
<td>-3780</td>
<td>0</td>
<td>-3780</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5097</td>
<td>-4551</td>
</tr>
<tr>
<td>-1317</td>
<td>771</td>
<td>2145</td>
<td>489</td>
<td>1635</td>
<td>1119</td>
</tr>
<tr>
<td>1506</td>
<td>-1338</td>
<td>-6</td>
<td>174</td>
<td>822</td>
<td>48</td>
</tr>
<tr>
<td>-690</td>
<td>300</td>
<td>1149</td>
<td>-1695</td>
<td>2631</td>
<td>-2085</td>
</tr>
<tr>
<td>-645</td>
<td>2145</td>
<td>15</td>
<td>1635</td>
<td>582</td>
<td>-750</td>
</tr>
<tr>
<td>-930</td>
<td>762</td>
<td>-204</td>
<td>822</td>
<td>48</td>
<td>-690</td>
</tr>
<tr>
<td>14433</td>
<td>7071</td>
<td>-18213</td>
<td>-3291</td>
<td>4155</td>
<td>-3009</td>
</tr>
<tr>
<td>-7935</td>
<td>3639</td>
<td>6306</td>
<td>2346</td>
<td>-7806</td>
<td>-1182</td>
</tr>
<tr>
<td>1698</td>
<td>-1056</td>
<td>-1830</td>
<td>708</td>
<td>-3669</td>
<td>-17835</td>
</tr>
<tr>
<td>-111</td>
<td>21615</td>
<td>-1875</td>
<td>4155</td>
<td>2505</td>
<td>-7935</td>
</tr>
<tr>
<td>-1590</td>
<td>-7062</td>
<td>1938</td>
<td>7050</td>
<td>-804</td>
<td>1698</td>
</tr>
<tr>
<td>960</td>
<td>-1830</td>
<td>-5097</td>
<td>1317</td>
<td>4551</td>
<td>-771</td>
</tr>
<tr>
<td>-1506</td>
<td>6</td>
<td>1338</td>
<td>-174</td>
<td>-2145</td>
<td>-1635</td>
</tr>
<tr>
<td>-489</td>
<td>1119</td>
<td>-822</td>
<td>690</td>
<td>-48</td>
<td>-300</td>
</tr>
<tr>
<td>-14433</td>
<td>18213</td>
<td>-7071</td>
<td>3291</td>
<td>-6306</td>
<td>7806</td>
</tr>
<tr>
<td>-2346</td>
<td>1182</td>
<td>-4155</td>
<td>7935</td>
<td>3009</td>
<td>-3639</td>
</tr>
<tr>
<td>-1698</td>
<td>1830</td>
<td>1056</td>
<td>-708</td>
<td>-1149</td>
<td>-2631</td>
</tr>
<tr>
<td>1695</td>
<td>2085</td>
<td>-582</td>
<td>930</td>
<td>750</td>
<td>-762</td>
</tr>
<tr>
<td>645</td>
<td>-15</td>
<td>-2145</td>
<td>-1635</td>
<td>204</td>
<td>-48</td>
</tr>
<tr>
<td>-822</td>
<td>690</td>
<td>3669</td>
<td>111</td>
<td>17835</td>
<td>-21615</td>
</tr>
<tr>
<td>1590</td>
<td>-1938</td>
<td>7062</td>
<td>-7050</td>
<td>1875</td>
<td>-2505</td>
</tr>
<tr>
<td>-4155</td>
<td>7935</td>
<td>804</td>
<td>-960</td>
<td>-1698</td>
<td>1830</td>
</tr>
<tr>
<td>0</td>
<td>2058</td>
<td>-2058</td>
<td>0</td>
<td>198</td>
<td>-618</td>
</tr>
<tr>
<td>264</td>
<td>-264</td>
<td>-198</td>
<td>-264</td>
<td>618</td>
<td>264</td>
</tr>
<tr>
<td>0</td>
<td>66</td>
<td>-66</td>
<td>0</td>
<td>4242</td>
<td>-6300</td>
</tr>
<tr>
<td>2520</td>
<td>-462</td>
<td>1482</td>
<td>198</td>
<td>576</td>
<td>264</td>
</tr>
<tr>
<td>1272</td>
<td>-810</td>
<td>-1020</td>
<td>138</td>
<td>402</td>
<td>0</td>
</tr>
<tr>
<td>-252</td>
<td>-66</td>
<td>-4242</td>
<td>-2520</td>
<td>6300</td>
<td>462</td>
</tr>
<tr>
<td>-1272</td>
<td>1020</td>
<td>810</td>
<td>-138</td>
<td>-1482</td>
<td>-576</td>
</tr>
<tr>
<td>-198</td>
<td>-264</td>
<td>-402</td>
<td>252</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>---</td>
<td>----</td>
</tr>
<tr>
<td>0</td>
<td>6762</td>
<td>-6762</td>
<td>0</td>
<td>180</td>
<td>-1272</td>
</tr>
<tr>
<td>-2238</td>
<td>810</td>
<td>-180</td>
<td>2238</td>
<td>1272</td>
<td>-810</td>
</tr>
<tr>
<td>0</td>
<td>-402</td>
<td>402</td>
<td>0</td>
<td>14055</td>
<td>7449</td>
</tr>
<tr>
<td>-17835</td>
<td>-3669</td>
<td>375</td>
<td>-1245</td>
<td>-4155</td>
<td>1875</td>
</tr>
<tr>
<td>-7050</td>
<td>-1938</td>
<td>7062</td>
<td>1590</td>
<td>-1830</td>
<td>960</td>
</tr>
<tr>
<td>1698</td>
<td>-804</td>
<td>-10653</td>
<td>-10851</td>
<td>14433</td>
<td>7071</td>
</tr>
<tr>
<td>-375</td>
<td>2379</td>
<td>4155</td>
<td>-3009</td>
<td>7806</td>
<td>1182</td>
</tr>
<tr>
<td>-6306</td>
<td>-2346</td>
<td>1830</td>
<td>-708</td>
<td>-1698</td>
<td>1056</td>
</tr>
<tr>
<td>1695</td>
<td>-1149</td>
<td>2085</td>
<td>-2631</td>
<td>2145</td>
<td>-645</td>
</tr>
<tr>
<td>1635</td>
<td>15</td>
<td>750</td>
<td>-582</td>
<td>-762</td>
<td>930</td>
</tr>
<tr>
<td>822</td>
<td>-204</td>
<td>-690</td>
<td>48</td>
<td>-5097</td>
<td>4551</td>
</tr>
<tr>
<td>1317</td>
<td>-771</td>
<td>-2145</td>
<td>-489</td>
<td>-1635</td>
<td>1119</td>
</tr>
<tr>
<td>-1506</td>
<td>1338</td>
<td>6</td>
<td>-174</td>
<td>-822</td>
<td>-48</td>
</tr>
<tr>
<td>690</td>
<td>-300</td>
<td>10920</td>
<td>3780</td>
<td>-14700</td>
<td>0</td>
</tr>
<tr>
<td>5040</td>
<td>-840</td>
<td>0</td>
<td>0</td>
<td>-10500</td>
<td>0</td>
</tr>
<tr>
<td>7140</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9114</td>
<td>7854</td>
<td>-8484</td>
<td>-8484</td>
<td>1606</td>
<td>-3286</td>
</tr>
<tr>
<td>-2446</td>
<td>2446</td>
<td>-4998</td>
<td>-4998</td>
<td>3822</td>
<td>3822</td>
</tr>
<tr>
<td>-1292</td>
<td>1292</td>
<td>1060</td>
<td>-1060</td>
<td>9240</td>
<td>-1890</td>
</tr>
<tr>
<td>-5460</td>
<td>-1890</td>
<td>-2520</td>
<td>420</td>
<td>-2520</td>
<td>420</td>
</tr>
<tr>
<td>2436</td>
<td>-756</td>
<td>924</td>
<td>756</td>
<td>-1008</td>
<td>168</td>
</tr>
<tr>
<td>1008</td>
<td>-168</td>
<td>2226</td>
<td>2856</td>
<td>-2856</td>
<td>-2226</td>
</tr>
<tr>
<td>914</td>
<td>-74</td>
<td>-74</td>
<td>914</td>
<td>462</td>
<td>714</td>
</tr>
<tr>
<td>714</td>
<td>462</td>
<td>284</td>
<td>52</td>
<td>-52</td>
<td>-284</td>
</tr>
<tr>
<td>11100</td>
<td>-4338</td>
<td>-4338</td>
<td>-2424</td>
<td>1830</td>
<td>1008</td>
</tr>
<tr>
<td>-1368</td>
<td>1050</td>
<td>1830</td>
<td>-1368</td>
<td>1008</td>
<td>1050</td>
</tr>
<tr>
<td>120</td>
<td>432</td>
<td>432</td>
<td>-432</td>
<td>-3288</td>
<td>-3474</td>
</tr>
<tr>
<td>5346</td>
<td>1416</td>
<td>-1830</td>
<td>1596</td>
<td>1368</td>
<td>-714</td>
</tr>
<tr>
<td>690</td>
<td>-1152</td>
<td>-2016</td>
<td>-42</td>
<td>-120</td>
<td>408</td>
</tr>
<tr>
<td>-432</td>
<td>96</td>
<td>-3474</td>
<td>1416</td>
<td>-3288</td>
<td>5346</td>
</tr>
<tr>
<td>-1152</td>
<td>-42</td>
<td>690</td>
<td>-2016</td>
<td>-1596</td>
<td>714</td>
</tr>
<tr>
<td>1830</td>
<td>-1368</td>
<td>-408</td>
<td>-96</td>
<td>120</td>
<td>432</td>
</tr>
<tr>
<td>2466</td>
<td>-408</td>
<td>-4524</td>
<td>2466</td>
<td>1152</td>
<td>-294</td>
</tr>
<tr>
<td>-690</td>
<td>-588</td>
<td>588</td>
<td>294</td>
<td>690</td>
<td>-1152</td>
</tr>
<tr>
<td>408</td>
<td>-240</td>
<td>-120</td>
<td>408</td>
<td>8280</td>
<td>-3030</td>
</tr>
<tr>
<td>-4668</td>
<td>-582</td>
<td>-1800</td>
<td>972</td>
<td>-216</td>
<td>204</td>
</tr>
<tr>
<td>-360</td>
<td>-690</td>
<td>2010</td>
<td>300</td>
<td>-360</td>
<td>228</td>
</tr>
<tr>
<td>120</td>
<td>-104</td>
<td>1308</td>
<td>4446</td>
<td>-3240</td>
<td>-2514</td>
</tr>
<tr>
<td>772</td>
<td>-844</td>
<td>-856</td>
<td>760</td>
<td>-654</td>
<td>108</td>
</tr>
<tr>
<td>1296</td>
<td>930</td>
<td>-56</td>
<td>8</td>
<td>312</td>
<td>-268</td>
</tr>
<tr>
<td>-4908</td>
<td>1338</td>
<td>1296</td>
<td>2274</td>
<td>1224</td>
<td>-372</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>792</td>
<td>-804</td>
<td>-1182</td>
<td>552</td>
<td>-360</td>
<td>-690</td>
</tr>
<tr>
<td>456</td>
<td>-160</td>
<td>-360</td>
<td>228</td>
<td>-1044</td>
<td>-2022</td>
</tr>
<tr>
<td>2976</td>
<td>90</td>
<td>-488</td>
<td>248</td>
<td>572</td>
<td>-164</td>
</tr>
<tr>
<td>-132</td>
<td>-582</td>
<td>-654</td>
<td>108</td>
<td>-136</td>
<td>68</td>
</tr>
<tr>
<td>-56</td>
<td>8</td>
<td>-10851</td>
<td>-10653</td>
<td>7071</td>
<td>14433</td>
</tr>
<tr>
<td>-2379</td>
<td>375</td>
<td>3009</td>
<td>-4155</td>
<td>1182</td>
<td>7806</td>
</tr>
<tr>
<td>-2346</td>
<td>-6306</td>
<td>708</td>
<td>-1830</td>
<td>-1056</td>
<td>1698</td>
</tr>
<tr>
<td>7449</td>
<td>14055</td>
<td>-3669</td>
<td>-17835</td>
<td>1245</td>
<td>-375</td>
</tr>
<tr>
<td>-1875</td>
<td>4155</td>
<td>-1938</td>
<td>-7050</td>
<td>1590</td>
<td>7062</td>
</tr>
<tr>
<td>-960</td>
<td>1830</td>
<td>804</td>
<td>-1698</td>
<td>4551</td>
<td>-5097</td>
</tr>
<tr>
<td>-771</td>
<td>1317</td>
<td>489</td>
<td>2145</td>
<td>-1119</td>
<td>1635</td>
</tr>
<tr>
<td>1338</td>
<td>-1506</td>
<td>-174</td>
<td>6</td>
<td>48</td>
<td>822</td>
</tr>
<tr>
<td>300</td>
<td>-690</td>
<td>-1149</td>
<td>1695</td>
<td>-2631</td>
<td>2085</td>
</tr>
<tr>
<td>645</td>
<td>-2145</td>
<td>-15</td>
<td>-1635</td>
<td>-582</td>
<td>750</td>
</tr>
<tr>
<td>930</td>
<td>-762</td>
<td>204</td>
<td>-822</td>
<td>-48</td>
<td>690</td>
</tr>
<tr>
<td>-7854</td>
<td>-9114</td>
<td>8484</td>
<td>8484</td>
<td>-3286</td>
<td>1606</td>
</tr>
<tr>
<td>2446</td>
<td>-2446</td>
<td>4998</td>
<td>4998</td>
<td>-3822</td>
<td>-3822</td>
</tr>
<tr>
<td>1292</td>
<td>-1292</td>
<td>-1060</td>
<td>1060</td>
<td>-3780</td>
<td>-10920</td>
</tr>
<tr>
<td>0</td>
<td>14700</td>
<td>-840</td>
<td>5040</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10500</td>
<td>0</td>
<td>-7140</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2856</td>
<td>-2226</td>
<td>2226</td>
<td>2856</td>
</tr>
<tr>
<td>-74</td>
<td>914</td>
<td>914</td>
<td>-74</td>
<td>-714</td>
<td>462</td>
</tr>
<tr>
<td>-462</td>
<td>-714</td>
<td>52</td>
<td>284</td>
<td>-284</td>
<td>-52</td>
</tr>
<tr>
<td>1890</td>
<td>-9240</td>
<td>1890</td>
<td>5460</td>
<td>420</td>
<td>-2520</td>
</tr>
<tr>
<td>420</td>
<td>-2520</td>
<td>756</td>
<td>-2436</td>
<td>-756</td>
<td>-924</td>
</tr>
<tr>
<td>168</td>
<td>-1008</td>
<td>-168</td>
<td>1008</td>
<td>-3474</td>
<td>-3288</td>
</tr>
<tr>
<td>1416</td>
<td>5346</td>
<td>-1596</td>
<td>1830</td>
<td>714</td>
<td>-1368</td>
</tr>
<tr>
<td>-1152</td>
<td>690</td>
<td>-42</td>
<td>-2016</td>
<td>-408</td>
<td>120</td>
</tr>
<tr>
<td>-96</td>
<td>432</td>
<td>-4338</td>
<td>11100</td>
<td>-2424</td>
<td>-4338</td>
</tr>
<tr>
<td>-1008</td>
<td>-1830</td>
<td>-1050</td>
<td>1368</td>
<td>-1368</td>
<td>1830</td>
</tr>
<tr>
<td>1050</td>
<td>1008</td>
<td>-432</td>
<td>-120</td>
<td>432</td>
<td>-432</td>
</tr>
<tr>
<td>-408</td>
<td>2466</td>
<td>2466</td>
<td>-4524</td>
<td>294</td>
<td>-1152</td>
</tr>
<tr>
<td>588</td>
<td>690</td>
<td>294</td>
<td>588</td>
<td>-1152</td>
<td>690</td>
</tr>
<tr>
<td>240</td>
<td>-408</td>
<td>-408</td>
<td>120</td>
<td>1416</td>
<td>-3474</td>
</tr>
<tr>
<td>5346</td>
<td>-3288</td>
<td>42</td>
<td>1152</td>
<td>2016</td>
<td>-690</td>
</tr>
<tr>
<td>714</td>
<td>-1596</td>
<td>-1368</td>
<td>1830</td>
<td>96</td>
<td>408</td>
</tr>
<tr>
<td>-432</td>
<td>-120</td>
<td>-4446</td>
<td>-1308</td>
<td>2514</td>
<td>3240</td>
</tr>
<tr>
<td>-844</td>
<td>772</td>
<td>760</td>
<td>-856</td>
<td>-108</td>
<td>654</td>
</tr>
<tr>
<td>-930</td>
<td>-1296</td>
<td>8</td>
<td>-56</td>
<td>-268</td>
<td>312</td>
</tr>
<tr>
<td>3030</td>
<td>-8280</td>
<td>582</td>
<td>4668</td>
<td>972</td>
<td>-1800</td>
</tr>
<tr>
<td>204</td>
<td>-216</td>
<td>690</td>
<td>360</td>
<td>-300</td>
<td>-2010</td>
</tr>
<tr>
<td>228</td>
<td>-360</td>
<td>-104</td>
<td>120</td>
<td>2022</td>
<td>1044</td>
</tr>
<tr>
<td>-90</td>
<td>-2976</td>
<td>248</td>
<td>-488</td>
<td>-164</td>
<td>572</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>582</td>
<td>132</td>
<td>-108</td>
<td>654</td>
<td>68</td>
<td>-136</td>
</tr>
<tr>
<td>8</td>
<td>-56</td>
<td>-1338</td>
<td>4908</td>
<td>-2274</td>
<td>-1296</td>
</tr>
<tr>
<td>-372</td>
<td>1224</td>
<td>-804</td>
<td>792</td>
<td>-552</td>
<td>1182</td>
</tr>
<tr>
<td>690</td>
<td>360</td>
<td>-160</td>
<td>456</td>
<td>228</td>
<td>-360</td>
</tr>
<tr>
<td>-2085</td>
<td>2631</td>
<td>-1695</td>
<td>1149</td>
<td>-1635</td>
<td>-15</td>
</tr>
<tr>
<td>-2145</td>
<td>645</td>
<td>-762</td>
<td>930</td>
<td>750</td>
<td>-582</td>
</tr>
<tr>
<td>-690</td>
<td>48</td>
<td>822</td>
<td>-204</td>
<td>-1317</td>
<td>771</td>
</tr>
<tr>
<td>5097</td>
<td>-4551</td>
<td>1635</td>
<td>-1119</td>
<td>2145</td>
<td>489</td>
</tr>
<tr>
<td>6</td>
<td>-174</td>
<td>-1506</td>
<td>1338</td>
<td>690</td>
<td>-300</td>
</tr>
<tr>
<td>-822</td>
<td>-48</td>
<td>17835</td>
<td>3669</td>
<td>-14055</td>
<td>-7449</td>
</tr>
<tr>
<td>4155</td>
<td>-1875</td>
<td>-375</td>
<td>1245</td>
<td>7062</td>
<td>1590</td>
</tr>
<tr>
<td>-7050</td>
<td>-1938</td>
<td>1698</td>
<td>-804</td>
<td>-1830</td>
<td>960</td>
</tr>
<tr>
<td>-14433</td>
<td>-7071</td>
<td>10653</td>
<td>10851</td>
<td>-4155</td>
<td>3009</td>
</tr>
<tr>
<td>375</td>
<td>-2379</td>
<td>-6306</td>
<td>-2346</td>
<td>7806</td>
<td>1182</td>
</tr>
<tr>
<td>-1698</td>
<td>1056</td>
<td>1830</td>
<td>-708</td>
<td>5460</td>
<td>1890</td>
</tr>
<tr>
<td>-9240</td>
<td>1890</td>
<td>2520</td>
<td>-420</td>
<td>2520</td>
<td>-420</td>
</tr>
<tr>
<td>924</td>
<td>756</td>
<td>2436</td>
<td>-756</td>
<td>1008</td>
<td>-168</td>
</tr>
<tr>
<td>-1008</td>
<td>168</td>
<td>2856</td>
<td>2226</td>
<td>-2226</td>
<td>-2856</td>
</tr>
<tr>
<td>74</td>
<td>-914</td>
<td>-914</td>
<td>74</td>
<td>714</td>
<td>462</td>
</tr>
<tr>
<td>462</td>
<td>714</td>
<td>-52</td>
<td>-284</td>
<td>284</td>
<td>52</td>
</tr>
<tr>
<td>14700</td>
<td>0</td>
<td>-10920</td>
<td>-3780</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-5040</td>
<td>840</td>
<td>7140</td>
<td>0</td>
<td>-10500</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8484</td>
<td>8484</td>
</tr>
<tr>
<td>-9114</td>
<td>-7854</td>
<td>2446</td>
<td>-2446</td>
<td>-1606</td>
<td>3286</td>
</tr>
<tr>
<td>3822</td>
<td>3822</td>
<td>-4998</td>
<td>-4998</td>
<td>1060</td>
<td>-1060</td>
</tr>
<tr>
<td>-1292</td>
<td>1292</td>
<td>-3288</td>
<td>5346</td>
<td>-3474</td>
<td>1416</td>
</tr>
<tr>
<td>690</td>
<td>-2016</td>
<td>-1152</td>
<td>-42</td>
<td>-1830</td>
<td>1368</td>
</tr>
<tr>
<td>1596</td>
<td>-714</td>
<td>-120</td>
<td>-432</td>
<td>408</td>
<td>96</td>
</tr>
<tr>
<td>-4524</td>
<td>2466</td>
<td>2466</td>
<td>-408</td>
<td>-690</td>
<td>-588</td>
</tr>
<tr>
<td>1152</td>
<td>-294</td>
<td>-690</td>
<td>1152</td>
<td>-588</td>
<td>-294</td>
</tr>
<tr>
<td>120</td>
<td>-408</td>
<td>-408</td>
<td>240</td>
<td>-4338</td>
<td>-2424</td>
</tr>
<tr>
<td>11100</td>
<td>-4338</td>
<td>-1368</td>
<td>1050</td>
<td>1830</td>
<td>1008</td>
</tr>
<tr>
<td>-1008</td>
<td>-1050</td>
<td>-1830</td>
<td>1368</td>
<td>-432</td>
<td>432</td>
</tr>
<tr>
<td>-120</td>
<td>-432</td>
<td>5346</td>
<td>1416</td>
<td>-3288</td>
<td>-3474</td>
</tr>
<tr>
<td>1368</td>
<td>-714</td>
<td>-1830</td>
<td>1596</td>
<td>2016</td>
<td>42</td>
</tr>
<tr>
<td>-690</td>
<td>1152</td>
<td>432</td>
<td>-96</td>
<td>120</td>
<td>-408</td>
</tr>
<tr>
<td>1296</td>
<td>2274</td>
<td>-4908</td>
<td>1338</td>
<td>792</td>
<td>-804</td>
</tr>
<tr>
<td>1224</td>
<td>-372</td>
<td>360</td>
<td>690</td>
<td>1182</td>
<td>-552</td>
</tr>
<tr>
<td>360</td>
<td>-228</td>
<td>-456</td>
<td>160</td>
<td>2976</td>
<td>90</td>
</tr>
<tr>
<td>-1044</td>
<td>-2022</td>
<td>572</td>
<td>-164</td>
<td>-488</td>
<td>248</td>
</tr>
<tr>
<td>654</td>
<td>-108</td>
<td>132</td>
<td>582</td>
<td>56</td>
<td>-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>136</td>
<td>-68</td>
<td>-4668</td>
<td>-582</td>
<td>8280</td>
<td>-3030</td>
</tr>
<tr>
<td>-216</td>
<td>204</td>
<td>-1800</td>
<td>972</td>
<td>-2010</td>
<td>-300</td>
</tr>
<tr>
<td>360</td>
<td>690</td>
<td>-120</td>
<td>104</td>
<td>360</td>
<td>-228</td>
</tr>
<tr>
<td>-3240</td>
<td>-2514</td>
<td>1308</td>
<td>4446</td>
<td>-856</td>
<td>760</td>
</tr>
<tr>
<td>772</td>
<td>-844</td>
<td>-1296</td>
<td>-930</td>
<td>654</td>
<td>-108</td>
</tr>
<tr>
<td>-312</td>
<td>268</td>
<td>56</td>
<td>-8</td>
<td>771</td>
<td>-1317</td>
</tr>
<tr>
<td>-4551</td>
<td>5097</td>
<td>1119</td>
<td>-1635</td>
<td>-489</td>
<td>-2145</td>
</tr>
<tr>
<td>-174</td>
<td>6</td>
<td>1338</td>
<td>-1506</td>
<td>300</td>
<td>-690</td>
</tr>
<tr>
<td>48</td>
<td>822</td>
<td>2631</td>
<td>-2085</td>
<td>1149</td>
<td>-1695</td>
</tr>
<tr>
<td>15</td>
<td>1635</td>
<td>-645</td>
<td>2145</td>
<td>930</td>
<td>-762</td>
</tr>
<tr>
<td>-582</td>
<td>750</td>
<td>-48</td>
<td>690</td>
<td>204</td>
<td>-822</td>
</tr>
<tr>
<td>-7071</td>
<td>-14433</td>
<td>10851</td>
<td>10653</td>
<td>-3009</td>
<td>4155</td>
</tr>
<tr>
<td>2379</td>
<td>-375</td>
<td>-2346</td>
<td>-6306</td>
<td>1182</td>
<td>7806</td>
</tr>
<tr>
<td>-1056</td>
<td>1698</td>
<td>708</td>
<td>-1830</td>
<td>3669</td>
<td>17835</td>
</tr>
<tr>
<td>-7449</td>
<td>-14055</td>
<td>1875</td>
<td>-4155</td>
<td>-1245</td>
<td>375</td>
</tr>
<tr>
<td>1590</td>
<td>7062</td>
<td>-1938</td>
<td>-7050</td>
<td>804</td>
<td>-1698</td>
</tr>
<tr>
<td>-960</td>
<td>1830</td>
<td>-2226</td>
<td>-2856</td>
<td>2856</td>
<td>2226</td>
</tr>
<tr>
<td>-914</td>
<td>74</td>
<td>74</td>
<td>-914</td>
<td>-462</td>
<td>-714</td>
</tr>
<tr>
<td>-714</td>
<td>-462</td>
<td>-284</td>
<td>-52</td>
<td>52</td>
<td>284</td>
</tr>
<tr>
<td>-1890</td>
<td>-5460</td>
<td>-1890</td>
<td>9240</td>
<td>-420</td>
<td>2520</td>
</tr>
<tr>
<td>-420</td>
<td>2520</td>
<td>-756</td>
<td>-924</td>
<td>756</td>
<td>-2436</td>
</tr>
<tr>
<td>-168</td>
<td>1008</td>
<td>168</td>
<td>-1008</td>
<td>-8484</td>
<td>-8484</td>
</tr>
<tr>
<td>7854</td>
<td>9114</td>
<td>-2446</td>
<td>2446</td>
<td>3286</td>
<td>-1606</td>
</tr>
<tr>
<td>-3822</td>
<td>-3822</td>
<td>4998</td>
<td>4998</td>
<td>-1060</td>
<td>1060</td>
</tr>
<tr>
<td>1292</td>
<td>-1292</td>
<td>0</td>
<td>-14700</td>
<td>3780</td>
<td>10920</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>840</td>
<td>-5040</td>
<td>0</td>
<td>-7140</td>
</tr>
<tr>
<td>0</td>
<td>10500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2466</td>
<td>-4524</td>
<td>-408</td>
<td>2466</td>
<td>588</td>
<td>690</td>
</tr>
<tr>
<td>294</td>
<td>-1152</td>
<td>1152</td>
<td>-690</td>
<td>-294</td>
<td>-588</td>
</tr>
<tr>
<td>408</td>
<td>-120</td>
<td>-240</td>
<td>408</td>
<td>5346</td>
<td>-3288</td>
</tr>
<tr>
<td>1416</td>
<td>-3474</td>
<td>2016</td>
<td>-690</td>
<td>42</td>
<td>1152</td>
</tr>
<tr>
<td>1368</td>
<td>-1830</td>
<td>-714</td>
<td>1596</td>
<td>432</td>
<td>120</td>
</tr>
<tr>
<td>-96</td>
<td>-408</td>
<td>1416</td>
<td>5346</td>
<td>-3474</td>
<td>-3288</td>
</tr>
<tr>
<td>714</td>
<td>-1368</td>
<td>-1596</td>
<td>1830</td>
<td>42</td>
<td>2016</td>
</tr>
<tr>
<td>1152</td>
<td>-690</td>
<td>96</td>
<td>-432</td>
<td>408</td>
<td>-120</td>
</tr>
<tr>
<td>-2424</td>
<td>-4338</td>
<td>-4338</td>
<td>11100</td>
<td>-1050</td>
<td>1368</td>
</tr>
<tr>
<td>-1008</td>
<td>-1830</td>
<td>-1050</td>
<td>-1008</td>
<td>1368</td>
<td>-1830</td>
</tr>
<tr>
<td>-432</td>
<td>432</td>
<td>432</td>
<td>120</td>
<td>-90</td>
<td>-2976</td>
</tr>
<tr>
<td>2022</td>
<td>1044</td>
<td>-164</td>
<td>572</td>
<td>248</td>
<td>-488</td>
</tr>
<tr>
<td>108</td>
<td>-654</td>
<td>-582</td>
<td>-132</td>
<td>-8</td>
<td>56</td>
</tr>
<tr>
<td>-68</td>
<td>136</td>
<td>-2274</td>
<td>-1296</td>
<td>-1338</td>
<td>4908</td>
</tr>
<tr>
<td>-804</td>
<td>792</td>
<td>-372</td>
<td>1224</td>
<td>-690</td>
<td>-360</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>-552</td>
<td>690</td>
<td>-1224</td>
<td>-792</td>
<td>372</td>
<td>804</td>
</tr>
<tr>
<td>-456</td>
<td>360</td>
<td>160</td>
<td>-228</td>
<td>-1308</td>
<td>3240</td>
</tr>
<tr>
<td>-4446</td>
<td>2514</td>
<td>654</td>
<td>-1296</td>
<td>-108</td>
<td>-930</td>
</tr>
<tr>
<td>-772</td>
<td>856</td>
<td>844</td>
<td>-760</td>
<td>56</td>
<td>-312</td>
</tr>
<tr>
<td>-8</td>
<td>268</td>
<td>1044</td>
<td>-2976</td>
<td>2022</td>
<td>-90</td>
</tr>
<tr>
<td>132</td>
<td>654</td>
<td>582</td>
<td>-108</td>
<td>488</td>
<td>-572</td>
</tr>
<tr>
<td>-248</td>
<td>164</td>
<td>136</td>
<td>56</td>
<td>-68</td>
<td>-8</td>
</tr>
<tr>
<td>2085</td>
<td>1695</td>
<td>-2631</td>
<td>-1149</td>
<td>762</td>
<td>-750</td>
</tr>
<tr>
<td>-930</td>
<td>582</td>
<td>1635</td>
<td>2145</td>
<td>15</td>
<td>-645</td>
</tr>
<tr>
<td>690</td>
<td>-822</td>
<td>-48</td>
<td>204</td>
<td>-17835</td>
<td>14055</td>
</tr>
<tr>
<td>-3669</td>
<td>7449</td>
<td>-7062</td>
<td>7050</td>
<td>-1590</td>
<td>1938</td>
</tr>
<tr>
<td>-4155</td>
<td>375</td>
<td>1875</td>
<td>-1245</td>
<td>-1698</td>
<td>1830</td>
</tr>
<tr>
<td>804</td>
<td>-960</td>
<td>1317</td>
<td>-5097</td>
<td>-771</td>
<td>4551</td>
</tr>
<tr>
<td>-6</td>
<td>1506</td>
<td>174</td>
<td>-1338</td>
<td>-1635</td>
<td>-2145</td>
</tr>
<tr>
<td>1119</td>
<td>-489</td>
<td>-690</td>
<td>822</td>
<td>300</td>
<td>48</td>
</tr>
<tr>
<td>14433</td>
<td>-10653</td>
<td>7071</td>
<td>-10851</td>
<td>6306</td>
<td>-7806</td>
</tr>
<tr>
<td>2346</td>
<td>-1182</td>
<td>4155</td>
<td>-375</td>
<td>-3009</td>
<td>2379</td>
</tr>
<tr>
<td>1698</td>
<td>-1830</td>
<td>-1056</td>
<td>708</td>
<td>3288</td>
<td>3474</td>
</tr>
<tr>
<td>-5346</td>
<td>-1416</td>
<td>1830</td>
<td>-1596</td>
<td>-1368</td>
<td>714</td>
</tr>
<tr>
<td>-690</td>
<td>1152</td>
<td>2016</td>
<td>42</td>
<td>120</td>
<td>-408</td>
</tr>
<tr>
<td>432</td>
<td>-96</td>
<td>4338</td>
<td>-11100</td>
<td>2424</td>
<td>4338</td>
</tr>
<tr>
<td>1008</td>
<td>1830</td>
<td>1050</td>
<td>-1368</td>
<td>1368</td>
<td>-1830</td>
</tr>
<tr>
<td>-1050</td>
<td>-1008</td>
<td>432</td>
<td>120</td>
<td>-432</td>
<td>432</td>
</tr>
<tr>
<td>4524</td>
<td>-2466</td>
<td>-2466</td>
<td>408</td>
<td>690</td>
<td>588</td>
</tr>
<tr>
<td>-1152</td>
<td>294</td>
<td>690</td>
<td>-1152</td>
<td>588</td>
<td>294</td>
</tr>
<tr>
<td>-120</td>
<td>408</td>
<td>408</td>
<td>-240</td>
<td>-5346</td>
<td>3288</td>
</tr>
<tr>
<td>-1416</td>
<td>3474</td>
<td>-2016</td>
<td>690</td>
<td>-42</td>
<td>-1152</td>
</tr>
<tr>
<td>-1368</td>
<td>1830</td>
<td>714</td>
<td>-1596</td>
<td>-432</td>
<td>-120</td>
</tr>
<tr>
<td>96</td>
<td>408</td>
<td>-5460</td>
<td>9240</td>
<td>-1890</td>
<td>-1890</td>
</tr>
<tr>
<td>-924</td>
<td>-2436</td>
<td>-756</td>
<td>756</td>
<td>-2520</td>
<td>-2520</td>
</tr>
<tr>
<td>420</td>
<td>420</td>
<td>-1008</td>
<td>1008</td>
<td>168</td>
<td>-168</td>
</tr>
<tr>
<td>-14700</td>
<td>10920</td>
<td>0</td>
<td>3780</td>
<td>-7140</td>
<td>10500</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5040</td>
<td>0</td>
<td>-840</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2856</td>
<td>2226</td>
</tr>
<tr>
<td>-2226</td>
<td>2856</td>
<td>-714</td>
<td>-462</td>
<td>-462</td>
<td>-714</td>
</tr>
<tr>
<td>-74</td>
<td>914</td>
<td>914</td>
<td>-74</td>
<td>52</td>
<td>-284</td>
</tr>
<tr>
<td>284</td>
<td>-52</td>
<td>-8484</td>
<td>9114</td>
<td>-8484</td>
<td>7854</td>
</tr>
<tr>
<td>-3822</td>
<td>4998</td>
<td>-3822</td>
<td>4998</td>
<td>-2446</td>
<td>1606</td>
</tr>
<tr>
<td>2446</td>
<td>-3286</td>
<td>-1060</td>
<td>1292</td>
<td>1060</td>
<td>-1292</td>
</tr>
<tr>
<td>-1296</td>
<td>4908</td>
<td>-2274</td>
<td>-1338</td>
<td>-360</td>
<td>-1182</td>
</tr>
<tr>
<td>-690</td>
<td>552</td>
<td>-792</td>
<td>-1224</td>
<td>804</td>
<td>372</td>
</tr>
<tr>
<td>-360</td>
<td>456</td>
<td>228</td>
<td>-160</td>
<td>4668</td>
<td>-8280</td>
</tr>
<tr>
<td></td>
<td>582</td>
<td>3030</td>
<td>2010</td>
<td>-360</td>
<td>300</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>216</td>
<td>1800</td>
<td>-204</td>
<td>-972</td>
<td>120</td>
<td>-360</td>
</tr>
<tr>
<td>-104</td>
<td>228</td>
<td>-2976</td>
<td>1044</td>
<td>-90</td>
<td>2022</td>
</tr>
<tr>
<td>-654</td>
<td>-132</td>
<td>108</td>
<td>-582</td>
<td>-572</td>
<td>488</td>
</tr>
<tr>
<td>164</td>
<td>-248</td>
<td>-56</td>
<td>-136</td>
<td>8</td>
<td>68</td>
</tr>
<tr>
<td>3240</td>
<td>-1308</td>
<td>2514</td>
<td>-4446</td>
<td>1296</td>
<td>-654</td>
</tr>
<tr>
<td>930</td>
<td>108</td>
<td>856</td>
<td>-772</td>
<td>-760</td>
<td>844</td>
</tr>
<tr>
<td>312</td>
<td>-56</td>
<td>-268</td>
<td>8</td>
<td>10851</td>
<td>-7071</td>
</tr>
<tr>
<td>10653</td>
<td>-14433</td>
<td>-1182</td>
<td>2346</td>
<td>-7806</td>
<td>6306</td>
</tr>
<tr>
<td>2379</td>
<td>-3009</td>
<td>-375</td>
<td>4155</td>
<td>-708</td>
<td>1056</td>
</tr>
<tr>
<td>1830</td>
<td>-1698</td>
<td>-4551</td>
<td>771</td>
<td>5097</td>
<td>-1317</td>
</tr>
<tr>
<td>-1338</td>
<td>174</td>
<td>1506</td>
<td>-6</td>
<td>-489</td>
<td>1119</td>
</tr>
<tr>
<td>-2145</td>
<td>-1635</td>
<td>-48</td>
<td>-300</td>
<td>-822</td>
<td>690</td>
</tr>
<tr>
<td>-7449</td>
<td>3669</td>
<td>-14055</td>
<td>17835</td>
<td>1938</td>
<td>-1590</td>
</tr>
<tr>
<td>7050</td>
<td>-7062</td>
<td>-1245</td>
<td>1875</td>
<td>375</td>
<td>-4155</td>
</tr>
<tr>
<td>960</td>
<td>-804</td>
<td>-1830</td>
<td>1698</td>
<td>1149</td>
<td>2631</td>
</tr>
<tr>
<td>-1695</td>
<td>-2085</td>
<td>582</td>
<td>-930</td>
<td>-750</td>
<td>762</td>
</tr>
<tr>
<td>-645</td>
<td>15</td>
<td>2145</td>
<td>1635</td>
<td>-204</td>
<td>48</td>
</tr>
<tr>
<td>822</td>
<td>-690</td>
<td>3474</td>
<td>-1416</td>
<td>3288</td>
<td>-5346</td>
</tr>
<tr>
<td>1152</td>
<td>42</td>
<td>-690</td>
<td>2016</td>
<td>1596</td>
<td>714</td>
</tr>
<tr>
<td>-1830</td>
<td>1368</td>
<td>408</td>
<td>96</td>
<td>-120</td>
<td>432</td>
</tr>
<tr>
<td>408</td>
<td>-2466</td>
<td>-2466</td>
<td>4524</td>
<td>-294</td>
<td>1152</td>
</tr>
<tr>
<td>-588</td>
<td>-690</td>
<td>-294</td>
<td>-588</td>
<td>1152</td>
<td>690</td>
</tr>
<tr>
<td>-240</td>
<td>408</td>
<td>408</td>
<td>2016</td>
<td>1596</td>
<td>714</td>
</tr>
<tr>
<td>-11100</td>
<td>4338</td>
<td>1368</td>
<td>-1050</td>
<td>-1830</td>
<td>-1008</td>
</tr>
<tr>
<td>1008</td>
<td>1050</td>
<td>1830</td>
<td>-1368</td>
<td>432</td>
<td>-432</td>
</tr>
<tr>
<td>120</td>
<td>432</td>
<td>-1416</td>
<td>-5346</td>
<td>3474</td>
<td>3288</td>
</tr>
<tr>
<td>-714</td>
<td>1368</td>
<td>1596</td>
<td>-1830</td>
<td>-42</td>
<td>-2016</td>
</tr>
<tr>
<td>-1152</td>
<td>690</td>
<td>-96</td>
<td>432</td>
<td>-408</td>
<td>120</td>
</tr>
<tr>
<td>7854</td>
<td>-8484</td>
<td>9114</td>
<td>-8484</td>
<td>-4998</td>
<td>3822</td>
</tr>
<tr>
<td>-4998</td>
<td>3822</td>
<td>3286</td>
<td>-2446</td>
<td>-1606</td>
<td>2446</td>
</tr>
<tr>
<td>-1292</td>
<td>1060</td>
<td>1292</td>
<td>-1060</td>
<td>2856</td>
<td>-2226</td>
</tr>
<tr>
<td>2226</td>
<td>-2856</td>
<td>714</td>
<td>462</td>
<td>462</td>
<td>714</td>
</tr>
<tr>
<td>74</td>
<td>-914</td>
<td>-914</td>
<td>74</td>
<td>-52</td>
<td>284</td>
</tr>
<tr>
<td>-284</td>
<td>52</td>
<td>3780</td>
<td>0</td>
<td>10920</td>
<td>-14700</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-10500</td>
<td>7140</td>
<td>840</td>
<td>0</td>
</tr>
<tr>
<td>-5040</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1890</td>
<td>-1890</td>
<td>9240</td>
<td>-5460</td>
<td>-756</td>
<td>756</td>
</tr>
<tr>
<td>2436</td>
<td>924</td>
<td>-420</td>
<td>-420</td>
<td>2520</td>
<td>2520</td>
</tr>
<tr>
<td>-168</td>
<td>168</td>
<td>1008</td>
<td>-1008</td>
<td>4446</td>
<td>-2514</td>
</tr>
<tr>
<td>1308</td>
<td>-3240</td>
<td>108</td>
<td>930</td>
<td>-654</td>
<td>1296</td>
</tr>
<tr>
<td>844</td>
<td>-760</td>
<td>-772</td>
<td>856</td>
<td>-8</td>
<td>268</td>
</tr>
<tr>
<td></td>
<td>-312</td>
<td>-2022</td>
<td>90</td>
<td>-1044</td>
<td>2976</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
<td>------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>108</td>
<td>8280</td>
<td>250</td>
</tr>
<tr>
<td>56</td>
<td>-132</td>
<td>-654</td>
<td>-248</td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>-360</td>
<td>1800</td>
<td>216</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>-228</td>
<td>1338</td>
<td>2274</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>-4908</td>
<td>-1182</td>
<td>-360</td>
</tr>
<tr>
<td>56</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>56</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-1324</td>
<td>-248</td>
<td>160</td>
<td>-228</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-582</td>
<td>1506</td>
<td>-5097</td>
</tr>
<tr>
<td>174</td>
<td>-1338</td>
<td>-6</td>
<td>-1119</td>
<td>489</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>10</td>
<td>-183</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>-930</td>
<td>-248</td>
<td>690</td>
<td>-822</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td>-1681</td>
<td>14055</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-572</td>
<td>488</td>
<td>8</td>
<td>68</td>
<td>-56</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>-----</td>
<td>---</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>-2514</td>
<td>4446</td>
<td>-3240</td>
<td>1308</td>
<td>-930</td>
<td>-108</td>
</tr>
<tr>
<td>-1296</td>
<td>654</td>
<td>-760</td>
<td>844</td>
<td>856</td>
<td>-772</td>
</tr>
<tr>
<td>-268</td>
<td>8</td>
<td>312</td>
<td>-56</td>
<td>2274</td>
<td>1338</td>
</tr>
<tr>
<td>1296</td>
<td>-4908</td>
<td>690</td>
<td>-552</td>
<td>360</td>
<td>1182</td>
</tr>
<tr>
<td>216</td>
<td>1800</td>
<td>-104</td>
<td>228</td>
<td>120</td>
<td>-360</td>
</tr>
<tr>
<td>0</td>
<td>6762</td>
<td>-6762</td>
<td>0</td>
<td>810</td>
<td>-2238</td>
</tr>
<tr>
<td>-1272</td>
<td>180</td>
<td>-810</td>
<td>1272</td>
<td>2238</td>
<td>-180</td>
</tr>
<tr>
<td>0</td>
<td>-402</td>
<td>402</td>
<td>0</td>
<td>-6300</td>
<td>-462</td>
</tr>
<tr>
<td>4242</td>
<td>2520</td>
<td>-810</td>
<td>138</td>
<td>1272</td>
<td>-1020</td>
</tr>
<tr>
<td>-198</td>
<td>-264</td>
<td>-1482</td>
<td>-576</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>-402</td>
<td>252</td>
<td>6300</td>
<td>-4242</td>
<td>462</td>
<td>-2520</td>
</tr>
<tr>
<td>198</td>
<td>1482</td>
<td>264</td>
<td>576</td>
<td>810</td>
<td>-1272</td>
</tr>
<tr>
<td>-138</td>
<td>1020</td>
<td>0</td>
<td>402</td>
<td>-66</td>
<td>-252</td>
</tr>
<tr>
<td>0</td>
<td>-2058</td>
<td>2058</td>
<td>0</td>
<td>-198</td>
<td>618</td>
</tr>
<tr>
<td>-264</td>
<td>264</td>
<td>198</td>
<td>264</td>
<td>-618</td>
<td>-264</td>
</tr>
<tr>
<td>0</td>
<td>-66</td>
<td>66</td>
<td>0</td>
<td>2220</td>
<td>3030</td>
</tr>
<tr>
<td>-5832</td>
<td>582</td>
<td>1800</td>
<td>-972</td>
<td>216</td>
<td>-204</td>
</tr>
<tr>
<td>-1740</td>
<td>690</td>
<td>2610</td>
<td>-300</td>
<td>360</td>
<td>-228</td>
</tr>
<tr>
<td>-120</td>
<td>104</td>
<td>4446</td>
<td>1308</td>
<td>-2514</td>
<td>-3240</td>
</tr>
<tr>
<td>844</td>
<td>-772</td>
<td>-760</td>
<td>856</td>
<td>108</td>
<td>-654</td>
</tr>
<tr>
<td>930</td>
<td>1296</td>
<td>-8</td>
<td>56</td>
<td>268</td>
<td>-312</td>
</tr>
<tr>
<td>5844</td>
<td>-2274</td>
<td>-2232</td>
<td>-1338</td>
<td>-792</td>
<td>804</td>
</tr>
<tr>
<td>-1224</td>
<td>372</td>
<td>1740</td>
<td>-690</td>
<td>78</td>
<td>552</td>
</tr>
<tr>
<td>-360</td>
<td>228</td>
<td>456</td>
<td>-160</td>
<td>90</td>
<td>2976</td>
</tr>
<tr>
<td>-2022</td>
<td>-1044</td>
<td>164</td>
<td>-572</td>
<td>-248</td>
<td>488</td>
</tr>
<tr>
<td>-108</td>
<td>654</td>
<td>582</td>
<td>132</td>
<td>8</td>
<td>-56</td>
</tr>
<tr>
<td>68</td>
<td>-136</td>
<td>-2220</td>
<td>5832</td>
<td>-3030</td>
<td>-582</td>
</tr>
<tr>
<td>1740</td>
<td>-2610</td>
<td>-690</td>
<td>300</td>
<td>-1800</td>
<td>-216</td>
</tr>
<tr>
<td>972</td>
<td>204</td>
<td>-360</td>
<td>120</td>
<td>228</td>
<td>-104</td>
</tr>
<tr>
<td>-5844</td>
<td>2232</td>
<td>2274</td>
<td>1338</td>
<td>-2514</td>
<td>-78</td>
</tr>
<tr>
<td>690</td>
<td>-552</td>
<td>792</td>
<td>1224</td>
<td>-804</td>
<td>-372</td>
</tr>
<tr>
<td>360</td>
<td>-456</td>
<td>-228</td>
<td>160</td>
<td>-4446</td>
<td>2514</td>
</tr>
<tr>
<td>-1308</td>
<td>3240</td>
<td>-108</td>
<td>-930</td>
<td>654</td>
<td>-1296</td>
</tr>
<tr>
<td>-844</td>
<td>760</td>
<td>772</td>
<td>-856</td>
<td>8</td>
<td>-268</td>
</tr>
<tr>
<td>-56</td>
<td>312</td>
<td>-90</td>
<td>2022</td>
<td>-2976</td>
<td>1044</td>
</tr>
<tr>
<td>108</td>
<td>-582</td>
<td>-654</td>
<td>-132</td>
<td>-164</td>
<td>248</td>
</tr>
<tr>
<td>572</td>
<td>-488</td>
<td>-8</td>
<td>-68</td>
<td>56</td>
<td>136</td>
</tr>
<tr>
<td>0</td>
<td>2100</td>
<td>-2100</td>
<td>0</td>
<td>0</td>
<td>-840</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1302</td>
<td>1302</td>
<td>32</td>
<td>808</td>
<td>388</td>
<td>-388</td>
</tr>
<tr>
<td>336</td>
<td>336</td>
<td>-476</td>
<td>-476</td>
<td>144</td>
<td>-144</td>
</tr>
<tr>
<td>-136</td>
<td>136</td>
<td>-2100</td>
<td>0</td>
<td>0</td>
<td>2100</td>
</tr>
<tr>
<td>-840</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-840</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1764</td>
<td>0</td>
<td>0</td>
<td>-1764</td>
<td>396</td>
<td>-88</td>
</tr>
<tr>
<td>-88</td>
<td>396</td>
<td>396</td>
<td>-88</td>
<td>-88</td>
<td>396</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
<td>0</td>
<td>-64</td>
<td>-1302</td>
<td>2982</td>
</tr>
<tr>
<td>-1302</td>
<td>-378</td>
<td>-476</td>
<td>336</td>
<td>-476</td>
<td>336</td>
</tr>
<tr>
<td>-388</td>
<td>808</td>
<td>388</td>
<td>32</td>
<td>-136</td>
<td>144</td>
</tr>
<tr>
<td>136</td>
<td>-144</td>
<td>462</td>
<td>-2520</td>
<td>6300</td>
<td>-4242</td>
</tr>
<tr>
<td>264</td>
<td>576</td>
<td>198</td>
<td>1482</td>
<td>138</td>
<td>-1020</td>
</tr>
<tr>
<td>-810</td>
<td>1272</td>
<td>66</td>
<td>252</td>
<td>0</td>
<td>-402</td>
</tr>
<tr>
<td>2058</td>
<td>0</td>
<td>0</td>
<td>-2058</td>
<td>-264</td>
<td>264</td>
</tr>
<tr>
<td>-198</td>
<td>618</td>
<td>618</td>
<td>264</td>
<td>-198</td>
<td>-264</td>
</tr>
<tr>
<td>-66</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>-6762</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6762</td>
<td>-1272</td>
<td>180</td>
<td>810</td>
<td>-2238</td>
</tr>
<tr>
<td>-2238</td>
<td>180</td>
<td>810</td>
<td>-1272</td>
<td>-402</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>402</td>
<td>4242</td>
<td>2520</td>
<td>-6300</td>
<td>-462</td>
</tr>
<tr>
<td>1272</td>
<td>-1020</td>
<td>-810</td>
<td>138</td>
<td>1482</td>
<td>576</td>
</tr>
<tr>
<td>198</td>
<td>264</td>
<td>402</td>
<td>-252</td>
<td>0</td>
<td>-66</td>
</tr>
<tr>
<td>-2232</td>
<td>-1338</td>
<td>5844</td>
<td>-2274</td>
<td>-1224</td>
<td>372</td>
</tr>
<tr>
<td>-792</td>
<td>804</td>
<td>-78</td>
<td>-552</td>
<td>-1740</td>
<td>690</td>
</tr>
<tr>
<td>-456</td>
<td>160</td>
<td>360</td>
<td>-228</td>
<td>-2022</td>
<td>-1044</td>
</tr>
<tr>
<td>90</td>
<td>2976</td>
<td>-248</td>
<td>488</td>
<td>164</td>
<td>-572</td>
</tr>
<tr>
<td>-582</td>
<td>-132</td>
<td>108</td>
<td>-654</td>
<td>-68</td>
<td>136</td>
</tr>
<tr>
<td>-8</td>
<td>56</td>
<td>-5832</td>
<td>582</td>
<td>2220</td>
<td>3030</td>
</tr>
<tr>
<td>216</td>
<td>-204</td>
<td>1800</td>
<td>-972</td>
<td>-2610</td>
<td>300</td>
</tr>
<tr>
<td>1740</td>
<td>-690</td>
<td>120</td>
<td>-104</td>
<td>-360</td>
<td>228</td>
</tr>
<tr>
<td>-2514</td>
<td>-3240</td>
<td>4446</td>
<td>1308</td>
<td>-760</td>
<td>856</td>
</tr>
<tr>
<td>844</td>
<td>-772</td>
<td>-930</td>
<td>-1296</td>
<td>-108</td>
<td>654</td>
</tr>
<tr>
<td>-268</td>
<td>312</td>
<td>8</td>
<td>-56</td>
<td>1308</td>
<td>-3240</td>
</tr>
<tr>
<td>4446</td>
<td>-2514</td>
<td>-654</td>
<td>1296</td>
<td>108</td>
<td>930</td>
</tr>
<tr>
<td>772</td>
<td>-856</td>
<td>-844</td>
<td>760</td>
<td>-56</td>
<td>312</td>
</tr>
<tr>
<td>8</td>
<td>-268</td>
<td>2976</td>
<td>-1044</td>
<td>90</td>
<td>-2022</td>
</tr>
<tr>
<td>654</td>
<td>132</td>
<td>-108</td>
<td>582</td>
<td>572</td>
<td>-488</td>
</tr>
<tr>
<td>-164</td>
<td>248</td>
<td>56</td>
<td>136</td>
<td>-8</td>
<td>-68</td>
</tr>
<tr>
<td>3030</td>
<td>582</td>
<td>2220</td>
<td>-5832</td>
<td>690</td>
<td>-300</td>
</tr>
<tr>
<td>-1740</td>
<td>2610</td>
<td>972</td>
<td>204</td>
<td>-1800</td>
<td>-216</td>
</tr>
<tr>
<td>228</td>
<td>-104</td>
<td>-360</td>
<td>120</td>
<td>-2274</td>
<td>-1338</td>
</tr>
<tr>
<td>5844</td>
<td>-2232</td>
<td>-690</td>
<td>552</td>
<td>1740</td>
<td>78</td>
</tr>
<tr>
<td>-804</td>
<td>-372</td>
<td>792</td>
<td>1224</td>
<td>-228</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>360</td>
<td>-456</td>
<td>-378</td>
<td>-1302</td>
<td>2982</td>
<td>-1302</td>
</tr>
<tr>
<td>-336</td>
<td>476</td>
<td>-336</td>
<td>476</td>
<td>-32</td>
<td>-388</td>
</tr>
<tr>
<td>-808</td>
<td>388</td>
<td>-144</td>
<td>136</td>
<td>144</td>
<td>-136</td>
</tr>
<tr>
<td>-1764</td>
<td>0</td>
<td>0</td>
<td>1764</td>
<td>-396</td>
<td>88</td>
</tr>
<tr>
<td>88</td>
<td>-396</td>
<td>-396</td>
<td>88</td>
<td>88</td>
<td>-396</td>
</tr>
<tr>
<td>-64</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>2100</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-2100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>840</td>
</tr>
<tr>
<td>840</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1302</td>
<td>1302</td>
<td>-2982</td>
<td>378</td>
</tr>
<tr>
<td>388</td>
<td>-388</td>
<td>-808</td>
<td>-32</td>
<td>476</td>
<td>476</td>
</tr>
<tr>
<td>-336</td>
<td>-336</td>
<td>136</td>
<td>-136</td>
<td>-144</td>
<td>144</td>
</tr>
<tr>
<td>0</td>
<td>2058</td>
<td>-2058</td>
<td>0</td>
<td>-264</td>
<td>264</td>
</tr>
<tr>
<td>-618</td>
<td>198</td>
<td>264</td>
<td>618</td>
<td>-264</td>
<td>-198</td>
</tr>
<tr>
<td>0</td>
<td>66</td>
<td>-66</td>
<td>0</td>
<td>-2520</td>
<td>462</td>
</tr>
<tr>
<td>-4242</td>
<td>6300</td>
<td>-576</td>
<td>-264</td>
<td>-1482</td>
<td>-198</td>
</tr>
<tr>
<td>-1020</td>
<td>138</td>
<td>1272</td>
<td>-810</td>
<td>-252</td>
<td>-66</td>
</tr>
<tr>
<td>402</td>
<td>0</td>
<td>2520</td>
<td>4242</td>
<td>-462</td>
<td>-6300</td>
</tr>
<tr>
<td>1020</td>
<td>-1272</td>
<td>-138</td>
<td>810</td>
<td>576</td>
<td>1482</td>
</tr>
<tr>
<td>264</td>
<td>198</td>
<td>252</td>
<td>-402</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-6762</td>
<td>6762</td>
<td>0</td>
<td>-180</td>
<td>1272</td>
</tr>
<tr>
<td>2238</td>
<td>-810</td>
<td>180</td>
<td>-2238</td>
<td>-1272</td>
<td>810</td>
</tr>
<tr>
<td>0</td>
<td>402</td>
<td>-402</td>
<td>0</td>
<td>1044</td>
<td>2022</td>
</tr>
<tr>
<td>-2976</td>
<td>-90</td>
<td>488</td>
<td>-248</td>
<td>-572</td>
<td>164</td>
</tr>
<tr>
<td>132</td>
<td>582</td>
<td>654</td>
<td>-108</td>
<td>136</td>
<td>-68</td>
</tr>
<tr>
<td>56</td>
<td>-8</td>
<td>1338</td>
<td>2232</td>
<td>2274</td>
<td>-5844</td>
</tr>
<tr>
<td>372</td>
<td>-1224</td>
<td>804</td>
<td>-792</td>
<td>552</td>
<td>78</td>
</tr>
<tr>
<td>-690</td>
<td>1740</td>
<td>160</td>
<td>-456</td>
<td>-228</td>
<td>360</td>
</tr>
<tr>
<td>3240</td>
<td>2514</td>
<td>-1308</td>
<td>-4446</td>
<td>856</td>
<td>-760</td>
</tr>
<tr>
<td>-772</td>
<td>844</td>
<td>1296</td>
<td>930</td>
<td>-654</td>
<td>108</td>
</tr>
<tr>
<td>312</td>
<td>-268</td>
<td>-56</td>
<td>8</td>
<td>-582</td>
<td>5832</td>
</tr>
<tr>
<td>-3030</td>
<td>-2220</td>
<td>-204</td>
<td>216</td>
<td>-972</td>
<td>1800</td>
</tr>
<tr>
<td>-300</td>
<td>2610</td>
<td>690</td>
<td>-1740</td>
<td>-104</td>
<td>120</td>
</tr>
<tr>
<td>228</td>
<td>-360</td>
<td>-1044</td>
<td>2976</td>
<td>-2022</td>
<td>90</td>
</tr>
<tr>
<td>-132</td>
<td>-654</td>
<td>-582</td>
<td>108</td>
<td>-488</td>
<td>572</td>
</tr>
<tr>
<td>248</td>
<td>-164</td>
<td>-136</td>
<td>-56</td>
<td>68</td>
<td>8</td>
</tr>
<tr>
<td>-3240</td>
<td>1308</td>
<td>-2514</td>
<td>4446</td>
<td>-1296</td>
<td>654</td>
</tr>
<tr>
<td>-930</td>
<td>-108</td>
<td>-856</td>
<td>772</td>
<td>760</td>
<td>-844</td>
</tr>
<tr>
<td>-312</td>
<td>56</td>
<td>268</td>
<td>-8</td>
<td>-1338</td>
<td>-2274</td>
</tr>
<tr>
<td>-2232</td>
<td>5844</td>
<td>-552</td>
<td>690</td>
<td>-78</td>
<td>-1740</td>
</tr>
<tr>
<td>-372</td>
<td>-804</td>
<td>1224</td>
<td>792</td>
<td>-160</td>
<td>228</td>
</tr>
<tr>
<td>456</td>
<td>-360</td>
<td>582</td>
<td>3030</td>
<td>-5832</td>
<td>2220</td>
</tr>
<tr>
<td>300</td>
<td>-690</td>
<td>-2610</td>
<td>1740</td>
<td>204</td>
<td>972</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>-216</td>
<td>-1800</td>
<td>104</td>
<td>-228</td>
<td>-120</td>
<td>360</td>
</tr>
<tr>
<td>0</td>
<td>1764</td>
<td>-1764</td>
<td>0</td>
<td>88</td>
<td>-396</td>
</tr>
<tr>
<td>-396</td>
<td>88</td>
<td>-88</td>
<td>396</td>
<td>396</td>
<td>-88</td>
</tr>
<tr>
<td>0</td>
<td>-64</td>
<td>64</td>
<td>0</td>
<td>1302</td>
<td>378</td>
</tr>
<tr>
<td>1302</td>
<td>-2982</td>
<td>476</td>
<td>-336</td>
<td>476</td>
<td>-336</td>
</tr>
<tr>
<td>388</td>
<td>32</td>
<td>-388</td>
<td>808</td>
<td>136</td>
<td>-144</td>
</tr>
<tr>
<td>-136</td>
<td>144</td>
<td>-1302</td>
<td>-1302</td>
<td>-378</td>
<td>2982</td>
</tr>
<tr>
<td>-388</td>
<td>388</td>
<td>-32</td>
<td>-808</td>
<td>-476</td>
<td>-476</td>
</tr>
<tr>
<td>336</td>
<td>336</td>
<td>-136</td>
<td>136</td>
<td>144</td>
<td>-144</td>
</tr>
<tr>
<td>0</td>
<td>-2100</td>
<td>2100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>840</td>
<td>0</td>
<td>0</td>
<td>-840</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
This is the content of ea.dat file

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9288</td>
<td>-4563</td>
<td>-4563</td>
<td>-162</td>
<td>4878</td>
<td>3303</td>
</tr>
<tr>
<td>-153</td>
<td>1422</td>
<td>4878</td>
<td>-153</td>
<td>3303</td>
<td>1422</td>
</tr>
<tr>
<td>2043</td>
<td>468</td>
<td>468</td>
<td>-1107</td>
<td>-4563</td>
<td>9288</td>
</tr>
<tr>
<td>-162</td>
<td>-4563</td>
<td>-3303</td>
<td>-4878</td>
<td>-1422</td>
<td>153</td>
</tr>
<tr>
<td>-153</td>
<td>4878</td>
<td>1422</td>
<td>3303</td>
<td>-468</td>
<td>-2043</td>
</tr>
<tr>
<td>1107</td>
<td>-468</td>
<td>-4563</td>
<td>-162</td>
<td>9288</td>
<td>-4563</td>
</tr>
<tr>
<td>-153</td>
<td>1422</td>
<td>4878</td>
<td>3303</td>
<td>-3303</td>
<td>-1422</td>
</tr>
<tr>
<td>-4878</td>
<td>153</td>
<td>-468</td>
<td>1107</td>
<td>-2043</td>
<td>-468</td>
</tr>
<tr>
<td>-162</td>
<td>-4563</td>
<td>-4563</td>
<td>9288</td>
<td>-1422</td>
<td>153</td>
</tr>
<tr>
<td>-3303</td>
<td>-4878</td>
<td>-1422</td>
<td>-3303</td>
<td>153</td>
<td>-4878</td>
</tr>
<tr>
<td>-1107</td>
<td>468</td>
<td>468</td>
<td>2043</td>
<td>4878</td>
<td>-3303</td>
</tr>
<tr>
<td>-153</td>
<td>-1422</td>
<td>6048</td>
<td>1818</td>
<td>252</td>
<td>1332</td>
</tr>
<tr>
<td>2043</td>
<td>-468</td>
<td>468</td>
<td>1107</td>
<td>1848</td>
<td>348</td>
</tr>
<tr>
<td>-252</td>
<td>-702</td>
<td>3303</td>
<td>-4878</td>
<td>1422</td>
<td>153</td>
</tr>
<tr>
<td>1818</td>
<td>6048</td>
<td>1332</td>
<td>252</td>
<td>468</td>
<td>-2043</td>
</tr>
<tr>
<td>-1107</td>
<td>-468</td>
<td>348</td>
<td>1848</td>
<td>-702</td>
<td>-252</td>
</tr>
<tr>
<td>-153</td>
<td>-1422</td>
<td>4878</td>
<td>-3303</td>
<td>252</td>
<td>1332</td>
</tr>
<tr>
<td>6048</td>
<td>1818</td>
<td>-468</td>
<td>-1107</td>
<td>-2043</td>
<td>468</td>
</tr>
<tr>
<td>252</td>
<td>702</td>
<td>-1848</td>
<td>-348</td>
<td>1422</td>
<td>153</td>
</tr>
<tr>
<td>3303</td>
<td>-4878</td>
<td>1332</td>
<td>252</td>
<td>1818</td>
<td>6048</td>
</tr>
<tr>
<td>1107</td>
<td>468</td>
<td>-468</td>
<td>2043</td>
<td>702</td>
<td>252</td>
</tr>
<tr>
<td>-348</td>
<td>-1848</td>
<td>4878</td>
<td>-153</td>
<td>-3303</td>
<td>-1422</td>
</tr>
<tr>
<td>2043</td>
<td>468</td>
<td>-468</td>
<td>1107</td>
<td>6048</td>
<td>252</td>
</tr>
<tr>
<td>1818</td>
<td>1332</td>
<td>1848</td>
<td>-252</td>
<td>348</td>
<td>-702</td>
</tr>
<tr>
<td>-153</td>
<td>4878</td>
<td>-1422</td>
<td>-3303</td>
<td>-468</td>
<td>-2043</td>
</tr>
<tr>
<td>-1107</td>
<td>468</td>
<td>252</td>
<td>6048</td>
<td>1332</td>
<td>1818</td>
</tr>
<tr>
<td>252</td>
<td>-1848</td>
<td>702</td>
<td>-348</td>
<td>3303</td>
<td>1422</td>
</tr>
<tr>
<td>-4878</td>
<td>153</td>
<td>468</td>
<td>-1107</td>
<td>-2043</td>
<td>-468</td>
</tr>
<tr>
<td>1818</td>
<td>1332</td>
<td>6048</td>
<td>252</td>
<td>348</td>
<td>-702</td>
</tr>
<tr>
<td>1848</td>
<td>-252</td>
<td>1422</td>
<td>3303</td>
<td>153</td>
<td>-4878</td>
</tr>
<tr>
<td>1107</td>
<td>-468</td>
<td>468</td>
<td>2043</td>
<td>1332</td>
<td>1818</td>
</tr>
<tr>
<td>252</td>
<td>6048</td>
<td>702</td>
<td>-348</td>
<td>252</td>
<td>-1848</td>
</tr>
<tr>
<td>2043</td>
<td>-468</td>
<td>-468</td>
<td>-1107</td>
<td>1848</td>
<td>348</td>
</tr>
<tr>
<td>252</td>
<td>702</td>
<td>1848</td>
<td>252</td>
<td>348</td>
<td>702</td>
</tr>
<tr>
<td>1408</td>
<td>-232</td>
<td>-232</td>
<td>-332</td>
<td>468</td>
<td>-2043</td>
</tr>
<tr>
<td>1107</td>
<td>468</td>
<td>348</td>
<td>1848</td>
<td>702</td>
<td>252</td>
</tr>
<tr>
<td>-252</td>
<td>-1848</td>
<td>-702</td>
<td>-348</td>
<td>-232</td>
<td>1408</td>
</tr>
<tr>
<td>-332</td>
<td>-232</td>
<td>468</td>
<td>1107</td>
<td>-2043</td>
<td>468</td>
</tr>
<tr>
<td>-252</td>
<td>-702</td>
<td>-1848</td>
<td>-348</td>
<td>348</td>
<td>702</td>
</tr>
<tr>
<td>1848</td>
<td>252</td>
<td>-232</td>
<td>-332</td>
<td>1408</td>
<td>-232</td>
</tr>
<tr>
<td>-1107</td>
<td>-468</td>
<td>-468</td>
<td>2043</td>
<td>-702</td>
<td>-252</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>-348</td>
<td>-1848</td>
<td>-702</td>
<td>-348</td>
<td>-252</td>
<td>-1848</td>
</tr>
<tr>
<td>-332</td>
<td>-232</td>
<td>-232</td>
<td>1408</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This is the content of q1.dat file

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>.78304832760789860</td>
<td>.10185185185185180</td>
<td>.10185185185185180</td>
</tr>
<tr>
<td>.01324796868839772</td>
<td>-.44245008972987510</td>
<td>-.44245008972987510</td>
</tr>
<tr>
<td>-.05754991027012477</td>
<td>-.05754991027012477</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>-.05754991027012477</td>
<td>-.44245008972987510</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>-.76634603522555250</td>
<td>.76634603522555250</td>
<td>-.09967936858885910</td>
</tr>
<tr>
<td>.09967936858885910</td>
<td>-.76634603522555250</td>
<td>-.250000000000000000</td>
</tr>
<tr>
<td>.76634603522555250</td>
<td>.09967936858885910</td>
<td>.250000000000000000</td>
</tr>
<tr>
<td>.250000000000000000</td>
<td>.250000000000000000</td>
<td>.250000000000000000</td>
</tr>
<tr>
<td>.10185185185185180</td>
<td>.78304832760789860</td>
<td>.013247968688397720</td>
</tr>
<tr>
<td>.10185185185185180</td>
<td>.44245008972987510</td>
<td>.44245008972987510</td>
</tr>
<tr>
<td>.05754991027012477</td>
<td>.05754991027012477</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>-.442450089729875100</td>
<td>-.05754991027012477</td>
<td>-.442450089729875100</td>
</tr>
<tr>
<td>.76634603522555250</td>
<td>-.76634603522555250</td>
<td>.09967936858885910</td>
</tr>
<tr>
<td>-.09967936858885910</td>
<td>-.09967936858885910</td>
<td>-.76634603522555250</td>
</tr>
<tr>
<td>.09967936858885910</td>
<td>.76634603522555250</td>
<td>.09967936858885910</td>
</tr>
<tr>
<td>-.250000000000000000</td>
<td>-.250000000000000000</td>
<td>-.250000000000000000</td>
</tr>
<tr>
<td>.10185185185185180</td>
<td>.013247968688397720</td>
<td>.78304832760789860</td>
</tr>
<tr>
<td>.10185185185185180</td>
<td>-.05754991027012477</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>-.442450089729875100</td>
<td>-.442450089729875100</td>
<td>.05754991027012477</td>
</tr>
<tr>
<td>.05754991027012477</td>
<td>.05754991027012477</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>-.442450089729875100</td>
<td>-.442450089729875100</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>.76634603522555250</td>
<td>-.76634603522555250</td>
<td>.09967936858885910</td>
</tr>
<tr>
<td>-.09967936858885910</td>
<td>-.09967936858885910</td>
<td>-.76634603522555250</td>
</tr>
<tr>
<td>.76634603522555250</td>
<td>.76634603522555250</td>
<td>.09967936858885910</td>
</tr>
<tr>
<td>-.76634603522555250</td>
<td>-.76634603522555250</td>
<td>.09967936858885910</td>
</tr>
<tr>
<td>.09967936858885910</td>
<td>.09967936858885910</td>
<td>.76634603522555250</td>
</tr>
<tr>
<td>-.250000000000000000</td>
<td>-.250000000000000000</td>
<td>-.250000000000000000</td>
</tr>
<tr>
<td>.01324796868839772</td>
<td>.10185185185185180</td>
<td>.10185185185185180</td>
</tr>
<tr>
<td>.78304832760789860</td>
<td>.05754991027012477</td>
<td>.05754991027012477</td>
</tr>
<tr>
<td>.442450089729875100</td>
<td>.442450089729875100</td>
<td>.05754991027012477</td>
</tr>
<tr>
<td>.442450089729875100</td>
<td>.442450089729875100</td>
<td>-.05754991027012477</td>
</tr>
<tr>
<td>.09967936858885910</td>
<td>.09967936858885910</td>
<td>.76634603522555250</td>
</tr>
<tr>
<td>-.76634603522555250</td>
<td>-.76634603522555250</td>
<td>.09967936858885910</td>
</tr>
<tr>
<td>-.09967936858885910</td>
<td>-.09967936858885910</td>
<td>-.76634603522555250</td>
</tr>
<tr>
<td>.09967936858885910</td>
<td>-.76634603522555250</td>
<td>.250000000000000000</td>
</tr>
<tr>
<td>.250000000000000000</td>
<td>.250000000000000000</td>
<td>.250000000000000000</td>
</tr>
<tr>
<td>.23263292271279750</td>
<td>.062333803773785870</td>
<td>.03025878881806710</td>
</tr>
<tr>
<td>.008107818027943149</td>
<td>.255448678408517600</td>
<td>-.255448678408517600</td>
</tr>
<tr>
<td>.033226456186295350</td>
<td>-.033226456186295350</td>
<td>.131445855765802100</td>
</tr>
<tr>
<td>-.035220810900864550</td>
<td>-.131445855765802100</td>
<td>-.035220810900864550</td>
</tr>
<tr>
<td>1.208796124955427000</td>
<td>.323895945495677300</td>
<td>-.157229782890010600</td>
</tr>
<tr>
<td>.042129458288761130</td>
<td>-.227670900630739700</td>
<td>-.061004233964073060</td>
</tr>
<tr>
<td>.227670900630739700</td>
<td>.061004233964073060</td>
<td>-.144337567297406300</td>
</tr>
<tr>
<td>.144337567297406300</td>
<td>-.144337567297406300</td>
<td>.144337567297406300</td>
</tr>
<tr>
<td>-.062333803773785870</td>
<td>-.232632922712797500</td>
<td>-.008107818027943149</td>
</tr>
<tr>
<td></td>
<td>0.030258788818806710</td>
<td>0.255448678408517600</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>0.033226456186295350</td>
<td>0.033226456186295350</td>
<td>0.035220810900864550</td>
</tr>
<tr>
<td>0.1314458557656802100</td>
<td>0.035220810900864550</td>
<td>0.1314458557656802100</td>
</tr>
<tr>
<td>0.323895945495677300</td>
<td>1.208796124955427000</td>
<td>0.042129458288761130</td>
</tr>
<tr>
<td>1.5722978829010600</td>
<td>0.061004233964073060</td>
<td>0.227670900630739700</td>
</tr>
<tr>
<td>0.061004233964073060</td>
<td>0.227670900630739700</td>
<td>0.144337567297406300</td>
</tr>
<tr>
<td>0.144337567297406300</td>
<td>0.144337567297406300</td>
<td>-0.033226456186295350</td>
</tr>
<tr>
<td>0.030258788818806710</td>
<td>0.030258788818806710</td>
<td>0.255448678408517600</td>
</tr>
<tr>
<td>0.062333803773785870</td>
<td>0.062333803773785870</td>
<td>-0.323895945495677300</td>
</tr>
<tr>
<td>0.062333803773785870</td>
<td>0.062333803773785870</td>
<td>0.061004233964073060</td>
</tr>
<tr>
<td>0.061004233964073060</td>
<td>0.061004233964073060</td>
<td>0.061004233964073060</td>
</tr>
<tr>
<td>0.035220810900864550</td>
<td>0.035220810900864550</td>
<td>0.1314458557656802100</td>
</tr>
<tr>
<td>0.1314458557656802100</td>
<td>0.1314458557656802100</td>
<td>0.1314458557656802100</td>
</tr>
<tr>
<td>0.035220810900864550</td>
<td>0.035220810900864550</td>
<td>0.1314458557656802100</td>
</tr>
<tr>
<td>0.035220810900864550</td>
<td>0.035220810900864550</td>
<td>0.1314458557656802100</td>
</tr>
<tr>
<td>0.1314458557656802100</td>
<td>0.1314458557656802100</td>
<td>0.1314458557656802100</td>
</tr>
<tr>
<td>0.061004233964073060</td>
<td>-0.018518518518518510</td>
<td>0.018518518518518510</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>-0.020334744654691040</td>
<td>0.075890300210246580</td>
<td>-0.075890300210246580</td>
</tr>
<tr>
<td>-0.020334744654691040</td>
<td>-0.075890300210246580</td>
<td>0.075890300210246580</td>
</tr>
<tr>
<td>-0.035220810900864550</td>
<td>-0.035220810900864550</td>
<td>-0.035220810900864550</td>
</tr>
<tr>
<td>-0.033226456186295350</td>
<td>0.033226456186295350</td>
<td>0.033226456186295350</td>
</tr>
</tbody>
</table>

260
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-.359116756396541800</td>
<td>.025783423063208550</td>
<td>.096225044864937560</td>
</tr>
<tr>
<td>-.096225044864937560</td>
<td>-.359116756396541800</td>
<td>.083333333333333320</td>
</tr>
<tr>
<td>-.083333333333333320</td>
<td>-.083333333333333320</td>
<td>.083333333333333320</td>
</tr>
</tbody>
</table>
Bibliography


