SPECTRAL (h-p) ELEMENT METHODS APPROACH TO THE SOLUTION OF POISSON AND HELMHOLTZ EQUATIONS USING MATLAB

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TUĞRUL MARAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

DECEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

_____

Prof. Dr. Kemal İDER
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____ _____

Asst. Prof. Dr. Cüneyt SERT
Co-Supervisor

Asst. Prof. Dr. İlker TARI
Supervisor

Examining Committee Members

Prof. Dr. M. Haluk AKSEL (METU, ME) _____

Asst. Prof. Dr. İlker TARI (METU, ME) _____

Asst. Prof. Dr. Cüneyt SERT (METU, ME) _____

Instructor Dr. Tahsin A. ÇETİNKAYA (METU, ME) _____

Prof. Dr. Ercan ATAER (Gazi U., ME) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Tuğrul MARAL

Signature :

ABSTRACT

SPECTRAL (h-p) ELEMENT METHODS APPROACH

TO THE SOLUTION OF

POISSON AND HELMHOLTZ EQUATIONS USING MATLAB

MARAL, Tuğrul

M.S., Department of Mechanical Engineering

Supervisor : Asst. Prof. Dr. İlker TARI

Co-Supervisor: Asst. Prof. Dr. Cüneyt SERT

December 2006, 129 pages

A spectral element solver program using MATLAB is written for the solution of Poisson and Helmholtz equations. The accuracy of spectral methods (p-type high order) and the geometric flexibility of the low-order h-type finite elements are combined in spectral element methods.

Rectangular elements are used to solve Poisson and Helmholtz equations with Dirichlet and Neumann boundary conditions which are homogeneous or non homogeneous. Robin (mixed) boundary conditions are also implemented.

Poisson equation is also solved by discretising the domain with curvilinear quadrilateral elements so that the accuracy of both isoparametric quadrilateral and rectangular element stiffness matrices and element mass matrices are tested.

Quadrilateral elements are used to obtain the stream functions of the inviscid flow around a cylinder problem. Nonhomogeneous Neumann boundary conditions are imposed to the quadrilateral element stiffness matrix to solve the velocity potentials.

Keywords: Spectral element method, Matlab, Poisson and Helmholtz equations.

ÖZ

SPEKTRAL (h-p) ELEMAN METODU YAKLAŞIMIYLA POISSON ve HELMHOLTZ DENKLEMLERİNİN MATLAB KULLANILARAK ÇÖZÜMÜ

MARAL, Tuğrul
Yüksek Lisans, Makine Mühendisliği Bölümü
Tez yöneticisi : Asst. Prof. Dr. İlker TARI
Ortak Tez yöneticisi : Asst. Prof. Dr. Cüneyt SERT

Aralık 2006, 129 sayfa

Poisson ve Helmholtz denklemlerinin çözümü için MATLAB kullanılarak bir spektral eleman çözücü program yazılmıştır. Yüksek dereceli ve p tipindeki spektral metodun isabetliliği ve düşük dereceli h tipi sonlu elemanlar metodunun geometrik esnekliği spektral eleman metodlarinda birleştirilir.

Homojen veya homojen olmayan Dirichlet ve Neumann sınır koşulları içeren Poisson ve Helmholtz denklemleri, dikdörtgen elemanlar kullanılarak çözülmüştür. Ayrıca, Robin sınır koşulları da gerçekleştirilmiştir.

Poisson denkleminin çözüm alanı, eğri kenarlı dörtgenlere bölünerek tekrar çözülmüştür. Bu sayede, izoparametrik eğri dörtgen ve dikdörtgen elemanların sertlik matrislerinin ve kütle matrislerinin isabetliliği test edilmiştir.

Silindir etrafındaki ideal akışın stream fonksiyonları, eğri kenarlı elemanlar kullanılarak elde edilmiştir. Homojen olmayan Neumann sınır koşulları, eğri kenarlı eleman sertlik matrisine yerleştirilerek hız potansiyelleri çözülmüştür.

Anahtar Sözcükler: Spektral eleman metodu, Matlab, Poisson ve Helmholtz denklemleri.

ACKNOWLEDGEMENTS

I would like to thank Asst. Prof. Dr. İlker Tarı for his continuous support, patience and invaluable guidance throughout this work. I am grateful to have been included in this subject. I would like to thank Asst. Prof. Dr. Cüneyt Sert for his insight and helpful suggestions.

I would like to thank Prof. Dr. Ö. Ercan Ataer who recommended me as an M.Sc student to Asst. Prof. Dr. İlker Tarı.

I would like to thank Prof. Dr. Haluk. Aksel and Asst. Prof. Dr. Tahsin Çetinkaya, who are my finite volume methods and advanced fluid mechanics instructors. I would also like to thank my other instructors at METU.

I would like to thank Assoc. Prof. Dr. Hakan I. Tarman and Prof. Dr. Turgut Tokdemir, who are my pseudospectral methods and finite element instructors.

I would like to thank my mother Mayise Maral and my father Abdullah Maral. The encouragement and inspiration of my parents was invaluable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

∇^2	Laplacian operator
Ω	Solution Domain
Γ	Boundary of Ω
Φ_i	Analytic trial (or expansion) function
N_{dof}	Number of global degrees of Freedom
v	Weight (test) functions
$\oint_{\Gamma_h}$	Boundary integral of Neumann BC
$\psi_p(\xi_1)\psi_q(\xi_2)$	Weight functions in two dimensions
$\psi_i(\xi_1)\psi_j(\xi_2)$	Trial functions in two dimensions
ξ_1 and ξ_2	Local Cartesian coordinates
Ω^e	Elemental region on physical coordinates (x, y)
$\mathcal{P}_{N_{\deg}}(\Omega_{st})$	Polynomial space of order $N_{\deg}$ on Ω_{st}
$\Phi_p^B(x)$	A nodal expansion (Lagrange expansion basis)
$h_p(x)$	One–dimensional pth order Lagrange polynomial
$\mathcal{I}$	Interpolation operator
$\hat{u}_p$	Vector of pth degree expansion coefficients
$\hat{M}$ and $\hat{M}_{ij}$	1D mass matrix evaluated with GLL polynomials
$\hat{K}$	1D stiffness matrix evaluated with GLL polynomials
$\bar{H}$	Helmholtz operator
$\otimes$	Kronecker product
x_1, x_2, $\mathbf{x}$	Global Cartesian Coordinates
$f^A(\xi_1), f^B(\xi_1), f^C(\xi_2),$ and $f^D(\xi_2)$	The shape mapping functions of each edge of the quadrilateral element

$\left\|J_{2D}\right\|$	Jacobian (Determinant of the Jacobi matrix)
$\mathcal{G}_{ij}$	Geometric terms
G_{ij} and $\left(G_{ij}\right)_{\hat{k}\hat{k}}$	Geometric factors
$\mathcal{F}(v)$	Load vector
Γ_h	Neumann Boundary conditions
$\Re$	Residual
ϕ_i	Truncated trial (or expansion) functions
$\mathcal{X}$	Space of Trial functions
$\mathcal{V}$	Space of weight (test) functions
$\mathcal{X}_N$	Finite dimensional space of trial solutions. $\mathcal{X}_N \subset \mathcal{X}$ sunspace of $\mathcal{X}$
$\mathcal{V}_{\mathrm{N}}$	Finite dimensional space of trial solutions $\mathcal{V}_{\mathrm{N}} \subset \mathcal{V}$
n	Unit outward normal
$\Phi_p^A(x)$	Modal or a hierarchical expansion
$\Phi_p^C(x)$	Modal expansion with Legendre polynomials
δ_{pq}	Kronecker delta
$L_p(\xi)$	Lagrange interpolants through the zeros of the pth degree Gauss-Lobatto polynomials
L_1 and L_2	Lengths of the rectangular domain
$(.,.)_N$	Discrete inner product given by Gauss Lobatto quadrature in each spatial direction.
$D_{N,ij}^{(1)}$ and D, $\hat{D}$	The nodal values of the first derivative of the GLL Lagrangian polynomials are called 1D differentiation matrix.
$\mathcal{L}$	Steady Diffusion operator

$\hat{D}^T$	Tranpose of 1D differentiation matrix.
x_i^e	Global Cartesian Coordinates of the element
$J_{2D}(\xi_1, \xi_2)$	Two dimensional Jacobi matrix
$\frac{\partial}{\partial x_1}$ and $\frac{\partial}{\partial x_2}$	Partial derivatives of the global coordinates
$J(\xi)$	Dimensionless Jacobian. (For 2D $J(\xi) = \lvert J_{2D} \rvert$)
I	Identity matrix
λ	Helmholtz Equation Constant

Abbreviations

SEM	Spectral element method
FEM	Finite element method
GLL	Gauss Lobatto Legendre
GL	Gauss Legendre
DOF	Global degrees of freedom

Subscripts

g	Dirichlet
h	Neumann
st	Standard (master) element in ξ_1 and ξ_2

CHAPTER 1

INTRODUCTION

Spectral element methods that combine the geometric flexibility of the finite element method with the pth order accurate spectral methods are investigated and implemented for the solution of Poisson and Helmholtz Equations using MATLAB.

The idea of the Spectral Element method (SEM) is first introduced as a global element method with Gauss Chebyshev trial functions which are not C^0 continuous at the element boundaries. As a result, a single global element is used for the solution of elliptic equations like Poisson and Helmholtz in the earliest applications of Delves et al. [21, 22]. SEM can be referred with various names like domain decomposition pseudospectral and Legendre-Galerkin methods in various references. Both of them are features of the method but it is generally called as Spectral h-p element method.

Spectral methods involve the expansion of the solution of a differential equation in a high-order orthogonal expansion, the coefficients of which are determined by a weighted residual projection technique. The schemes are "infinite order accurate" if the expansion functions are properly selected.

The finite element procedure is, in the most general sense, a weighted residual technique applied to a series of expansions, each with support over only a small region of space (an "element"). When the weighted-residual technique is directly derived from an associated variational principle, continuity of natural boundary conditions is implicitly satisfied at element boundaries as part of the convergence process.

Although the finite element and spectral methods are in fact related, but the practitioners and performers of the two methods had not dealed with each other's work until 1984. In 1984 A. Patera, in his paper [10], remembers the finite element

and spectral methods come from the same family of Galerkin methods which are the specific application of the weighted residual methods. For spectral methods the trial functions are infinitely differentiable global functions. In finite element methods, the domain Ω is divided into elements, and trial functions are specified in each element and are local in character.

Patera[10] showed the advantage and the effectiveness of the spectral element by first solving elliptic equations. Its main advantage is Gauss Lobatto Chebyshev polynomials used as trial functions which are C^0 continuous compared to the trial functions of the global element method. Most of the problems have worked on clusters of computers because of its parallel work availability.

Ronquist's thesis [16] presents a new optimal-order Legendre spectral element discretization and solution procedures for elliptic and parabolic equations. He first used Gauss Lobatto Legendre polynomials to solve 1D Helmholtz equations. The two dimensional Poisson Equation on a rectilinear domain with homogeneous Dirichlet boundary conditions is solved with GLL quadrature. Quadrilateral elements are formulated and used to solve 2D Poisson equation.

The mortar element method is first presented as a new nonconforming discretisation in Mavriplis's PhD thesis [9] in 1989. It improves the flexibility of spectral element approach in regards to the automatic mesh generation and the non-propagating local mesh refinement. Single mesh posteriori error estimators are developed to estimate the actual error incurred by the discretisation on a local elemental basis and predict the convergence behavior for decision between h and p refinement.

Fischer presented high-efficiency medium grained parallel spectral element method [20] for solution of the incompressible Navier-Stokes equations in two and three dimensional domains. It is based upon naturally concurrent Uzawa and Jacobi-preconditioned conjugate gradient iterative methods [9, 11]; data-parallel geometry-based distribution work between processors; neighbor sparsity and high-order substructuring for minimum communication; general heterogeneous locally structured (vector) and globally unstructured (parallel) constructs; and efficient embedding of vector reduction operates for inner product and norm calculations.

An important consideration for using unstructured expansion of time-dependent computations that typically arise in fluid dynamics is the numerical efficiency of the algorithm which is in the context of cost per time step. To be competitive, an unstructured expansion must be as numerically efficient as the structured expansion arising from the tensor product construction. It will be better to use a similar procedure to construct expansions within the unstructured domains. A suitable modal basis was proposed by Dubiner [17] in two dimensions and extended to three dimensions in Sherwin and Karniadakis [18, 19].

Karniadakis and Sherwin [2] presented the spectral element formulations for unstructured elements and provide many large scale applications of the partial-differential equations. They introduced a complete formulation using a modal basis which has been implemented in a new code NEKTAR. Their basis has the following properties: Jacobi polynomials of mixed weights; semi-orthogonality; hierarchical structure; generalized tensor (warped) product; variable order; and a new apex coordinate system allowing automated integration with Gaussian quadrature. They have discussed the formulation using a matrix notation which allows for an easy interpretation of the forward and backward transformations [19].

Pathria and Karniadakis [23] investigated the methods for overcoming the geometric singularities. The advantages of the method of auxiliary mapping compared to the other ways (supplementary basis functions, eigenfunctions and graded meshs) are investigated. They studied the method of auxiliary mapping with the use of supplementary basis functions. The error estimates of combined approach were confirmed through a number of numerical experiments for the Laplace, Poisson and Helmholtz equations. The method is more effective for achieving exponential convergence and analytical solution.

Pasquetti and Rapetti [24] discussed a straight edged triangle based spectral element method (SEM) with the classical quadrangle based SEM and with a standard spectral method. They solved Helmhotz equation (4.9) with both quadrangle and triangle elements for two different force functions. Because of having no Gauss quadrature rule for triangle with fekete points, the element mass matrices are full and

computational work is heavier than rectangular elements. They also proved that the condition number grows significantly faster for triangles than for quadrilaterals.

Three dimensional time dependent simulations of variable density and viscosity, miscible flows in a circular tube were done by Wilhelm and Meiburg [25]. They used an approach based on a mixed (hybrid) spectral element and Fourier spectral scheme for the spatial discretisation. The result of the temporal discretisation done with a semi-implicit method made up of 2^{nd} order backward differencing and extrapolation is a Helmholtz equation which was solved by a fast diagonalization method.

Frutos and Novo [26] presented an approximate inertial manifold based on postprocessing Galerkin method to enhance the accuracy of the spectral element method for evolutionary equations of dissipative type. The postprocess consists of the resolution of a discrete elliptic problem only once when the time evolution has been completed. A better accuracy has achieved with little increase in solution time.

A spectral element method is used in Mehdizadeh and Paraschivoiu [27] for solving the two-dimensional Helmholtz's equation, which is the equation governing time-harmonic acoustic wave. They compared SEM and FEM with Green's function, closed wave-guide and semi-infinite wave guide problem. The concluded that the omputational cost for solving Helmholtz's equation with the Galerkin finite element method increases as the wave number increases, due to the pollution effect. Spectral element method needs fewer grid points per wavelength and less computational time for the same accuracy.

Spectral h-p element discretisation was applied to the incompressible Navier–Stokes equations in three dimensions using a splitting approach in Karniadakis et al. [28] by Sherwin and Casarin[29]. After the time discretization which decouples the viscous and inviscid parts of the operator, the most computationally intensive parts of the solver are a series of elliptic equation solutions, namely one Poisson equation solution and three Helmholtz equation solutions, which are performed at each time step. Each of these elliptic solutions is preconditioned with an iterative substructuring type domain decomposition method which takes advantage of the natural splitting of the basis into interior, face, edge, and vertex basis functions. Currently, once a

suitable computational mesh has been generated the limiting computational cost of the algorithm is the solution of the four elliptic problems. They have builded an efficient preconditioning strategy for substructured solvers based on a transformation expansion basis to a low energy basis on the work of Bica [30]. By applying an additive Schwarz block preconditioner to the low-energy basis combined with a coarse space linear vertex solver they have observed reductions in execution time of up to three times for tetrahedral elements and 10 times for prismatic elements when compared to a standard diagonal preconditioner.

Guermond and Shen[31] introduced a new class of splitting schemes based on a weak form of the pressure Poisson equation and, at each time step, they only require to solve a set of Helmholtz-type equations for the velocity and a Poisson equation (in the weak form) for the pressure, just as pressure-correction and velocity-correction schemes for incompressible flows. However, unlike pressure-correction and velocity-correction schemes, the new splitting schemes are free of splitting errors and deliver full accuracy on the vorticity and the pressure.

Pavarino and Widlund [32] considered two types of iterative substructuring methods. First is designed for the Galerkin formulation of the problem. The second applies to linear systems for a discrete model derived by using Gauss-Lobatto-Legendre quadrature. For both methods, it is established that the condition number of the relevant operator grows only in proportion to $(1+\log p)^2$.

Hybrid discretisations of complex domains include triangle and quadrilateral elements. Unstructured meshes based on triangles and tetrahedral were defined in Sherwin and Karniadakis [19, 34, 35] and their parallel code Nektar. Such an approach combines the simplicity and convenience of structured domains with the geometric flexibility of an unstructured discretisation. To increase the computational efficiency of the hybrid discretisation, a new coordinate transformation from master rectangle element to master triangle element was defined by Evangelinos et al. [33]. In order to evaluate the performance of the solver they conducted tests on a scalar Helmholtz problem which forms the backbone of a splitting scheme [28] used to solve the Navier-Stokes equations.

In his PhD thesis, Warburton [36] develops a unified description of hybrid basis functions following earlier developments in [17, 19, 34, 35]. He also develops five types of basis functions which are either modal or nodal or mixed and which may or may not be hierarchical. He shows how polymorphic elements can be built and interfaced to enhance the efficiency of the unstructured spectral element method. A discontinuous Galerkin formulation is developed with a discontinuous trial basis. This basis is orthogonal, hierarchical and maintains a tensor product property (even for non-orthogonal elements), a key property for efficient implementation of high-order methods.

Sert and Beskok [38] presented spectral element formulations on polynomial (p-type) and geometric (h-type) non-conforming grids using both the pointwise matching (also known as the Constrained Approximation) and integral projection (also known as the Mortar Element) methods. These formulations were tested to solve Poisson Equation with four different types of meshes including such as p-type and h-type non-conforming elements.

A spectral method using MATLAB for the solution of the most of the PDEs including the Poisson and Helmholtz equations are discussed and implemented in Trefethen [1].

1.1 Objectives of the thesis

1. Implementing a spectral element method using MATLAB for the solution of some fundamental problems.

2. Comparing the results with the the exact results to observe and appreciate the accuracy of the approach.

3. Element stiffness matrices K and element mass matrices M are evaluated for both of the quadrilateral and rectangular elements to form steady diffusion and Helmholtz operators.

4. Steady Diffusion operator, which is equal to rectangular element stiffness matrix for a single element is used to solve the Poisson problems including both of the Dirichlet and Neumann boundary conditions which are homogeneous or nonhomogeneous.

5. In order to solve Poisson Equation with multi-elements, element stiffness matrices are assembled to form the global stiffness matrix (steady diffusion operator of multi-element domain).

6. The mesh of domain with an inner corner is drawn to investigate the geometric singularity. The capacity and accuracy of curvilinear quadrilateral elements with single or two curvy edges investigated on this domain.

7. After testing isoparametric quadrilateral element matrix evaluator program, it is used to solve inviscid flow over a cylinder problem.

8. Helmholtz equations are solved with the Helmholtz operator evaluated for a single and multi-element. The effect of the Gauss Lobatto Legendre quadrature on the accuracy of the Helmholtz operator is investigated.

1.2 Thesis Organization

In this chapter, the results of the literature survey is presented. The importance of Poisson and Helmholtz equations is discussed for the solution of other PDE's (like Navier-Stokes).

In Chapter 2, the fundamentals of the spectral element method is investigated and discussed for the solution of Poisson and Helmholtz Equations with rectangular and quadrilateral elements. The evaluation process of element stiffness and mass matrices is formulated for both of the rectangular and quadrilateral (isoparametric) elements. Load vector and Helmholtz operator evaluation is formulated by using mass matrix formulas for both of the elements. Differentiation and integration within a quadrilateral element discussed in Section 2.5.3 and 2.5.4 contains Jacobi matrix and

Jacobian evaluation. Jacobi mapping is also formulated for quadrilateral (curvilinear edged) elements.

In chapter 3, the implementations of spectral element methods for the Poisson and Helmholtz equations using MATLAB on rectangular and deformed domains discretised with a single and multi-element are discussed including the element stiffness and mass matrices evaluation discussed in Section 3.1 and 3.2. How to impose the Dirichlet and Neumann boundary conditions are also included for both kinds of the elements. The global node numbering of the elements is discussed in Connectivity matrix generation. The assembly (Direct Stiffness Summation) of element stiffness matrices to form global element stiffness matrix is implemented in Section 3.3.2. The assembly of element load vectors to form global load vector is implemented in Section 3.3.3. The implementation procedure for the SEM with two quadrilateral elements to the solution of theinviscid flow around a cylinder is discussed for a velocity potential equation including nonhomogeneous Neumann boundary conditions. The Helmholtz operator of quadrilateral elements is implemented for the solution of Helmholtz Equations in Section 3.4.

In Chapter 4, the results and discussions of the SEM solution for the Poisson equation with single element using MATLAB on rectangular domains with all type of boundary conditions (zero or nonzero Dirichlet and Neumann) are discussed in Section 4.1. The comparison between finite element and SEM is also discussed in Section 4.1.2. The results and discussions of the Poisson equation (4.4) using MATLAB on multi-element domains are in Section 4.2. These discussions include how the solution and error results are affected from the domain discretisation with three types of mesh containing both of the rectangular and quadrilateral elements. A domain containing an inner corner is selected to discuss the effects of the geometric singularity on the solution. The results and discussions of the solution of the inviscid flow around a cylinder with two quadrilateral elements are presented in Section 4.3. Velocity potential and stream function results are discussed in Section 4.3.1 and 4.3.2. The results and discussions of the solution of the Helmholtz equations on single and multi element domains are given in Section 4.4. Helmholtz equations are solved with the Helmholtz operator evaluated for a single and multi elements in

Section 4.4.1 and 4.4.2. The effect of the Gauss Lobatto Legendre quadrature on the accuracy of the Helmholtz operator is investigated.

In Chapter 5, thesis is concluded with some comments and discussions especially on the ease of implementation of the algorithms using MATLAB.

CHAPTER 2

CONFORMING SPECTRAL ELEMENT DISCRETISATION

The application of the Uzawa algorithm [16, 20] to the steady stokes or Navier-Stokes equations form a linear algebraic system. This discrete system includes element stiffness matrix called as discrete Laplacian operator [16], element mass matrix and a 2D first order differentiation matrix that is called the gradient operator [16, 20].

The splitting approach of Karniadakis et al. [28], which is an alternative method, decouples the Navier-Stokes equations into four elliptic equations. Three of them are Helmholtz equations and one of them is a Poisson equation. Therefore, the fundamentals of the spectral element method is investigated and discussed for the solution of Poisson and Helmholtz Equations with rectangular and quadrilateral elements [3]. The evaluation process of element stiffness and mass matrices is formulated for both of the rectangular and isoparametric quadrilateral elements [2, 3]. Load vector and Helmholtz operator evaluation is formulated by using mass matrix formulas for both of the elements [3]. Differentiation and integration within a quadrilateral element is discussed in Section 2.5.3 and 2.5.4 for 2D first order differentiation matrix evaluation which contains Jacobi matrix and Jacobian evaluation [2]. Jacobi mapping is also formulated for isoparametric quadrilateral (curvilinear edged) elements [2]. Spectral element method begins with the spatial discretisation of the domain.

Section 2.1 includes general discussions which are adapted from the book by Karniadakis and Sherwin[2], and by Sert[5]. Section 2.2, 2.3 and 2.4 are adapted mostly from Deville, Fischer and Mund [3] with some modifications. Head of the section 2.5 and sections from 2.5.1 to 2.5.4 are again mostly from Karniadakis and Sherwin[2]. Section 2.5.5 and 2.5.6 are mostly adapted from [3] and Fischer [20] with some modifications.

2.1 Spatial Discretisation

In order to introduce the fundamentals of spectral element formulation, strong formulation of the Poisson equation will be discussed at the beginning.

2.1.1 Strong Formulation of the Poisson equation

$$-\nabla^2 u = f \text{ on } \Omega \tag{2.1a}$$

$$u = g \text{ on } \Gamma_g \text{ the Dirichlet (essential) boundary conditions} \tag{2.1b}$$

$$n \cdot \nabla u = h \text{ on } \Gamma_h \text{ the Neumann (natural) boundary conditions} \tag{2.1c}$$

∇^2, 2^{nd} degree differentiation operator for PDE, is called Laplacian or diffusion operator, where u is the scalar unknown. Ω is the domain of the problem and $\Gamma = \Gamma_g \bigcup \Gamma_h$ is the boundary of Ω.They are the Dirichlet (essential) and Neumann (natural) boundary conditions. The unit normal n points outward from boundary Γ_h, and f, g and h are known functions.

2.1.2 Residual form

The Poisson equation is written in the residual form as Equation (2.2) by equating left-hand side to zero.

$$\Re = \nabla^2 u + f = 0 \tag{2.2}$$

2.1.3 Weighted Residual Formulation

Weighted residual assumes that the solution u can be accurately represented by the approximate solution of the form

$$u_N = u_0 + \sum_{i=1}^{N_{dof}} u_i \Phi_i \tag{2.3}$$

where Φ_i are analytic functions called the trial (or expansion) functions, and u_i are the N_{dof} unknown coefficients. By definition Φ_i is equal to zero on Dirichlet boundaries to satisfy the homogeneous boundary conditions and u_0 is selected to satisfy the initial and non-zero Dirichlet boundary conditions [2].

To construct the weighted residual form the residual is multiplied with a weight (test) function *w*. The approximate solution is forced to satisfy the residual equation in a weighted integral sense [1, 2]. The below formulation is equivalent to forcing the residual to vanish when projected onto the test space [8].

$$\int_{\Omega} (\nabla^2 u + f) v d\Omega = 0 \tag{2.4}$$

The approximate solution u and the weight function *v* belong to the following Hilbert spaces. The *trial space* is the Hilbert space where the approximate (trial) solutions are lied in and is denoted by χ.

$$\chi = \{u : u \in H^1(\Omega),\ \mathrm{u} = \mathrm{g\ on}\ \Gamma_g\} \tag{2.5}$$

$$\mathcal{V} = \{w : w \in H^1(\Omega),\ v = 0\ \mathrm{on}\ \Gamma_g\} \tag{2.6}$$

The trial and test spaces χ and $\mathcal{V}$ contain infinite number of functions, as a result it is an infinite dimensional problem. The trial and test subspaces, χ_N and $\mathcal{V}_{\mathrm{N}}$, are selected to contain finite number of functions and are considered as two finite dimensional approximation spaces that belong to the $\chi_N \subset \chi$ and $\mathcal{V}_{\mathrm{N}} \subset \mathcal{V}$. The test space $\mathcal{V}_{\mathrm{N,0}}$ belongs to the $\mathcal{V}_{\mathrm{N}} \subset \mathcal{V}_{\mathrm{N,0}}$ where the subscript 0 refers to the fact that it

satisfies the boundary conditions on Γ_g. This means weight (test) function *v* (or *w* in [4]) is zero on all Dirichlet boundaries.

The approximate solution $u_N \subset \chi_N$ is then rewritten as

$$u_N = \sum_{i=0}^{N} u_i \phi_i \tag{2.7}$$

The trial functions ϕ_i are used as basis functions for a truncated series expansion of the solution.

2.1.4 Weak formulation

To construct the weak form, integration by parts is applied to the first term of Equation (2.4).

$$\int_{\Omega} \nabla u \cdot \nabla v d\Omega = \int_{\Omega} f \cdot v d\Omega + \oint_{\Gamma_h} hv ds \tag{2.8}$$

To impose the Neumann (natural) BC naturally and lower the order of the first term, integration by parts is applied; as a result linear order polynomials can be used for preconditioning the mesh.

2.1.5 Domain Discretization

It is an application of the weak formulation to each element individually.

$$\int_{\Omega} \nabla u^e \cdot \nabla v^e d\Omega^e = \int_{\Omega} f^e \cdot v^e d\Omega^e + \oint_{\Gamma_h^e} h^e v^e ds \tag{2.9}$$

2.1.5.1 Weak formulation for a single element

The Equation (2.9) becomes

$$\int_{\Omega} \frac{\partial v}{\partial x}\frac{\partial u}{\partial x}dxdy + \int_{\Omega} \frac{\partial v}{\partial y}\frac{\partial u}{\partial y}dxdy = \int_{\Omega} f(x,y)\cdot v(x,y)dxdy + \oint_{\Gamma_h^e} h^e v^e ds \tag{2.10}$$

If the whole solution domain is treated as a single element, then the p-type method becomes a spectral method [8, 16]. A lot of methods are produced by the choice of the expansion (trial) function ϕ_i and the test function *v*, for example least squares and collocation. Our main concern is the Galerkin method also known as Bubnov-Galerkin.

2.1.5.2 Weak Formulation for Homogeneous Neumann BC

The last term of Equation (2.10) is the boundary term which is zero for homogeneous Neumann boundary conditions $n\cdot\nabla u = h = 0$ on Γ_h.Thus, Equation (2.10) becomes

$$\int_{\Omega} \frac{\partial v}{\partial x}\frac{\partial u}{\partial x}dxdy + \int_{\Omega} \frac{\partial v}{\partial y}\frac{\partial u}{\partial y}dxdy = \int_{\Omega} f(x,y)\cdot v(x,y)dxdy \tag{2.11}$$

As it is seen, natural boundary conditions are included in the solution naturally.

2.2 Element Discretisation for 2D Problems

Tensor product forms are used to develop steady diffusion operator and other spectral element operators for elliptic problems like Poisson equation in a rectangular domain. u is approximated in the element Ω^e by mapping each element in physical coordinate (x, y) to a master element in $(\xi_1,\xi_2)\in[-1,1]^2$ reference coordinate system (computational domain). Typical function u(x,y)$\in$ $\mathcal{V}_N$ has the representation

$$\mathrm{u}(\xi_1,\xi_2) = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij}\psi_i(\xi_1)\psi_j(\xi_2) \tag{2.12}$$

2.2.1 Two dimensional differentiations in the computational domains

The differentiations with respect to ξ_1 and ξ_2 are expressed in the reference coordinates as

$$\frac{\partial u}{\partial \xi_1} = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij} \frac{\partial \psi_i(\xi_1)}{\partial \xi_1} \psi_j(\xi_2) \tag{2.13}$$

$$\frac{\partial u}{\partial \xi_2} = \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij} \psi_i(\xi_1) \frac{\partial \psi_j(\xi_2)}{\partial \xi_2} \tag{2.14}$$

Here, it is needed to select expansion bases for trial and weight functions. Various expansion bases are discussed and compared with each other in Appendix B.

2.3 Galerkin methods

The test functions are chosen to be same as the trial (or expansion) functions. As a result the spaces $\mathcal{X}$ and $\mathcal{V}$ are chosen to be the same, and the weak formulation Equation (2.9) is used as a starting point of the method which is called a Galerkin weighted-residual method so the weight (test) function *v* is written as

$$v = \phi_{pq}(\xi_1, \xi_2) = \sum_{p=0}^{N}\sum_{q=0}^{N} \psi_p(\xi_1)\psi_q(\xi_2) \tag{2.15}$$

2.4 Rectangular geometries

In this section, 2D rectangular stiffness matrix and mass matrix evaluation are formulated. Helmholtz operator and variable-coefficient case is also formulated.

2.4.1 2D Mass matrix Evaluation

The 2D mass matrix is derived by simply evaluating (*u, v*) as seen in Equation (2.16) for all $u, v \in \mathcal{V}_N$ for rectangular domains. Mass matrix is used for evaluating load

vector with force function and some operators like Helmholtz in some problems like Poisson, wave and advection-diffusion (Burgers) etc.

$$(u,v)=\int_{\Omega} vudV=\sum_{pj}\sum_{iq} v_{pq}\left(\int_{\Omega}\psi_p(\xi_1)\psi_q(\xi_2)_q\psi_i(\xi_1)\psi_j(\xi_2)dx_1dx_2\right)u_{ij} \tag{2.16}$$

For a rectangular domain [x,y]$\in$ [$0, L_1$]$\times$[$0, L_2$]

$$M_{\hat{k}k}=\int_{-1}^{1}\int_{-1}^{1}\psi_p(\xi_1)\psi_i(\xi_1)\psi_j(\xi_2)\psi_q(\xi_2)\frac{L_1}{2}d\xi_1\frac{L_2}{2}d\xi_2 \tag{2.17}$$

$$M=\frac{L_1L_2}{4}\hat{M}\otimes\hat{M} \tag{2.18}$$

L_1 and L_2 are the lengths of the edges of the rectangular element or domain.

The availability of a diagonal mass matrix is a particularly useful feature in unsteady or temporal discretization problems that require frequent application of M^{-1} because computational cost of inverting the mass matrix is an important issue. However, the most important cost is the computational cost of constructing the matrix system which involves numerical integration including the mass matrix.

$\hat{M}$ will be diagonal if the basis functions are orthogonal with respect to the inner product. One possibility is to choose $\{\psi_p\}_{i=0}^{N}$ to be a set of orthogonal functions such as Legendre polynomials, which will be discussed as Modal Legendre expansion $\Phi_p^C(x)$ in Appendix B.1.2.2; however, this expansion will not automatically satisfy the essential boundary conditions. The difficulty arises when it is tried to ensure a degree of continuity in the global expansion at elemental boundaries. For a domain with a single element, element and domain boundaries are the same. For accurate results, it is sufficient to guarantee u_N. Typically, in the finite element methods this is satisfied by imposing a C_0 continuity between elemental regions; that is, the global

expansion modes are continuous everywhere in the solution domain although the derivatives may not be.

As an alternative to fully orthogonal basis, one can use localized Lagrangian interpolants coupled with mass lumping as FEM, in which the mass matrix is replaced by a diagonal matrix with an identical row sum. This is achieved by the SEM, which is discussed in Appendix B.2.2 in a more formal setting.

The SEM is defined not only by its choice of Lagrangian basis functions, $\psi_i = L_i$, but also by the associated quadrature rule or inner product (.,.) of Equation (2.16) which is approximated by the discrete inner product $(.,.)_N$ given by Gauss Lobatto quadrature in each spatial direction. For a single coordinate direction and with $\{\xi_0, \xi_1, ..., \xi_N\}$ and $\{w_0, w_1, ..., w_N\}$ denoting the quadrature nodes and weights respectively, the integral of g(ξ) function can be defined as

$$\int_{-1}^{1} g(\xi) d\xi \approx \sum_{k=0}^{N} w_k g(\xi_k) \tag{2.19}$$

Each inner product in the SEM is computed by first evaluating the integrand, then substituting the quadrature (2.19) for integration. Thus, entries in $\hat{M}$ become

$$\hat{M}_{ij} = \sum_{k=0}^{N} w_k L_i(\xi_k) L_j(\xi_k) \tag{2.20}$$

However, because the basis is Lagrangian [i.e., $L_i(\xi_j) = \delta_{ij}$, where δ_{ij} is the Kronecker Delta], it is clear that due to the cardinality property of the Lagrangian basis on the GLL grid, for spectral elements, $\hat{M}$ is diagonal

$$\hat{M} = diag(w_i) \tag{2.20a}$$

The stiffness matrix K is derived in a similar manner. At first, p(x) and q(x) are defined as constant, leading to a particularly simple form that is amenable to both fast evaluation and fast inversion.

2.4.2 Rectangular Element Stiffness Matrix Evaluation

In $\mathbb{R}^2$, the energy inner product is

$$\mathcal{A}(u,v)=\int_{\Omega}\left(p\frac{\partial v}{\partial x}\frac{\partial u}{\partial x}+p\frac{\partial v}{\partial y}\frac{\partial u}{\partial y}+qvu\right)dx \tag{2.21}$$

For Helmholtz problem, q has got a value. For Poisson problems p=1 and q=0, as a result last term of Equation (2.21) is zero for Poisson problems.

Using the expansions Equations (2.12), (2.13), (2.14) and (2.15) for u and v, the first term on the RHS of Equation (2.21)

$$\int_{\Omega} p\frac{\partial v}{\partial x}\frac{\partial u}{\partial x}dx=\sum_{pq}\sum_{ij}v_{pq}\,p\frac{L_2}{L_1}\left(\int_{-1}^{1}\psi_q\psi_j d\xi_2\right)\left(\int_{-1}^{1}\frac{\partial\psi_p}{\partial\xi_1}\frac{\partial\psi_i}{\partial\xi_1}d\xi_1\right)u_{ij}$$

$$\int_{\Omega} p\frac{\partial v}{\partial x}\frac{\partial u}{\partial x}dx=\sum_{pq}\sum_{ij}v_{pq}\,p\frac{L_2}{L_1}\hat{M}_{qj}\hat{K}_{pi}u_{ij} \tag{2.22}$$

where $\hat{K}$ is the one-dimensional stiffness matrix on [-1,1]. Using tensor notation as it is in Equation (2.18) gives

$$\int_{\Omega} p\frac{\partial v}{\partial x}\frac{\partial u}{\partial x}dx=p\frac{L_2}{L_1}\underline{v}^T\left(\hat{M}\otimes\hat{K}\right)\underline{u} \tag{2.22a}$$

Using the expansions Equations (2.12), (2.13), (2.14) and (2.15) for u and v, the second term on the RHS of Equation (2.21) is

$$\int_{\Omega} p\frac{\partial v}{\partial y}\frac{\partial u}{\partial y}dx=\sum_{pq}\sum_{ij}v_{pq}\,p\frac{L_1}{L_2}\left(\int_{-1}^{1}\frac{\partial\psi_q}{\partial\xi_2}\frac{\partial\psi_j}{\partial\xi_2}d\xi_2\right)\left(\int_{-1}^{1}\psi_p\psi_i d\xi_1\right)u_{ij}$$

$$=\sum_{pq}\sum_{ij}v_{pq}\,p\frac{L_1}{L_2}\hat{K}_{qj}\hat{M}_{pi}u_{ij} \tag{2.23}$$

where $\hat{K}$ is the one-dimensional stiffness matrix on [-1,1]. Using tensor notation as it is in equation (2.18) gives

$$\int_{\Omega} p \frac{\partial v}{\partial y} \frac{\partial u}{\partial y} dx = p \frac{L_1}{L_2} \underline{v}^T \left(\hat{K} \otimes \hat{M} \right) \underline{u} \tag{2.23a}$$

Numerical quadrature can be applied also for evaluation of $\hat{K}$

$$\hat{K}_{ij} = \sum_{k=0}^{N} w_k \psi_i^{'}(\xi_{1k}) \psi_j^{'}(\xi_{1k}) = \sum_{k=0}^{N} w_k \frac{\partial \psi_i(\xi_{1k})}{\partial \xi_1} \frac{\partial \psi_j(\xi_{1k})}{\partial \xi_1}$$

$$\hat{K}_{ij} = \sum_{k=0}^{N} w_k \frac{\partial L_i(\xi_{1k})}{\partial \xi_1} \frac{\partial L_j(\xi_{1k})}{\partial \xi_1} \tag{2.24}$$

ξ_{1k} is the tensor product matrix of the ξ_1 (local) coordinates of the element. Since the number of nodes in x and y direction is equal to each other, $\hat{K}_{pi}$ is equal to $\hat{K}_{qj}$. This equation is remembered as 1D stiffness matrix and its implementation is also similar.

Exact evaluation of mass matrix means accurate evaluation of 1D stiffness matrix ($\psi_i^{'}\psi_j^{'}$ has got a degree of 2N-2). Because the N+1 point Gauss-Lobatto Legendre quadrature rule is exact for all polynomials of degree 2N-1 or less.

$\frac{\partial L_i(\xi_{1k})}{\partial \xi_1}$ is the derivative of the Lagrangian polynomials. It will be the differentiation matrix $D_{N,ij}^{(1)}$ and can be written from [3] as

$$D_{N,ij}^{(1)} := \left. \frac{dL_j}{d\xi} \right|_{\xi=\xi_i} = \begin{Bmatrix} \frac{L_N(\xi_i)}{L_N(\xi_j)} \frac{1}{\xi_i - \xi_j}, \rightarrow i \neq j \\ -\frac{(N+1)N}{4}, \rightarrow i = j = 0 \\ \frac{(N+1)N}{4}, \rightarrow i = j = N \\ 0, \rightarrow i = j = 1, ..., N-1 \end{Bmatrix} \tag{2.24a}$$

The matrix elements are the nodal values of the first derivative of the GLL Lagrangian polynomials.

Combining Equations (2.22a) and (2.23a) with a similar expression for the *y*-derivatives and with the mass matrix derived in Equation (2.18) yields

$$\mathcal{A}(u,v) = \underline{v}^T \left(p\frac{L_2}{L_1}\left(\hat{M}\otimes\hat{K}\right) + p\frac{L_1}{L_2}\left(\hat{K}\otimes\hat{M}\right) + q\frac{L_1L_2}{4}\left(\hat{M}\otimes\hat{M}\right)\right)\underline{u} \tag{2.25}$$

At the RHS of Equation (2.11), Calculation of the force (load) matrix is needed for all of the PDE's if there is a force function

$$\mathcal{F}(v) = \int_{\Omega} f(x,y)\cdot v(x,y)dxdy = \frac{L_1L_2}{4}\underline{v}^T\left(\hat{M}\otimes\hat{M}\right)\underline{f} \tag{2.26}$$

The second expression results from the insertion of the interpolant of f(x) into the RHS of Equation (2.11).

2.4.3 Helmholtz operator for a single element

Equations from (2.18) to (2.26) describe the essential mechanics for evaluating the bilinear form (u,v), $\mathcal{A}(u,v)$ and $\mathcal{F}(v)$ for any element pair, where $u,v\in V_N$. The discrete Helmholtz operator

$$\bar{H} := p\left[\frac{L_2}{L_1}\left(\hat{M}\otimes\hat{K}\right) + p\frac{L_1}{L_2}\left(\hat{K}\otimes\hat{M}\right) + q\frac{L_1L_2}{4}\left(\hat{M}\otimes\hat{M}\right)\right] \tag{2.27}$$

is sometimes referred as the *Neumann* operator, because it is the system governing the homogeneous Neumann problem. It is symmetric, positive and definite unless q=0, in which case it has 1D nullspace corresponding to the constant mode.

2.4.4 Laplacian operator for a single element

For q=0, the Laplacian operator, also called as Steady diffusion operator, will be

$$\mathcal{L} := p\left[\frac{L_2}{L_1}\left(\hat{M}\otimes\hat{K}\right)+p\frac{L_1}{L_2}\left(\hat{K}\otimes\hat{M}\right)\right] \tag{2.28}$$

For Laplacian problem, where $\mathcal{L}$ is Laplacian operator

$$\mathcal{L}u=\mathcal{F}(v) \tag{2.29}$$

Here, the local stiffness (element) matrix of rectangular element on the reference coordinates is calculated for steady diffusion (Laplacian) operator. Sometimes K is used as its symbol, because its name is stiffness matrix.

All of the stiffness matrices of the square element with the same degree, which are on different physical coordinates, are the same. $\mathcal{F}(v)$ will be different, whether force function is dependent on x,y coordinates or not. These issues will be handled in the implementation Chapter 3.

Dirichlet and Neumann boundaries are considered and discussed in Chapter 3 implementation process.

For nonzero Neumann if the domain is square, poldif.m can be used to find the differentiation matrix with respect to x or y of the element [7, 8 and 12].

2.4.5 The Variable-Coefficient Case

To develop the system matrices for the case of variable p(x,y), evaluation of the integrals in the first term at the RHS of Equation (2.21) requires an evaluation once again. The reference and physical coordinates are the same. For two-dimensional domains, the first term of Equation (2.21) is written as

$$\mathcal{A}_x(u,v) := \int_{-1}^{1}\int_{-1}^{1} p(\xi_1,\xi_2)\frac{\partial v}{\partial \xi_1}\frac{\partial u}{\partial \xi_1}d\xi_1 d\xi_2 \tag{2.30a}$$

which constitutes a single term in the energy inner product $\mathcal{A}(u,v)$. To generate the discrete operators, the equations of expansions (2.12), (2.13), (2.14) and (2.15) for *u*, *v*, and p are inserted into Equation (2.30a):

$$\mathcal{A}_x(u,v) = \sum_{pq}\sum_{ij} v_{pq} \sum_{mn} p_{mn} \left(\int_{-1}^{1} \psi_q \psi_j \psi_n d\xi_2 \right) \left(\int_{-1}^{1} \frac{\partial \psi_p}{\partial \xi_1} \frac{\partial \psi_i}{\partial \xi_1} \psi_m d\xi_1 \right) u_{ij} \tag{2.30b}$$

If left in this form, the cross term p_{mn} destroys the tensor-product form and leads to an unacceptable fill in the stiffness matrix

$$\mathcal{A}_y(u,v) = \sum_{pq}\sum_{ij} v_{pq} \sum_{mn} p_{mn} \left(\int_{-1}^{1} \frac{\partial \psi_q}{\partial \xi_2} \frac{\partial \psi_j}{\partial \xi_2} \psi_n d\xi_2 \right) \left(\int_{-1}^{1} \psi_p \psi_i \psi_m d\xi_1 \right) u_{ij} \tag{2.30c}$$

The SEM avoids this difficulty through the use of a high-order quadrature rule coupled with Lagrangian basis functions based on the Gauss Lobatto Legendre points ξ_i by employing the basis functions $\psi_i = L_i(\xi_1)$ described before. The first integral on the right in Equation (2.30c) is approximated as

$$\sum_{k=0}^{N} \frac{dL_p(\xi_{1k})}{d\xi_1} \frac{dL_i(\xi_{1k})}{d\xi_1} L_m(\xi_{1k}) w_k = \frac{dL_p(\xi_{1m})}{d\xi_1} \frac{dL_i(\xi_{1m})}{d\xi_1} w_m = \hat{D}_{mp} \hat{D}_{mi} w_m \tag{2.31}$$

where $\hat{D}_{mp} := \frac{dL_p(\xi_{1m})}{d\xi_1}$. The second integral in Equation (2.30c) is approximated by

$$\sum_{k=0}^{N} L_q(\xi_{1k}) L_j(\xi_{1k}) L_n(\xi_{1k}) w_{1k} = \delta_{qn} \delta_{jn} w_n \tag{2.32}$$

Let P:=diag($p_{\hat{m}}$) and W:=diag($w_{\hat{m}}$) be the diagonal matrices having entries $p_{\hat{m}} = p_{mn}$ and $w_{\hat{m}} := p_{mn} w_m w_n$ respectively, where $\hat{m}$ =1+m+(N+1)n corresponds to the natural ordering of the nodes. *W* can be expressed in terms of tensor-product forms where $\hat{M} = diag(w_i)$

$$W = P(\hat{M} \otimes \hat{M}) \tag{2.33}$$

and the integral expression eqn. (2.30) recast in terms of *W*,

$$\mathcal{A}_x(u,v) = \underline{v}^T \left(I \otimes \hat{D}^T\right) W \left(I \otimes \hat{D}\right) \underline{u} \tag{2.34}$$

From this, the spectral–element stiffness matrix can be concluded for Equation (2.21) for q=0 with variable *p*(x,y) is of the form

$$\bar{K} = \left(I \otimes \hat{D}^T\right) W \left(I \otimes \hat{D}\right) + \left(\hat{D}^T \otimes I\right) W \left(\hat{D} \otimes I\right) \tag{2.35}$$

Dirichlet and Neumann boundary conditions are also an implementation issue.

The presence of matrix *P* in Equation (2.33) is generally not in tensor form, and therefore the fast diagonalization method cannot be used to invert $\bar{K}$. However, because the cost of applying the diagonal matrix *W* to a vector is only $O(N^d)$, the leading order of complexity of forward application of $\bar{K}$ is governed by the differentiation associated with the matrices $\hat{D}$ and $\hat{D}^T$ and is only $O(N^{d+1})$. Note that if *W* were full rather than diagonal, the cost of applying $\bar{K}$ would be $O(N^{2d})$. It is precisely the use of the diagonal mass matrix $\hat{M}$ [the approximation equations (2.31) and (2.32)] that lead to a favorable complexity estimate in the variable – coefficient case and that is central to the utility of the high-order methods in complex geometries.

2.5 Local elemental operations for quadrilateral elements

It is recalled that to solve the Galerkin formulation of the Laplace equation in deformed geometries, the inner products of the form within every elemental region is needed to evaluate. The equation (2.21) for q=0 can be written again like

$$\mathcal{A}(u,v) = \int_{\Omega^e} p(\mathbf{x}) \nabla v \cdot \nabla u d\mathbf{x} = \int_{\Omega_{st}} p(\mathbf{x}) \nabla v \cdot \nabla u \; J(\xi) d\xi \tag{2.36}$$

where Ω^e denotes the element region, Ω_{st} denotes the standard elemental region (computational domain),**x** denotes the Cartesian coordinates and *J* is the Jacobian of the mapping between these two regions. From the structure of the inner product, there are three important concepts First, integration within Ω_{st}, second; differentiation in the standard region Ω_{st} and, last, differentiation in the elemental region Ω^e. To perform the differentiation and integration within the elemental region, a mapping between these regions is defined. It is called elemental mapping.

Operations within general-shaped elements

To consider these cases a one-to-one mapping between the Cartesian coordinates (x_1, x_2), and the Local Cartesian coordinates (ξ_1, ξ_2) which are denoted by

$$x = x_1^e(\xi_1, \xi_2), \quad y = x_2^e(\xi_1, \xi_2) \tag{2.5.1}$$

are defined in two dimensions, and similarly

$$x = x_1^e(\xi_1, \xi_2, \xi_3), \qquad y = x_2^e(\xi_1, \xi_2, \xi_3), \qquad z = x_3^e(\xi_1, \xi_2, \xi_3) \tag{2.5.2}$$

are defined in three dimensions. They have been called physical coordinates(x,y) and reference coordinates (ξ_1, ξ_2) as mentioned in Section 2.2.

A mapping x_i^e is defined from the elemental region to the standard region for straight–sided elements

2.5.1 Elemental mappings for general straight-sided elements

For elemental shapes with straight sides a simple mapping may be constructed using the linear vertex modes of a modified hierarchical modal expansion.

Because of linear order elements, it is called bilinear mapping for an arbitrary-shaped straight-sided quadrilateral where only the Cartesian coordinates of the vertices need

to be prescribed. For the straight-sided quadrilateral with vertices labeled with letters have counter-clockwise order

$$x_i = x_i^A \frac{(1-\xi_1)}{2}\frac{(1-\xi_2)}{2} + x_i^B \frac{(1+\xi_1)}{2}\frac{(1-\xi_2)}{2} + x_i^D \frac{(1-\xi_1)}{2}\frac{(1+\xi_2)}{2} + x_i^C \frac{(1+\xi_1)}{2}\frac{(1+\xi_2)}{2}, \qquad \text{i=1, 2} \tag{2.37}$$

For high-order element matrices, Equation (2.37) will be subparametric. When developing a mapping, it is important to ensure that the Jacobian (determinant of the Jacobi matrix) of the mapping to the standard region is nonzero and of the same sign. To ensure this condition that is satisfied when using the mappings given above, all elemental regions are required to have internal corners with angles that are less than 180°. Accordingly, quadrilaterals must be convex. It is impossible to generate local stiffness (element) matrix for quadrilaterals with an interior angle greater than 180° (Concave quadrilateral elements).

2.5.2 Elemental mappings for general curvilinear elements

From Equation (2.37), it is seen that this simply involves the vertex modes of the modified hierarchical expansion basis within a quadrilateral domain. The mapping can be written as

$$x_1 = x_1^e(\xi_1,\xi_2) = \sum_{p=0}^{N_1}\sum_{q=0}^{N_2}(x_1)_{pq}\,\phi_{pq}(\xi_1,\xi_2) \tag{2.38a}$$

$$x_2 = x_2^e(\xi_1,\xi_2) = \sum_{p=0}^{N_1}\sum_{q=0}^{N_2}(x_2)_{pq}\,\phi_{pq}(\xi_1,\xi_2) \tag{2.38b}$$

where $\phi_{pq} = \psi_p^a(\xi_1)\psi_q^a(\xi_1)$ and x_{pq}^i=0, except for the vertex modes which have a value of

$$(x_1)_{0,0} = x_1^A \qquad (x_1)_{N_1,0} = x_1^B \tag{2.38c}$$

$$(x_1)_{N_1,N_2} = x_1^C \qquad (x_1)_{0,N_1} = x_1^D$$

The construction of a mapping based upon the expansion modes in this form can be extended to include curved-sided regions using an isoparametric representation. In this technique this geometry is represented with an expansion of the same form and polynomial order as the unknown variables.

To describe a curved region as seen in Fig. 2.1 requires more information than the values of the vertex locations as a straight-sided region. A definition of a mapping of the shape of each edge in terms of the local Cartesian coordinates is denoted by $f_i^A(\xi_1), f_i^B(\xi_1), f_i^C(\xi_2),$ and $f_i^D(\xi_2)$. The process of defining the mapping functions is considered as part of mesh generation process of the isoparametric quadrilateral.

Knowing the definition of the edges (or faces in three dimensions), a mapping for a curvilinear domains can be determined using the iso-parametric form of equation to include more nonzero expansion coefficients than simply the vertex contributions. If it isn't represented by a polynomial of appropriate order, it is needed to approximate the shape mapping $f_i(\xi)$.

This can be done by approximating the edge function in terms of the Lagrange polynomial. The following approximations for $f_i^A(\xi_1)$ is

$$f^A(\xi_1) \simeq \sum_{p=0} f^A(\xi_i, i) h_p(\xi_1) \tag{2.39a}$$

$$\simeq \sum_{p=0} \hat{x}_{p0}^i \psi_p(\xi_1) \tag{2.39b}$$

One important feature of the approximation, and, consequently, the mapping x_i is that the vertices of each element coincide so that elements remain continuous. One way to ensure this is to use a collocation projection where the collocation points include the endpoints $\xi_1 \pm 1$. The Lagrange representation of Equation (2.39a) is,

therefore, a consistent way of approximating $f_i^A(\xi)$. Using the Gauss-Lobatto Legendre quadrature points for the collocation projection is beneficial. By making collocation projections at a series of nodal points, the function f^A as a polynomial can be equivalently expressed in terms of a hierarchical expansion, $\psi_p(\xi)$, to obtain the coefficients $\hat{x}_{p0}^i$ in Equation (2.39b). This final transformation can be performed either by a collocation or Galerkin projection if the polynomials span the same space. If more collocation points are used, then a modified Galerkin projection can be applied. Having determined the coordinate expansion coefficients, $\hat{x}_{p0}^i$, Equation (2.38) can be evaluated to determine the iso-parametric mapping from the standard region to the curvilinear region.

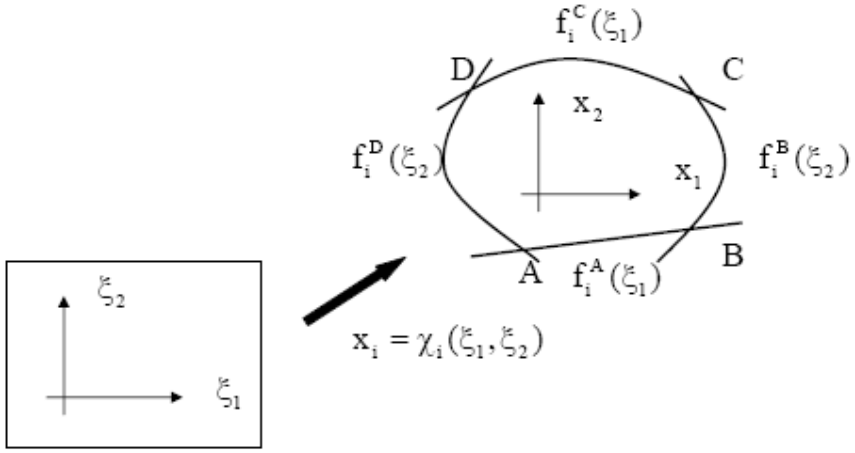


Figure 2.1 A general curved quadrilateral element can be described in terms of a series of parametric functions $f^A(\xi_1), f^B(\xi_1), f^C(\xi_2),$ and $f^D(\xi_2)$ [2]. Representing these functions as a discrete expansion, An isoparametric mapping $x_i^e(\xi_1, \xi_2)$ relating the standard region (ξ_1, ξ_2) to the deformed region (x_1, x_2) can be constructed [2]. (Adopted from [8])

The form of the boundary-interior decomposition of the modal quadrilateral and hexahedral expansion is discretely equivalent to using a linear blending function, as

originally proposed by Gordon and Hall [13]. For the quadrilateral region shown in Figure 2.1 the linear blending function is written as

$$\begin{aligned}\chi_i(\xi) = f^A(\xi_1)\frac{(1-\xi_2)}{2} + f^C(\xi_1)\frac{(1+\xi_2)}{2} + f^D(\xi_2)\frac{(1-\xi_1)}{2} + f^B(\xi_2)\frac{(1+\xi_1)}{2} \\ -\frac{(1-\xi_1)}{2}\frac{(1-\xi_2)}{2}f^A(-1) - \frac{(1+\xi_1)}{2}\frac{(1-\xi_2)}{2}f^A(1) \\ -\frac{(1-\xi_1)}{2}\frac{(1+\xi_2)}{2}f^C(-1) + \frac{(1+\xi_1)}{2}f^C(1)\end{aligned} \tag{2.40}$$

where the vertex points are continuous (for example, $f^A(-1), f^D(-1)$). If the analytic curves $f^A(\xi_1), f^B(\xi_1), f^C(\xi_2),$ and $f^D(\xi_2)$ replaced in Equation (2.39a) and rearranged, the expansion of the form given by Eqn. (2.38a, b) can be obtained. The blending function Eqn. (2.40) with approximations of the form Eqn. (2.39a) to the mapped edges has been applied in spectral element methods. For curved triangular, tetrahedral or unstructured elements, the linear blending function expressed in terms of the local collapsed coordinates should not be used as this can generate a non-smooth Jacobian at the singular vertices. C^0 continuity can be lost.

2.5.3 Integration within an elemental region

After the coordinates of the element's inner and surface (face) nodes, all the partial derivatives required to determine the Jacobian can be evaluated. $\frac{\partial \xi_1}{\partial x_1}$, $\frac{\partial \xi_2}{\partial x_1}$, $\frac{\partial \xi_1}{\partial x_2}$ and $\frac{\partial \xi_2}{\partial x_2}$ must be calculated to find the Laplacian operator .

$$\left|J_{2D}\right| = \begin{vmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} \end{vmatrix} = \frac{\partial x_1}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_1} \tag{2.41}$$

2.5.4 Differentiation within an elemental region

To differentiate a function within the arbitrary elemental region Ω^e, the chain rule is applied for the two-dimensional case, gives

$$\nabla = \begin{bmatrix} \dfrac{\partial}{\partial x_1} \\ \dfrac{\partial}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \xi_1}{\partial x_1}\dfrac{\partial}{\partial \xi_1} + \dfrac{\partial \xi_2}{\partial x_1}\dfrac{\partial}{\partial \xi_2} \\ \dfrac{\partial \xi_1}{\partial x_2}\dfrac{\partial}{\partial \xi_1} + \dfrac{\partial \xi_2}{\partial x_2}\dfrac{\partial}{\partial \xi_2} \end{bmatrix} \tag{2.42}$$

In order to evaluate partial derivatives of the form $\dfrac{\partial \xi_1}{\partial x_1}$, they are expressed in terms of ξ_1, ξ_2 which is the terms of Jacobian.

The total change $x_1 = x_1^e(\xi_1, \xi_2)$ and $x_2 = x_2^e(\xi_1, \xi_2)$ is

$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x_1}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_2} \\ \dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_2}{\partial \xi_2} \end{bmatrix} \begin{bmatrix} d\xi_1 \\ d\xi_2 \end{bmatrix} \tag{2.42a}$$

which can be inverted to obtain

$$\begin{bmatrix} d\xi_1 \\ d\xi_2 \end{bmatrix} = \underbrace{\frac{1}{|J_{2D}|} \begin{bmatrix} \dfrac{\partial x_2}{\partial \xi_2} & -\dfrac{\partial x_1}{\partial \xi_2} \\ -\dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_1} \end{bmatrix}}_{J^{-1}} \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} \tag{2.43}$$

However, as the mapping is assumed to be one to one and have an inverse

$$\xi_1 = \left(x_1^e\right)^{-1}(x_1, x_2), \qquad \xi_2 = \left(x_2^e\right)^{-1}(x_1, x_2) \tag{2.5.4.1}$$

and, as a result, it is obtained that

$$\begin{bmatrix} d\xi_1 \\ d\xi_2 \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \xi_1}{\partial x_1} & \dfrac{\partial \xi_1}{\partial x_2} \\ \dfrac{\partial \xi_2}{\partial x_1} & \dfrac{\partial \xi_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} \tag{2.44}$$

which by comparison, finally gives

$$\begin{aligned} \frac{\partial \xi_1}{\partial x_1} &= \frac{1}{|J_{2D}|}\frac{\partial x_2}{\partial \xi_2} \qquad & \frac{\partial \xi_1}{\partial x_2} &= -\frac{1}{|J_{2D}|}\frac{\partial x_1}{\partial \xi_2} \\ \frac{\partial \xi_2}{\partial x_1} &= -\frac{1}{|J_{2D}|}\frac{\partial x_2}{\partial \xi_1} \qquad & \frac{\partial \xi_2}{\partial x_2} &= \frac{1}{|J_{2D}|}\frac{\partial x_1}{\partial \xi_1} \end{aligned} \tag{2.44b}$$

The two-dimensional gradient operator can be evaluated as all the partial derivatives are expressed in terms of differentials with respect to ξ_1, ξ_2.

2.5.5 Quadrilateral (Curvilinear Edged) Element Stiffness Matrix Evaluation for the Discretization of Deformed Geometries

The case where Ω may be deformed will be considered. After suitable transformations called mapping to the computational domain $\hat{\Omega}$, the preceding methodology developed for the variable-coefficient case can be readily extended to develop a compact formulation of the stiffness matrix in the case of deformed geometries. To highlight the many symmetries in the problem, the results are derived for two dimensional domain (d=2).

There exists an invertible map $x_i(\xi)$ from the physical deformed domain Ω^e to the reference domain Ω_{st} (standard elemental region) for which the Jacobian is nonvanishing and, therefore, of the same sign everywhere on Ω_{st}. (Specifically all vertex angles should be bounded away 0 and 180°). Without loss of generality, it is also assumed that the Jacobian (determinant of the Jacobi matrix) is positive, implying that an element volume in the transformed (mapped) coordinates is positive.

$$J(\xi)=\det\begin{pmatrix}\frac{\partial x_1}{\partial \xi_1} & \cdots & \frac{\partial x_1}{\partial \xi_d}\\ \vdots & & \vdots\\ \frac{\partial x_d}{\partial \xi_1} & \cdots & \frac{\partial x_d}{\partial \xi_d}\end{pmatrix} \tag{2.5.5.1}$$

For 2D problems

$$J_{2D}(\xi_1,\xi_2)=\begin{pmatrix}\frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2}\\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2}\end{pmatrix} \tag{2.5.5.2}$$

Equation (2.21) for q=0 is remembered again like

$$\mathcal{A}(u,v)=\int_{\Omega^e} p(\mathbf{x})\nabla v\cdot\nabla u d\mathbf{x}=\sum_{k=1}^{d}\int_{\Omega^e} p\frac{\partial u}{\partial \xi_k}\frac{\partial v}{\partial \xi_k}d\mathbf{x} \tag{2.45}$$

For d=2, $d\mathbf{x} = dx_1 dx_2$. Going from Ω^e to Ω_{st}, partial derivatives of u are evaluated according to the chain rule

$$\frac{\partial u}{\partial x_k}=\sum_{i=1}^{d}\frac{\partial u}{\partial \xi_i}\frac{\partial \xi_i}{\partial x_k} \qquad \text{k=1,…,d} \tag{2.5.5.3}$$

Combining with a similar expression for v leads to

$$\mathcal{A}(u,v)=\sum_{k=1}^{d}\int_{\Omega_{st}} p\left(\sum_{i=1}^{d}\frac{\partial v}{\partial \xi_i}\frac{\partial \xi_i}{\partial x_k}\right)\left(\sum_{i=1}^{d}\frac{\partial u}{\partial \xi_j}\frac{\partial \xi_j}{\partial x_k}\right)J(\xi)d\xi$$

$$\mathcal{A}(u,v)=\sum_{i=1}^{d}\sum_{j=1}^{d}\int_{\Omega_{st}} p\frac{\partial v}{\partial \xi_i}\left(\sum_{k=1}^{d}\frac{\partial \xi_i}{\partial x_k}\frac{\partial \xi_j}{\partial x_k}J(\xi)\right)\frac{\partial u}{\partial \xi_j}d\boldsymbol{\xi} \tag{2.46}$$

where in the last equation , $d\boldsymbol{\xi} = d\xi_1 d\xi_2$ and the geometric factors associated with the matrics $\frac{\partial \xi_i}{\partial x_k}$, and the jacobian have been assembled within a set of functions

$$\mathcal{G}_{ij}(\xi) = \sum_{k=1}^{d} \frac{\partial \xi_i}{\partial x_k} \frac{\partial \xi_j}{\partial x_k} J(\xi), \; 1 \le i, j \le \mathrm{d} \tag{2.47}$$

It is seen that Equation (2.46) has a form similar to Equation (2.30a). As in that case, it is evaluated by using numerical quadrature on the tensor product of the Gauss-Lobatto grid points, Equation (2.46) is calculated as

$$\mathcal{A}(u,v) = \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{klm} \left[\frac{\partial v}{\partial \xi_i} p \mathcal{G}_{ij} \frac{\partial u}{\partial \xi_j} \right]_{(\xi_k, \xi_l, \xi_m)} w_k w_l w_m \tag{2.48}$$

The coefficient p, geometric terms $\mathcal{G}_{ij}$, and the quadrature weights w can all be conveniently combined into a set of d^2 diagonal matrices, G_{ij}, $i, j \in \{1, ..., d\}^2$. Let

$$\left(G_{ij}\right)_{\hat{k}\hat{k}} := [p\mathcal{G}_{ij}]_{(\xi_k, \xi_l, \xi_m)} w_k w_l w_m \tag{2.49}$$

For 2D and using equation (2.33), Equation (2.49) becomes

$$\left(G_{ij}\right)_{\hat{k}\hat{k}} := diag([p\mathcal{G}_{ij}]_{(\xi_k, \xi_l)})(\hat{M} \otimes \hat{M}) \tag{2.50}$$

with $\hat{k} = 1 + k + (N+1)l + (N+1)^2 m$, $k, l, m \in \{0, ..., N\}^3$, defining a natural ordering of the quadrature points. Note that multiplication of $\underline{u}$ by each G_{ij} simply pointwise multiplication (collocation) of the nodal values u_{klm} with the terms on the right of Equation (2.49) are evaluated, the derivatives in Equation (2.48) are evaluated, for example, as

$$\left.\frac{\partial u}{\partial \xi_1}\right|_{klm} = \sum_{p=0}^{N} \hat{D}_{kp} u_{plm} \qquad k,\ l,\ m \in \{0,\ldots,N\}^3$$
$$= \left(I \otimes I \otimes \hat{D}\right)\underline{u} \tag{2.51a}$$

Defining

$$D_1 := \left(I \otimes I \otimes \hat{D}\right) \qquad D_2 := \left(I \otimes \hat{D} \otimes I\right) \qquad D_3 := \left(\hat{D} \otimes I \otimes I\right) \tag{2.51b}$$

For two dimensional problems

$$D_1 := \left(I \otimes \hat{D}\right) \qquad D_2 := \left(\hat{D} \otimes I\right) \tag{2.51c}$$

When the derivative operators Equation (2.51b) with the geometric factors Equation (2.50) to yield a final compact form, the energy product Equation (2.46) will be

$$\mathcal{A}(u,v) = \underline{v}^T \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix}^T \begin{pmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} \underline{u} \tag{2.52}$$

$$= \underline{v}^T D^T G D \underline{u} \tag{2.52b}$$

Since $G_{ij} = G_{ji}$, only six of the geometric factors need to be computed. The leading-order storage requirement for the factored-stiffness matrix is thus only $6(N+1)^3$, compared with $(N+1)^6$ for the full matrix.

For the case d=2

$$A(u,v) = \underline{v}^T \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}^T \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} \underline{u} \tag{2.52a}$$

$D_1 = I \otimes D$ and $D_2 = D \otimes I$ is used.

The work required for a matrix-product is similarly reduced. In applying $\bar{K} := D^T G D$ to a vector $\underline{u}$, one begins with d tensor-product based derivative evaluations, $\underline{\hat{u}}_j = D_j \underline{u}$, followed by multiplication with the geometric factors $\tilde{u}_i = \sum_j G_{ij} \underline{\hat{u}}_j$, followed finally with a sum across the transposed derivative operators, $\bar{K}\underline{u} = \sum_i D_i^T \tilde{u}_i$ for a total operation count of $12(N+1)^4 + 15(N+1)^3$. This is a significant improvement over the $2(N+1)^6$ cost incurred if the stiffness matrix is computed and stored explicitly.

2.5.6 Mass matrix evaluation for deformed geometries

The extension of the mass matrix to the deformed geometry case is a straight-forward application of quadrature and leads (in R^3) to the diagonal form

$$M_{\hat{\imath}\hat{\imath}} = J(\xi_{1i}, \xi_{2j}, \xi_{3k}) w_i w_j w_k \qquad \hat{\imath} := 1 + i + (N+1)j + (N+1)^2 k \tag{2.53}$$

For two dimensional problem it will be

$$M_{\hat{\imath}\hat{\imath}} = |J_{2D}| \left(\hat{M} \otimes \hat{M} \right) \tag{2.53b}$$

Element load vector evaluation is

$$\mathcal{F}(v) = \underline{v}^T |J_{2D}| \left(\hat{M} \otimes \hat{M} \right) \underline{f} \tag{2.54}$$

For Helmholtz problems, the corresponding Helmholtz operator is created by augmenting the stiffness matrix $\underline{K}$ as

$$\bar{H} := \bar{K} + QM \tag{2.55}$$

where Q is the diagonal matrix corresponding the nodal values $q(\mathbf{x}(\xi_i, \xi_j, \xi_k))$. Although the presence of the variable q and Jacobian degrades the accuracy of the quadrature somewhat, the fact that it is high-order tends to diminish the severity of the "variational crime" [14]. The possibility of using higher-order integration rules to overcome this difficulty, it was better to recover the accuracy by simply increasing *N*.

In this chapter, element stiffness and mass matrices evaluation is discussed for the rectangular and isoparametric quadrilateral elements. Laplacian and Helmholtz operator equations are investigated for the rectangular and isoparametric quadrilateral elements. The evaluation of gradient operator (2D first order differentiation matrix) is also discussed for the rectangular and isoparametric quadrilateral elements in Sections 2.5.4 and 2.5.5.

In Chapter 3, implementation of the equations of Chapter 2 will be discussed.

CHAPTER 3

IMPLEMENTATION

High order finite element method was not famous until the 1980s because increasing the order means increasing the size of the element matrix. This increases the computation time. Reducing the band size of the assembled stiffness matrix has been a problem from the earliest times of the finite element. Therefore, a diagonal mass matrix is the dream of the most finite element people. Gauss Lobatto Legendre (GLL) Lagrangian polynomials and Gauss Lobatto Legendre quadrature is used because GLL Lagrangian polynomials are equal to kronecker delta at GLL nodes. This means identity matrix. If the same polynomials and quadrature is not used, the matrix will be full.

All of the spectral element MATLAB codes can be run both of MATLAB 6.5 and 7.0. Their built-in functions are also used like `mesh.m`. To see the results better, the Matlab 7.0 can be used because its capacity of array editor is larger than Matlab 6.5.

3.1 Poisson Equation Solution with a single rectangular spectral element

Rectangular element (local) stiffness matrix is found by using the Equation (2.28) with the formulas in Section 2.4 of Chapter 2. Rectangular element (local) stiffness matrix is equal to global stiffness matrix as a result local node numbering is the same as global node numbering.

3.1.1 Evaluation of Gauss Lobatto Legendre nodes and weights

`Lobatto.m` file function is used as `[w,x]=Lobatto(N)` to find the Gauss Lobatto Legendre weights and nodes from 1 to -1. The row vector `x` that found is also the roots of the Gauss Lobatto Legendre Lagrangian polynomials. `Lobatto.m` is written by Tarman [8] and is given in Appendix B and C.

3.1.2 Evaluation of mass and 1D stiffness matrices

As mentioned before, $L_i(\xi_j) = \delta_{ij}$, where δ_{ij} is the Kronecker Delta, it is clear that $L_i(\xi_j)$ is equal to `I=eye(N+1);` In Fig. 3.1, GLL polynomials with a degree of N=6 are plotted as an example.

`D=poldif(x,1);` this finds the differentiation matrix. Its component in 1st row and 1st column is derivative of $L_1(\xi_1)$ at x=1($L_1(\xi_1 = 1)$ is equal to 1 at x=1). D(:,p) is the derivative of $L_i(\xi_j)$ at all points between [1,-1] and means the matrix's pth column.. The first collocation point is 1. It is seen from the graph of lagrangian polynomials pth lagrangian polynomial's derivative is equal to 0 at pth collocation point except 1st and last. Example `D=poldif(x,2);` this finds the second order differentiation matrix. The poldif.m belongs to Weideman and Reddy [7].

3.1.2.1 Evaluation of 1D mass matrix

Both of the equation (2.20) and (2.20a) can be used. Equation (2.20a) is implemented as `RHS2=diag(w);` Equation (2.20) implemented as

Mathematical Formulation

$$\hat{M}_{ij} = \sum_{k=0}^{N} w_k L_i(\xi_k) L_j(\xi_k)$$

$$\hat{M} = diag(w_i)$$

Numerical implementation

```
for p=1:N+1
for q=1:N+1
L=I(:,q).*I(:,p);
RHS2(p,q)=dot(w,L);end
end
```

```
RHS2=diag(w);
```

`w` is the w_k and `L` is the $L_i(\xi_k)L_j(\xi_k)$. Here, `w` and `L` are column vectors with same size. For array multiplication ".*" is used. `I(:,q).*I(:,p)` is the entry-by-entry product of `I(:,q)` and `I(:,p)`. The dot product is

```
dot(w,L)=w(1)*L(1)+w(2)*L(2)+...+ w(N+1)*L(N+1)
```

4^{th} GLL Lagrangian polynomial seen in Fig. 3.1 oscillates less than 4^{th} equispaced Lagrangian polynomial at the boundaries.

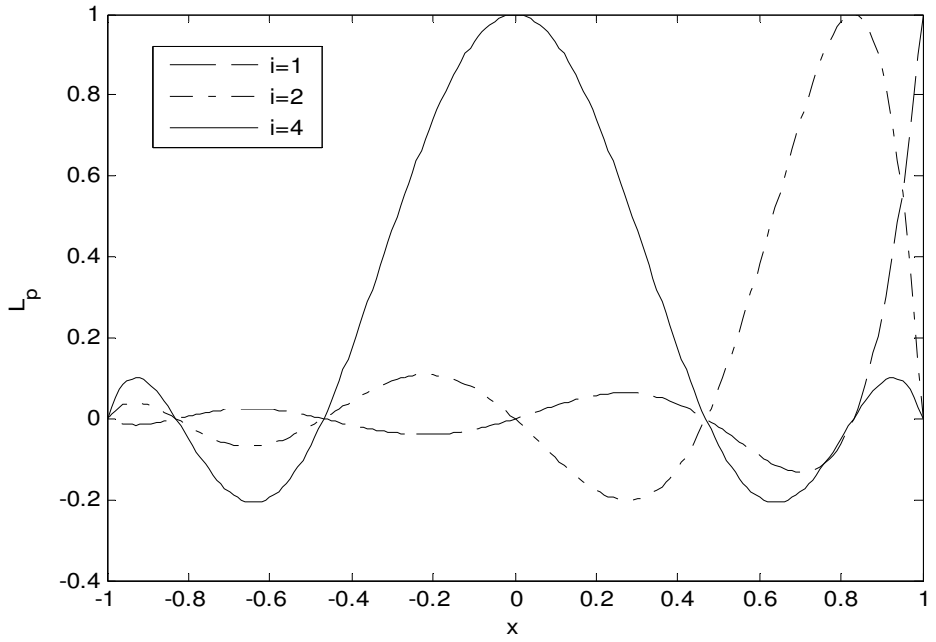


Figure 3.1 The plot of the Gauss Lobatto Legendre Lagrangian polynomials $L_p(\xi)$ (p=1,…, N+1) with a degree of N=6. There are 7 collocation points between 1 and-1. p = i

3.1.2.2 Evaluation of 2D mass matrix

Equation (2.18) implemented as

$$M = \frac{L_1 L_2}{4} \hat{M} \otimes \hat{M} \qquad \texttt{s=kron(RHS2,RHS2);}$$

2D mass matrix is diagonal. Here, the lengths L_1 and L_2 of the standard (master) element are equal to 2.

The kronecker product of two matrices A and B is denoted by $A \otimes B$ and is computed by the matlab built-in function `kron(A,B)`. If A and B have dimensions of $p \times q$ and $r \times s$, then $A \otimes B$ is the matrix of dimension $pr \times qs$ with $p \times q$ block form and is written as [1]

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b & 2a & 2b \\ c & d & 2c & 2d \\ 3a & 3b & 4a & 4b \\ 3c & 3d & 4c & 4d \end{pmatrix} \tag{3.1.1}$$

3.1.2.3 Evaluation of 1D stiffness matrix

Equation (2.24) is used for the evaluation of 1D stiffness matrix. It is implemented as

Mathematical Formulation

$$\hat{K}_{ij} = \sum_{k=0}^{N} w_k \frac{\partial L_i(\xi_{1k})}{\partial \xi_1} \frac{\partial L_j(\xi_{1k})}{\partial \xi_1}$$

Numerical implementation

```
for p=1:N+1
for q=1:N+1
Ae=D(:,q).*D(:,p);
df2(p,q)=dot(w,Ae);end
end
```

The implementation of Equation (2.20) and (2.24) looks same because of GLL quadrature.

3.1.3 Construction of the grid

The master element's local coordinates are calculated by `Lobatto.m`. It has same collocation points in both of the x and y direction for N=24, but having same degree for all directions is not an obligation because of having a single element.

`[xx,yy] = meshgrid(x(1:N+1),y(1:N+1));` returns the local coordinates of each node of the master element in Fig. 3.2.

3.1.3.1 Plotting the grid

The master element grid is plotted by the matlab command `plot(xx,yy,xx',yy')`

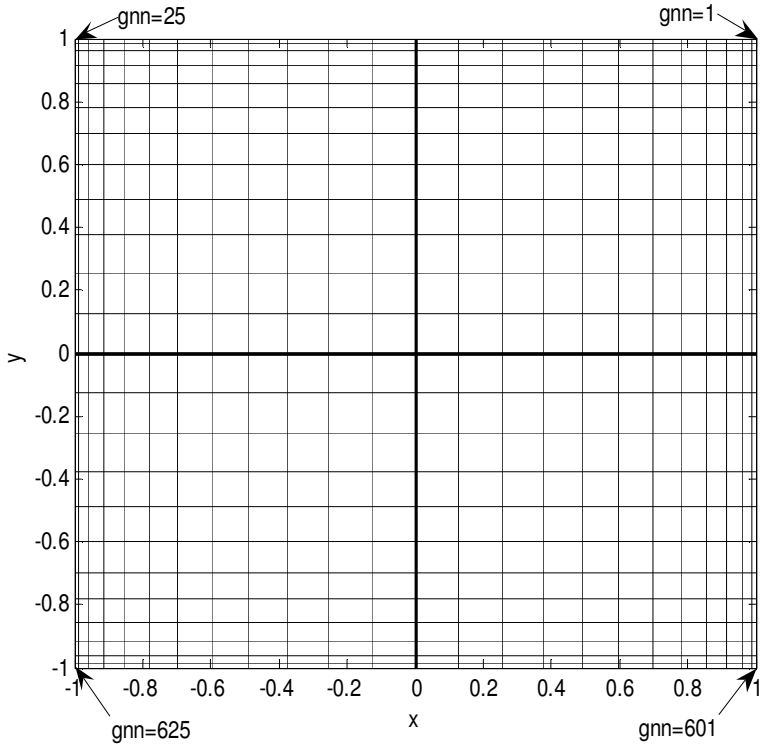


Figure 3.2 The plot of the spectral element grid for a single element with a degree of N=24. The bold lines are also ξ_1 and ξ_2 axis of the reference coordinates. There is a node at each intersection.

The grid becomes denser through the direction from the center of the element to the edges or the vertices of the element.

The distance between the nodes near the edges (boundary) or the vertices of the element (domain) is less than the distance between the nodes near the intersection of the midlines, because of the orthogonality property of GLL polynomials.

3.1.4 Evaluation of load vector

Both of the equation (2.18) and (2.26) are the same except the force function in Equation (2.26) The force function is calculated at `xx = xx(:); yy = yy(:);` which are the vectorized `xx` and `yy` and column vectors with sizes of 625. 2D mass matrix *M* is used for implementation as `s;` Equation (2.26) implemented as

Mathematical Formulation

$$\mathcal{F}(v)=\frac{L_1L_2}{4}\underline{v}^T\left(\hat{M}\otimes\hat{M}\right)\underline{f}$$

Numerical implementation

```
for p=1:(N+1)^2
for q=1:(N+1)^2
if (p==q)
RHS(q)=s(p,q).*f(q);end
end
end
```

3.1.5 Evaluation of steady diffusion operator

1D mass matrix $\hat{M}$ and 1D stiffness matrix $\hat{K}$ has been evaluated for the implementation of Equation (2.28). It is used for the evaluation of steady diffusion operator. Equation (2.28) is implemented as

Mathematical Formulation

$$\mathcal{L}:=p\left[\frac{L_2}{L_1}\left(\hat{M}\otimes\hat{K}\right)+p\frac{L_1}{L_2}\left(\hat{K}\otimes\hat{M}\right)\right]$$

Numerical implementation

```
delta=kron(RHS2,df2)+kron(df2,RHS2);
```

Kron.m file is used to implement the Equation (2.28) to evaluate the steady diffusion operator. 2D Rectangular Element Stiffness matrix equals to the steady diffusion operator for a single element.

3.1.6 Imposing the homogeneous Dirichlet BC

Homogeneous Dirichlet boundary conditions can be implemented for a single rectangular SEM by deleting (discarding) the first and/or last rows of $\hat{K}$ the 1D

stiffness calculated by using Equation (2.24) and $\hat{M}$ the 1D mass matrices evaluated from Equation (2.20). The steady diffusion operator Equation (2.28) is implemented as

```
delta=kron(RHS2,df2(2:N,2:N))+kron(df2(2:N,2:N),RHS2);
```

In Fig. 3.3, the sparsity graph of the calculated `delta` (The steady diffusion operator) is plotted. From Fig. 3.3, `delta` has got a size of 529×529, the `for` loop counters in load vector evaluation are changed to 529. Black regions correspond to non-zero matrix elements as seen in Fig. 3.3. From Fig. 3.3, there are 23805 non-zero (nz) elements in `delta`. An example problem from [1] will be discussed as Equations (4.1) and (4.2) in Chapter 4.

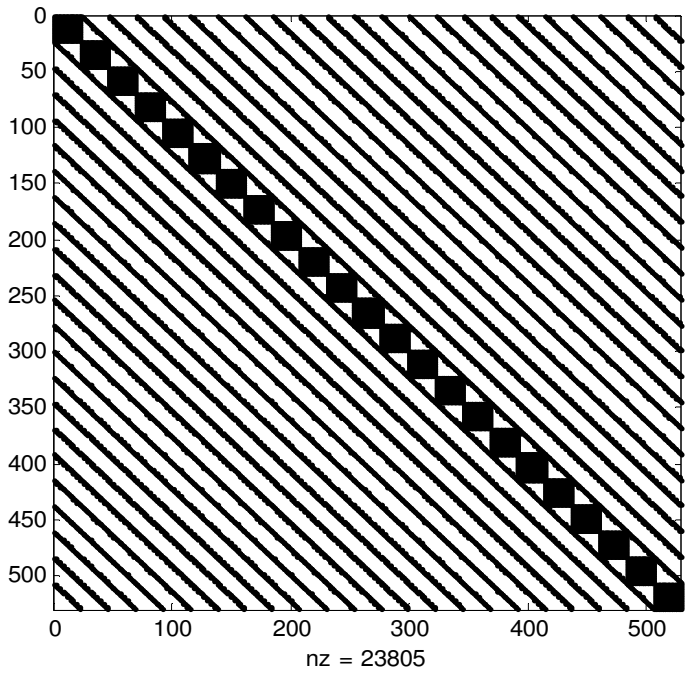


Figure 3.3 Sparsity plot of the element stiffness matrix evaluated with Equation (2.28). No threshold is used because it is kronecker delta

3.1.7 Imposing the nonhomogeneous Dirichlet BC

Poisson Equation with all nonzero Dirichlet boundaries is discussed in Section 4.1.4 with a problem from [5], The global node number of the nodes at the nonhomogeneous Dirichlet boundary are found by the MATLAB command `b=find(abs(xx)==1 | abs(yy)==1);` All components of the related rows defined by `b` is denoted as 0.

```
delta(b,:) = zeros(4*N,(N+1)^2);
```

The related components of related rows are equal to 1.

```
delta(b,b) = eye(4*N);
```

The related components of the column vector `RHS'` is changed with the value of u at the Dirichlet BC. There are 4*N nodes at the Dirichlet boundary.

```
RHS(b)=sin(4*pi*sqrt((xx(b)-2).^2+(yy(b)-2).^2));
```

3.1.8 Imposing the zero and nonzero Neumann BC

Zero Neumann BC's do not need implementation. Adding 0 to the load vector will not cause any change. As it is said in Section 2.1.5.2, natural boundary conditions are included naturally in the solution.

3.1.8.1 Imposing the nonzero Neumann BC

For nonzero Neumann there is another method for only 2D rectangular elements to change the related row of the steady diffusion operator (element stiffness matrix) with related row of the 2D first order differentiation matrix.

```
DNBC = kron(I,D)+kron(D,I);
```

All square element stiffness matrices evaluation and the master element matrix evaluation for the domain spans the area $\{-1 \le x \le 1, -1 \le y \le 1\}$ is done by the same Equations (2.20), (2.24) and (2.28). If the domain isn't [1,-1], for problems in Sections 4.1.2 and 4.1.4, D must be calculated at 1D mapped nodes from [1,-1] to [0, 1]. The differentiation matrix D used for calculation of the steady diffusion operator and 2D first order differentiation matrix must be same.

The related components of the column vector `RHS'` is changed with the value of u at the Neumann BC. Do not add it. . It is a trick from spectral method. [1]

```
RHS(c)=h;
```

Here, c is the node numbers of the Neumann boundaries. For problems in Sections 4.1.2 and 4.1.4, $\mathcal{F}(v)$ will be different from problem in Section 4.1.1, whether if force function is dependent on x,y coordinates or not. $\frac{L_1 L_2}{4}$ must not be used

3.1.9 Evaluation of the u values in Poisson Equations

```
un=-delta\RHS';
```

\ is the Backslash used for matrix left division. For problem 1, delta is a 529×529 matrix and `RHS'` is a column vector with n components, then `un=-delta\RHS';` is the solution to $\mathcal{L}u=\mathcal{F}(v)$ the equation (2.29) computed by Gaussian elimination. A warning message is displayed if delta is badly scaled or nearly singular.

The time used for `un=-delta\RHS';` is 0.09880954 seconds. The time used for `un=-inv(delta)*RHS';` is 0.22994469 sec., which is three times slower than the backslash.

3.1.10 Plotting the 3D graph of the solution

The part of program 16 in [1] is used for plotting the Fig 4.1.

```
uu=reshape(un,N+1,N+1);
[xx,yy] = meshgrid(x,y);
value = uu(N/4+1,N/4+1);
% Interpolate to finer grid and plot:
[xxx,yyy] = meshgrid(-1:.04:1,-1:.04:1);
uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
figure(2), clf,
mesh(xxx,yyy,uuu), colormap(1e-6*[1 1 1]);
xlabel x, ylabel y, zlabel u
```

Here, `interp2.m` is used to calculate the u values on an equispaced grid. `xxx` and `yyy` are the interpolation points where `xx` and `yy` are interpolated.

`uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');` returns matrix uuu containing elements corresponding to the elements of `xxx` and `yyy` and determined by interpolation within the two-dimensional function specified by matrices `xx`, `yy`, and `uuu`. x and y must be monotonic, and have the same format ("plaid") as if they were produced by meshgrid. Matrices xx and yy specify the points at which the data uu is given. Out of range values are returned as NaNs. Cubic interpolation is used, as long as data is uniformly-spaced. Otherwise, this method is the same as 'spline'.

A mesh is drawn as a surface graphics object with the viewpoint specified by view(3). Mesh.m create wireframe parametric surfaces specified by X, Y, and Z, with color specified by C.

`mesh(xxx,yyy,uuu)` draws a wireframe mesh with color determined by uuu as a result color is proportional to surface height. If xxx and yyy are vectors, `length(xxx) = n` and `length(yyy) = m`, where `[m,n] = size(uuu)`. `xxx` and `yyy` are the intersections of the wireframe grid lines.

3.2 Deformed Geometries

This is the situation where the physical domain is different from the computational domain. The formulas used are isoparametric. Same degree of GLL polynomials used to define the coordinate transformation from physical to computational domain. [4] The geometry is represented by the same order quadrilateral elements that are used to approximate the dependent variables.

3.2.1 2D Quadrilateral Element Stiffness matrix evaluation

`Elematrstmquad.m` is given in Appendix D. It is written by using the following equations. If the order of the element is same, the matrix size of the quadrilateral element is the same as the rectangular element

The first row of the program is

```
function[delta,Xesit,Yesit,Jab]=elematrs(N,xna,xnb,xnc,
xnd,yna,ynb,ync,ynd,orderofedge,yeqa,yeqc,xeqd,xeqb)
```

Left hand side is the output of the function. `delta` is the quadrilateral element stiffness matrix. `Xesit,Yesit` are the physical coordinates of the element which are vectorized column vectors. `Jab` is the determinant of the Jacobi matrix (Jacobian). `Jab` will be used to find the mass matrix. Mass matrix is used for the evaluation of the load vector.

Right hand side is the input of the function. `N` is the degree of the element in both of the x and y directions. `xna` and `yna` are the left and down vertices of the element. `xnb` and `ynb` are the right and down vertices. `xnc` and `ync` are the right and up vertices. `xnd` and `ynd` are the left and up vertices of the element. If the order of edge is 0, there is no need to input `yeqa`, `yeqc`, `xeqd` and `xeqb`. Thus, to describe a straight-sided region only the values of the vertex locations are necessary.

3.2.2 Evaluation of the physical coordinates of the nodes at the edges of the element

We expect to be given the shape of each edge in terms of a series of parametric functions, $f^A(\xi_1), f^C(\xi_1), f^D(\xi_2), f^B(\xi_2)$

$$f^A(\xi_1) = yeqa\,,\ f^C(\xi_1) = yeqc\,,\ f^D(\xi_2) = xeqd \text{ and } f^B(\xi_2) = xeqb$$

```
[w, xs] = Lobatto(N); xs = xs'; w = w';
ys = xs; [XS, YS] = meshgrid(xs,ys);
Dxs = poldif(xs,1); Dxs = Dxs(:,:,1); Dys = Dxs;
LXS1 = (1-XS)/2; LXS2 = (1+XS)/2;%  >> xna=0;xnb=0.5;xnc=0.5;xnd=0;
LYS1 = (1-YS)/2; LYS2 = (1+YS)/2;%  >> yna=0;ynb=0;ync=0.5;ynd=0.5;
```

LXS1, LXS2, LYS1 and LYS2 are known as the linear edge functions of the master rectangular element of the spectral element methods. These matrices store the values of the edge functions at the nodes of the master element seen as an example in Fig. 3.2 for N=24. From FEM for mapping, it is also remembered to multiply the nodes with edges that can not be neighbor of the edge the node found. XS and YS are the local Cartesian coordinates of the nodes of the master element.

$$f_i^A(\xi_1) = \sum_{q=0}^{N} f_i^A(\xi_{1,q}) L_q(\xi_1)$$

The physical coordinates of nodes at all edges of the quadrilateral element must be found. As it is said, GLL points are attractive. The parametric representation begins by expressing

$$x_a \le x_i \le x_b \quad \Leftarrow \quad x_i = x_a \frac{(1-\xi_i)}{2} + x_b \frac{(1+\xi_i)}{2} \quad \Rightarrow \quad -1 \le \xi_i \le 1$$

Above Equation is implemented by a modification of [8] as

```
if(orderofedge==0)
```

```
XA = xna*LXS1 + xnb*LXS2; YA = yna; % down face
XC = xnd*LXS1 + xnc*LXS2; YC = ync; % up face
YD = yna*LYS1 + ynd*LYS2; XD = xnd; % left face
YB = ynb*LYS1 + ync*LYS2; XB = xnb; % right face
else
XA = xna*LXS1 + xnb*LXS2; YA = subs(yeqa); % The value in the subs()
XC = xnd*LXS1 + xnc*LXS2; YC = subs(yeqc); % paranthesis can be
YD = yna*LYS1 + ynd*LYS2; XD = subs(xeqd); % a symbolic equation or
YB = ynb*LYS1 + ync*LYS2; XB = subs(xeqb); % a number
end
% XA = 0*LXS1 + 0.5*LXS2; YA = 0;
% XC = 0*LXS1 + 0.5*LXS2; YC = 0.5;
% YD = 0*LYS1 + 0.5*LYS2; XD = 0;
% YB = 0*LYS1 + 0.5*LYS2; XB = 0.5;
```

3.2.3 Evaluation of the physical coordinates of the interior nodes of the element

The sparsity and the accuracy of the local stiffness matrices which are evaluated by elemental mapping for curvilinear Equation (2.40) and straight-sided elements Equation (2.37) are similar for quadrilateral elements with straight sides.

Both of the curvilinear and straight-sided element's local stiffness (element) matrices are evaluated with Equation (2.40).

There are two xna, one is from XA and the other is from XD. So the term xna multiplied by the edges is subtracted in the linear blending function, which xna is not found. For the other coordinates we do the same. The equation (2.40) for linear blending function is implemented with a modification of [8] as

```
X = XA.*((1/2)*(1 - YS)) + XC.*((1/2)*(1 + YS)) + ...
XD.*((1/2)*(1 - XS)) + XB.*((1/2)*(1 + XS)) - ...
(xna)*((1/2)*(1 - XS))  .*   ((1/2)*(1 - YS)) - ...
(xnb)*((1/2)*(1 + XS))  .*   ((1/2)*(1 - YS)) - ...
xnd*((1/2)*(1 - XS))  .*   ((1/2)*(1 + YS)) - ...
(xnc)*((1/2)*(1 + XS))  .*   ((1/2)*(1 + YS));

Y = YA.*((1/2)*(1 - YS)) + YC.*((1/2)*(1 + YS)) + ...
YD.*((1/2)*(1 - XS)) + YB.*((1/2)*(1 + XS)) - ...
```

```
(yna)*((1/2)*(1 - XS))  .*   ((1/2)*(1 - YS)) - ...
(ynb)*((1/2)*(1 + XS))  .*   ((1/2)*(1 - YS)) - ...
(ynd)*((1/2)*(1 - XS))  .*   ((1/2)*(1 + YS)) - ...
(ync)*((1/2)*(1 + XS))  .*   ((1/2)*(1 + YS));
```

After finding the coordinates of the element nodes, all the partial derivatives can be evaluated, which is required to determine the Jacobian. To find the laplacian operator $\frac{\partial \xi_1}{\partial x_1}$, $\frac{\partial \xi_2}{\partial x_1}$, $\frac{\partial \xi_1}{\partial x_2}$ and $\frac{\partial \xi_2}{\partial x_2}$ must be found.

3.2.4 Evaluation of the Jacobian

Following equation is used

$$J = \begin{vmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} \end{vmatrix} = \frac{\partial x_1}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_1}$$

It is implemented in [8] as

```
J = (Dxs * X')' .* (Dys * Y) - (Dys * X) .* (Dxs * Y')';
```

3.2.5 Benchmarks

Jacobi matrix is used to find the local stiffness matrix of the quadrilateral element. To control its accuracy, the exact area of the element must be found by using the determinant of the Jacobi matrix.

$$A = \int_{-1}^{1}\int_{-1}^{1} J(r)drds$$

Here, the Jacobian J(r) is the determinant of Jacobi matrix. It is implemented in [8] as

```
Area = w' * J * w
```

Dividing rectangular domain to rectangular elements is not an obligation. Changing size or the type of the element like quadrilateral element is a way to achieve or obtain better results. Rectangular domain is divided to complex elements to control the success of the Jacobi mapping.

Example problem is solved from [4]. For the physical domain seen in Fig. 3.4, the equations of the edges are `yeqa=-XA/5; yeqc = 1/3*XC+11/3; xeqd = YD/4; xeqb =(29-YB)/6;` Subs will be input XA,XC,YD and YB values to find YA,YC,XD and XB. The x coordinates of the vertices are xna=0, xnb=5, xnc=4 and xnd=1. The y coordinates of the vertices are yna=0, ynb=-1, ync=5 and ynd=4. Jacobian is calculated for N=6. The element has 49 nodes. The area calculated by Jacobian is equal to 20. The area calculated by geometry is equal to 20 as a result the accuracy of Jacobian has been controlled. The element is seen in Fig. 3.4.

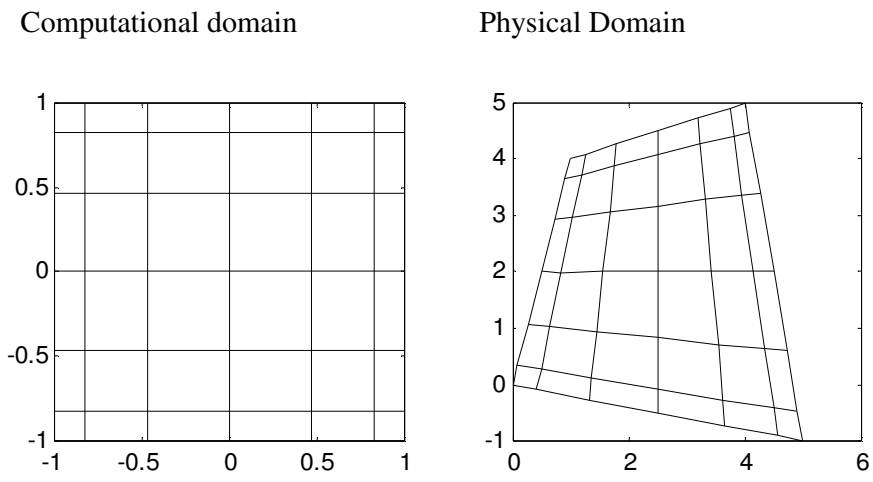


Figure 3.4 The grid of coordinate transformation from physical domain to computational domain for an element of N=6

3.2.6 The implementation of the Jacobian matrix

Mathematical Formulation

$$J_{2D}(\xi_1,\xi_2)=\begin{vmatrix}\frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2}\\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2}\end{vmatrix}$$

$$\frac{\partial x_1}{\partial \xi_1}$$

$$\frac{\partial x_1}{\partial \xi_2},\ \frac{\partial x_2}{\partial \xi_1}\ \text{and}\ \frac{\partial x_2}{\partial \xi_2}$$

Numerical implementation

```
Jlast=[(Dxs * X')' (Dys * X);
       (Dxs * Y')' (Dys * Y)];

J11y=Jlast(1:(N+1),1:(N+1))';

J12y=Jlast(1:(N+1),N+2:2*N+2)';
J21y=Jlast(N+2:2*N+2,1:(N+1))';
J22y=Jlast(N+2:2*N+2,N+2:2*N+2)';
```

3.2.7 Evaluation of the geometric terms

The inverse of the Jacobi matrix must be found to evaluate the geometric terms in Equation (2.47).

Mathematical Formulation

$$\frac{1}{|J_{2D}|}\begin{bmatrix}\frac{\partial x_2}{\partial \xi_2} & -\frac{\partial x_1}{\partial \xi_2}\\ -\frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_1}\end{bmatrix}$$

$$\frac{\partial \xi_1}{\partial x_1},\ \frac{\partial \xi_1}{\partial x_2},\ \frac{\partial \xi_2}{\partial x_1}\ \text{and}\ \frac{\partial \xi_2}{\partial x_2}$$

Numerical implementation

```
Jinv=[J22y(:)./Jab(:) -J12y(:)./Jab(:);
      -J21y(:)./Jab(:) J11y(:)./Jab(:)];

J11inv=Jinv(1:(N+1)^2,1);
J12inv=Jinv(1:(N+1)^2,2);
J21inv=Jinv((N+1)^2+1:2*(N+1)^2,1);
J22inv=Jinv((N+1)^2+1:2*(N+1)^2,2);
```

J12y and J21y or J12inv and J21inv are not equal to each other for quadrilaterals. All of them must be calculated.

The geometric terms Equation (2.47) can be written for two dimensional elements as

$$\mathcal{G}_{ij}(\xi)=\sum_{k=1}^{d}\frac{\partial\xi_i}{\partial x_k}\frac{\partial\xi_j}{\partial x_k}J(\xi)=\mathcal{G}_{11}+\mathcal{G}_{12}+\mathcal{G}_{21}+\mathcal{G}_{22} \tag{3.2.7.1}$$

Mathematical Formulation	Numerical implementation
$\mathcal{G}_{11}(\xi)=\frac{\partial\xi_1}{\partial x_1}\frac{\partial\xi_1}{\partial x_1}J(\xi)+\frac{\partial\xi_1}{\partial x_2}\frac{\partial\xi_1}{\partial x_2}J(\xi)$	`Gk11=J11inv.*J11inv.*Jab(:)` `+J12inv.*J12inv.*Jab(:);`
$\mathcal{G}_{12}(\xi)=\frac{\partial\xi_1}{\partial x_1}\frac{\partial\xi_2}{\partial x_1}J(\xi)+\frac{\partial\xi_1}{\partial x_2}\frac{\partial\xi_2}{\partial x_2}J(\xi)$	`Gk12=J11inv.*J21inv.*Jab(:)` `+J12inv.*J22inv.*Jab(:);`
$\mathcal{G}_{21}(\xi)=\frac{\partial\xi_2}{\partial x_1}\frac{\partial\xi_1}{\partial x_1}J(\xi)+\frac{\partial\xi_2}{\partial x_2}\frac{\partial\xi_1}{\partial x_2}J(\xi)$	`Gk21=J21inv.*J11inv.*Jab(:)` `+J22inv.*J12inv.*Jab(:);`
$\mathcal{G}_{21}(\xi)=\frac{\partial\xi_2}{\partial x_1}\frac{\partial\xi_2}{\partial x_1}J(\xi)+\frac{\partial\xi_2}{\partial x_2}\frac{\partial\xi_2}{\partial x_2}J(\xi)$	`Gk22=J21inv.*J21inv.*Jab(:)` `+J22inv.*J22inv.*Jab(:);`

3.2.8 Evaluation of the geometric factor

$(\hat{M}\otimes\hat{M})$ must be evaluated.

```
for p=1:N+1
for q=1:N+1
L=I(:,q).*I(:,p);
RHS2(p,q)=dot(w1,L);end % RHS2 is equal to 1D mass matrix
end
s=sparse(kron(RHS2,RHS2));
```

As it was said before, $\mathcal{G}_{11}$, $\mathcal{G}_{12}$, $\mathcal{G}_{21}$, and $\mathcal{G}_{22}$ are vectors. Therefore, they must be diagonalized to be a diagonal matrix. Equation (2.50) is implemented as

$$\left(G_{ij}\right)_{\hat{k}\hat{k}}:=diag([pG_{ij}]_{(\xi_k,\xi_l)})(\hat{M}\otimes\hat{M})$$

```
G11=sparse(s*diag(Gk11));
G12=sparse(s*diag(Gk12));
G22=sparse(s*diag(Gk22));
G21=sparse(s*diag(Gk21));
```

Since G12 and G21 are equal to each other, G11, G12 and G22 are needed to be evaluated. To dictate like `G12=G21;` is enough.

3.2.9 Evaluation of D_{ξ_1} and D_{ξ_2}

The master element has same degree in the both of the x and y direction. Thus, D_{ξ_1} and D_{ξ_2} are same. Equation (2.51c) is implemented as

Mathematical Formulation

$D_1 := \left(I \otimes \hat{D}\right)$ and $D_2 := \left(\hat{D} \otimes I\right)$

Numerical implementation

```
D1=sparse(kron(I,D));
D2=sparse(kron(D,I));
```

Sparse matrices are a special class of matrices that contain a significant number of zero-valued elements. This property allows MATLAB to store only the nonzero elements of the matrix, together with their indices. It reduces computation time by eliminating operations on zero elements.

Global stiffness matrix memory usage is 1850.9 KB. By the matlab command of Stiff2=sparse(stiff2); the memory used is reduced to 678.5 KB. Time consumed for matrix left division is reduced from 0.05093(full) to 0.02878 seconds with sparse. Without sparse CPU usage for left division has reached to a peak value of %50.

The result of assembly or multiplication of sparse matrices is a normal matrix. Using sparse for normal matrices is needless memory usage. For a 169×169 matrix, memory usage of sparse is 343.4 KB and the normal one uses 228.5 KB.

3.2.10 Evaluation of the 2D quadrilateral element stiffness matrix

Equation (2.52a) is implemented as

$$\begin{pmatrix} D_1 \\ D_2 \end{pmatrix}^T \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}$$

```
DekT=[D1;D2]';
G=[G11 G12;G21 G22];
delta=DekT*G*[D1;D2];
```

3.2.11 Evaluation of the element load vector

As mentioned before, the mass matrix for the quadrilateral element must be found. To have an element load vector, the element mass matrix is needed to be evaluated. The equation (2.54) is implemented as

$$\mathcal{F}(v) = \underline{v}^T \left|J_{2D}\right| \left(\hat{M} \otimes \hat{M}\right) \underline{f}$$

```
for p=1:(N+1)^2
for q=1:(N+1)^2
if (p==q)
RHS(q)=Jab(p).*s(p,q).*f3(q);end
end
end
```

3.3 Poisson Equation Solution with more elements

Problem in Section 4.1.4 will be solved with more than one element. The domain $\Omega \in [0,1]\times[0,1]$ is sliced to four elements and the element at the left and up corner is cut away. The solution domain is seen in Fig. 3.5. The global stiffness matrix is the steady diffusion operator which is the result of the assembly. Connectivity matrix is required to assemble the local stiffness matrices.

3.3.1 Connectivity Matrix Generation

Elnode.m is the code which generates a connectivity matrix for Fig. 3.5. The connectivity matrix generated for N=2 is seen in Table 3.1. In the connectivity matrix pth row is the node numbers of the pth element. Gauss Lobatto Legendre points evaluated with Lobatto.m are from 1 to -1, so the order of the local node numbering begins from 1stnode at the highest right corner, goes left through the line, and passes the node under the 1stnode, last node is the node at the lowest left corner. This is the order of local node numbering.

For global node numbering the highest right element is numbered first, so global node numbers and local node numbers of the first element are identically same. The order of the global node numbering is as same as the local node numbering.

The node numbers of the first element nodes on the joint lower edge of the first is same as the upper edge of the second element, the global node numbering continues from the node under the local 1st node. By numbering the second, the left edge of the second and the right edge of the third element have same global node numbers, so the global node numbering of third element is done in the same order without numbering the right edge as seen in Fig.3.5.

Table 3.1 Global node numbering of Mesh 1 for N=2

Local Node numbering	1	2	3	4	5	6	7	8	9
Element 1	1	2	3	4	5	6	7	8	9
Element 2	7	8	9	10	11	12	13	14	15
Element 3	9	16	17	12	18	19	15	20	21

Numbering the edge nodes first can help to reduce the band size of the assembled stiffness matrix like Karniadakis and Sherwin [2]. The program elnode.m is N independent so it doesn't need modifications for different N's. N is the only input and the matrix of the global node numbers of the elements is the only output. For different mesh types, different mesh generators must be used or written. Elematrstmquad.m program in Appendix D gives also the coordinates of x and y of the element's nodes and also the determinant of the Jacobi matrix to evaluate the load vector (RHS) of the problem. In Table 3.1, the output of `elnod=elnode(N);` is seen for N=2.

3.3.2 Assembly of spectral element stiffness matrix

Assemblies of the 2D rectangular spectral element and the 2D quadrilateral spectral element's local stiffness matrices are all same because they must have the same size if they are at the same order.

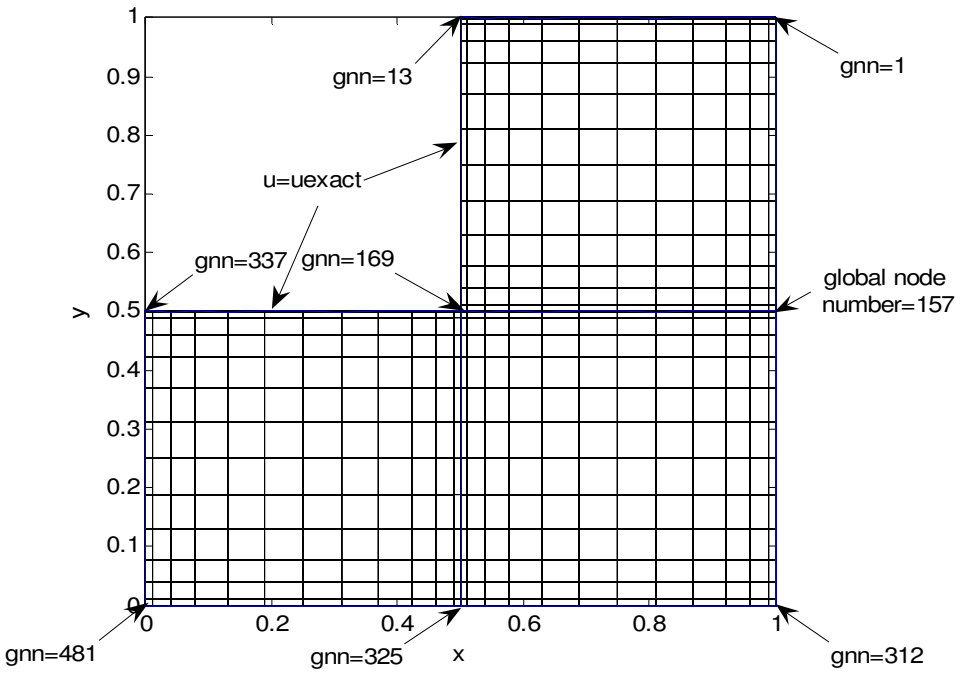


Figure 3.5 The grid of the Mesh 1 made up of three elements with a degree of 12. The global node numbers of the element's vertices are seen.

Local stiffness matrix of the curvilinear quadrilateral element is full matrix. Using curvilinear elements increase the band size and density of the global stiffness matrix as seen in Figures 3.11 and 3.13.

3.3.2.1 Evaluation of the Steady Diffusion operator

Steady diffusion operator is the global stiffness matrix. It is the result of the assembly of the local stiffness matrix. The program that is written to assemble the local stiffness matrices to form the global stiffness matrix

```
for i=1:(N+1)^2
for j=1:(N+1)^2
if(abs(delta(i,j))>=1e-15)
stiff2(elnod(e,i),elnod(e,j))=stiff2(elnod(e,i), ...
elnod(e,j))+delta(i,j);
```

```
end
end
end
```

The global stiffness matrix is stiff2. Here, `1e-15` within `if` statement is the threshold value; which is used as computation zero by the definition. It is used with if statement to decide which component of the element stiffness matrix is zero. If a component of the element stiffness matrix is smaller than the threshold value, it will be assumed as zero and won't be assembled into the global stiffness matrix. Sparsity plot of the Mesh1's elements is seen for the threshold value of 1.0×10^{-15} in the following Fig. 3.6. The white areas are the zero components of the element stiffness matrix.

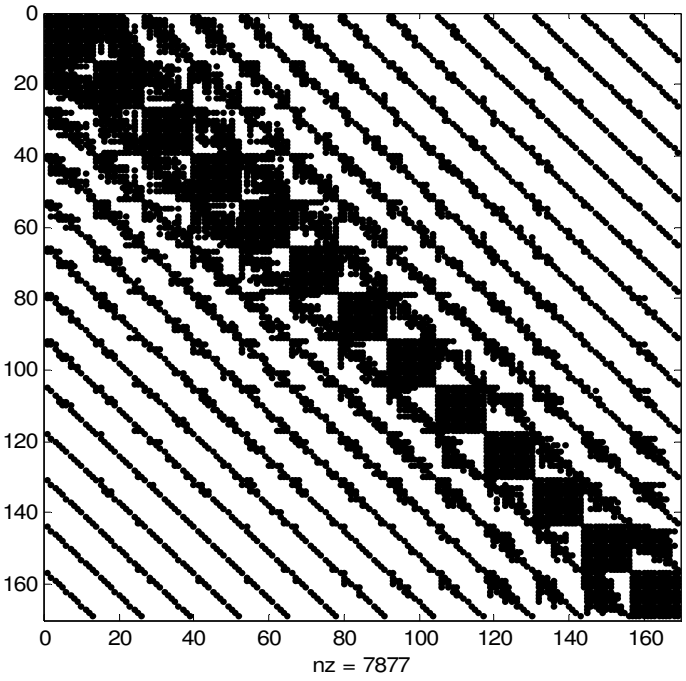


Figure 3.6 Sparsity plot of the element stiffness matrix for N=12. Computation zero is defined as 1.0×10^{-15}.

Element stiffness matrices calculated for rectangular elements with straight sided quadrilateral element formula are as sparse as quadrilateral element formula. Both of

them needed threshold value to be like in Fig. 3.3. The diagonal lines near the main band-diagonal are not clearly seen in Fig 3.6. As it is seen, the sparsity plot of the rectangular element's local stiffness matrix is not like Fig. 3.3. Both of them are square elements, as a result they must have same sparsity. So a threshold value will be 1.0×10^{-13}. Here, the local stiffness matrix is seen for Fig. 3.7.

For Fig. 3.7, the diagonal lines and the square matrices in the main band-diagonal are clearly seen. Contrary to Fig.3.6, all of the diagonal lines are clearly seen in Fig. 3.7.

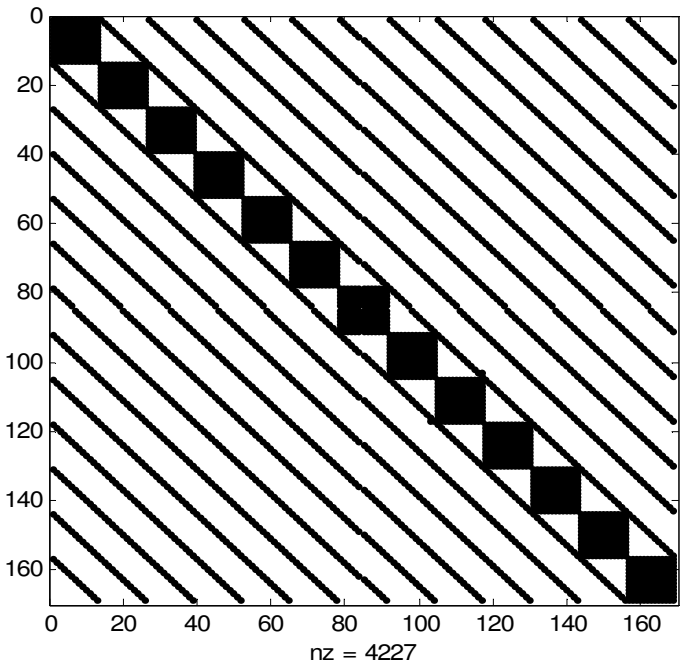


Figure 3.7 Sparsity plot of the element stiffness matrix for N=12. Computation zero is defined as 1.0×10^{-13}.

The sparsity of the global stiffness matrices evaluated for both of the threshold values of 1.0×10^{-13} and 1.0×10^{-15} is plotted in Fig. 3.8 and Fig. 3.9. Choosing a threshold value of 1.0×10^{-13} decrease the band size and density, as a result matrix is sparser and narrower. This means economic usage of implementation time and memory.

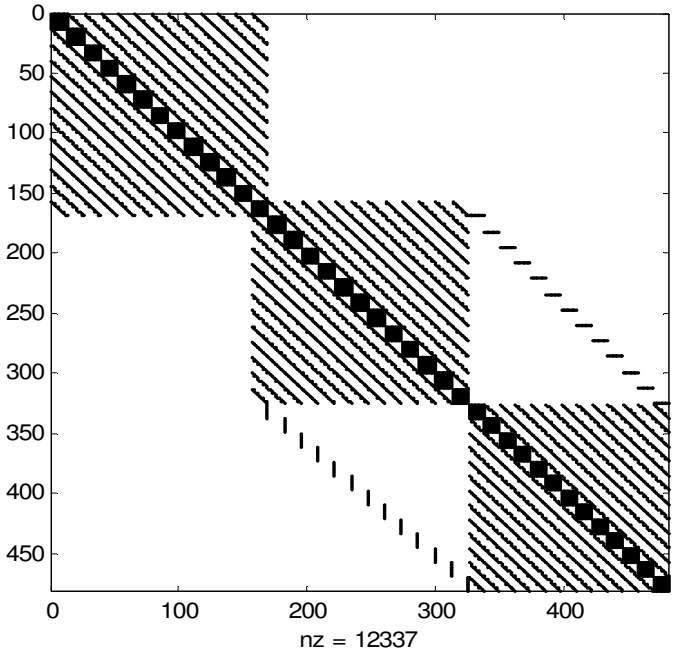


Figure 3.8 Sparsity plot of the global (assembled) stiffness matrix for N=12. Computation zero is defined as 1.0×10^{-13}.

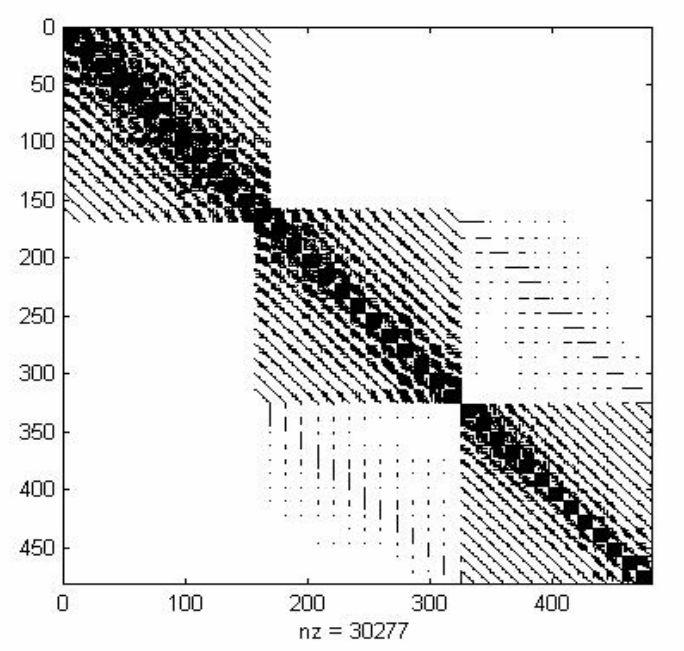


Figure 3.9 Sparsity plot of the global (assembled) stiffness matrix for N=12. Computation zero is defined as 1.0×10^{-15}.

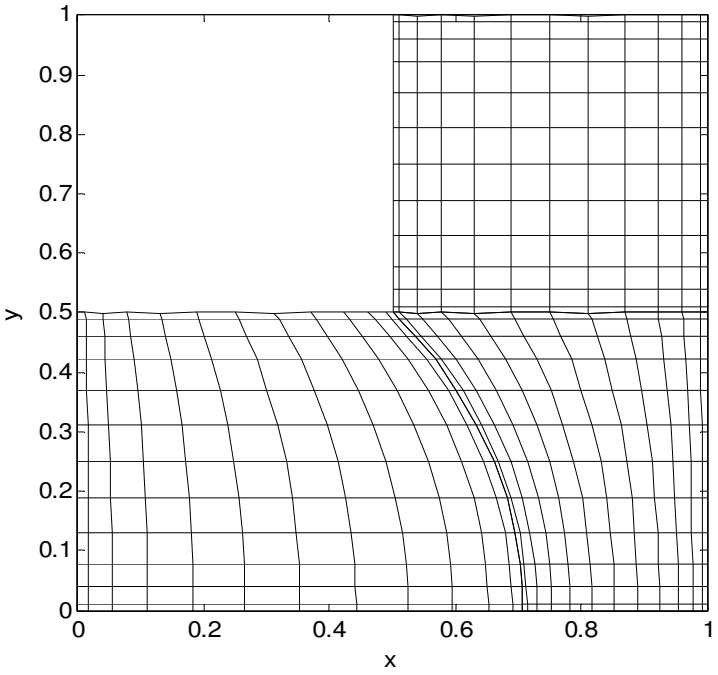


Figure 3.10 The grid of the Mesh 2 made up of a square and two quadrilateral elements with a degree of 12.

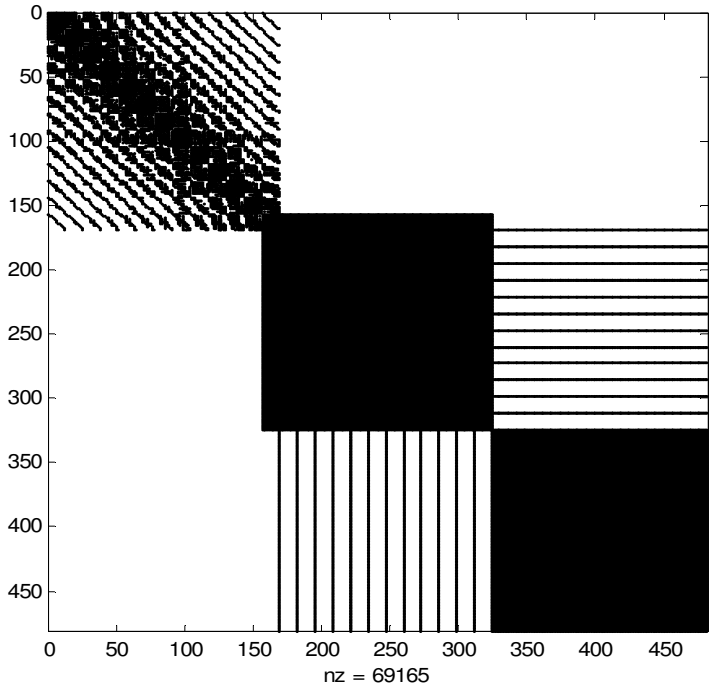


Figure 3.11 Sparsity plot of the global (assembled in Fig. 3.10) stiffness matrix for N=12. Computation zero is defined as 1.0×10^{-15}.

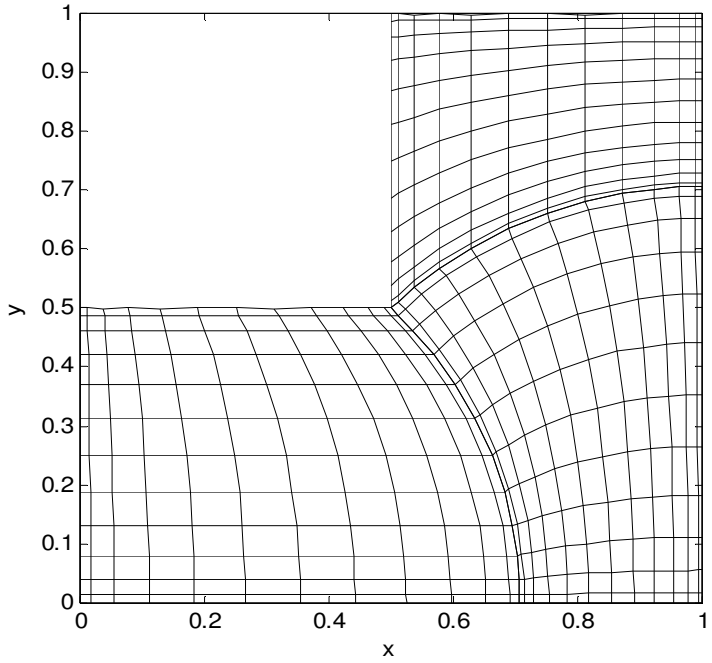


Figure 3.12 The grid of the Mesh 3 made up of a three quadrilateral elements with a degree of 12.

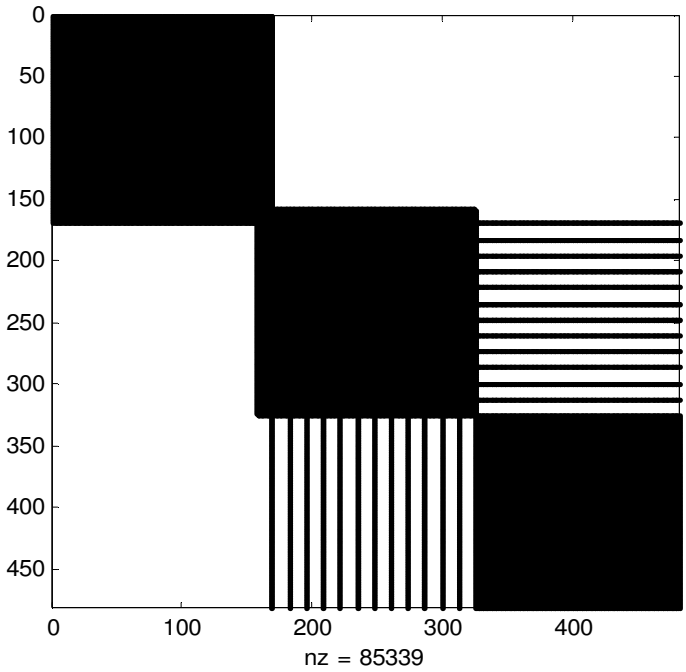


Figure 3.13 Sparsity plot of the global stiffness matrix of the mesh in Fig. 3.12 for N=12. Computation zero is defined as 1.0×10^{-13}.

In Fig. 3.10, right face of the 3rd element is an arc drawn at the center of (0, 0) with a radius of $\sqrt{2}/2$. In Fig. 3.12, down face of the 1st element is an arc drawn at the center of (1, 1) with a radius of $\sqrt{2}/2$.

3.3.3 Evaluation of the global load vector

The program part written looks like the program part written for global stiffness matrix.

```
for j=1:(N+1)^2 % j can be i, it will be better
load(elnod(e,j))=load(elnod(e,j))+RHS(j);
end
```

Now, Global stiffness matrix and load vector is evaluated. Imposing the boundary conditions will be our next issue. For imposing the boundary conditions, the global node numbers of the element's nodes at the domain boundaries must be found.

3.3.4 Imposing the Nonhomogeneous Dirichlet Boundary Conditions

The local node numbers of the element's Dirichlet boundary nodes must be found. Poisson Equation (4.4) in Section 4.1.4 is solved on mesh 1 seen in Fig. 3.5. From Fig 3.5, up, right and left face of the first element is Dirichlet boundaries. Right and down faces of the second element and up, down and left faces of the third element are Dirichlet boundaries. The Dirichlet boundaries is found like that

```
nodirichel1es = find(abs(yy)==1 | xx==1); % 1st element
nodirichel2es = find(xx==-1 | yy==1); % 2nd element
nodirichel3es = find(abs(xx)==1 | yy==-1); % 3rd element
```

Their global node numbers will be

```
for i=1:3*N+1
assemel3(i)=elnod(3,nodirichel3es(i));
end
```

```
for i=1:3*N+1
assemel1(i)=elnod(1,nodirichel1es(i));
end
for i=1:2*N+1 % Shows how many nodes are on the Dirichlet
% boundaries
assemel2(i)=elnod(2,nodirichel2es(i));
end
```

The Dirichlet boundary conditions can be imposed like

```
stiff2(assemel2,:) = zeros(2*N+1,3*(N+1)^2-2*(N+1));
stiff2(assemel2,assemel2) = eye(2*N+1);
stiff2(assemel1,:) = zeros(3*N+1,3*(N+1)^2-2*(N+1));
stiff2(assemel1,assemel1) = eye(3*N+1);
stiff2(assemel3,:) = zeros(3*N+1,3*(N+1)^2-2*(N+1));
stiff2(assemel3,assemel3) = eye(3*N+1);
load(assemel1)=uex(nodirichel1es,1);
load(assemel2)=uex(nodirichel2es,2);
load(assemel3)=uex(nodirichel3es,3);
```

Since the domain boundaries are the same, these implementation features of imposing Dirichlet boundaries are the same for both of the meshes in Fig. 3.10 and Fig. 3.12

3.3.5 Imposing the Nonhomogeneous Neumann Boundary Conditions

The example problem will be inviscid flow around a cylinder from [4]. The irrotational flow of an ideal fluid about a circular cylinder, placed with its axis perpendicular to the plane of the flow between two long horizontal walls.

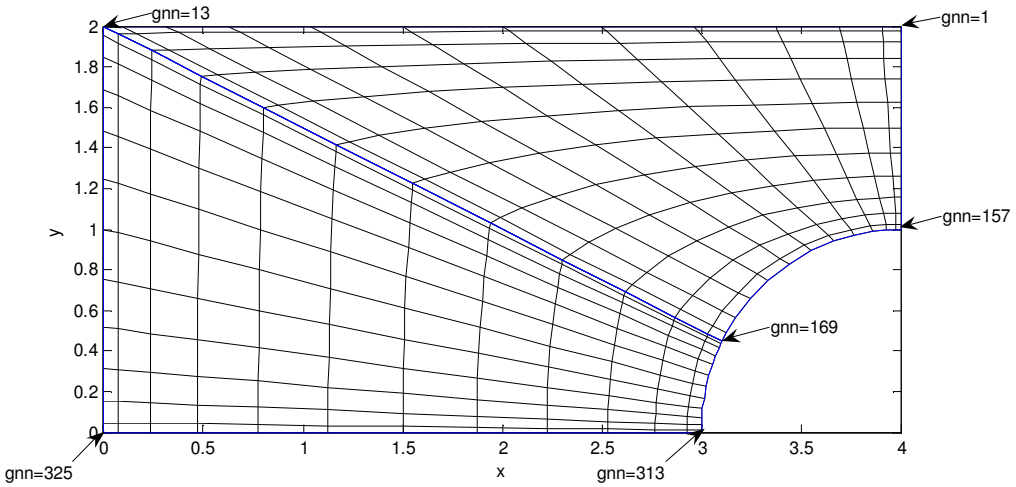


Figure 3.14 The mesh of the inviscid flow around a cylinder drawn on the upper right quadrant of the domain.

The mesh is seen in Fig. 3.14. The domain is divided from the diagonal between (0,2) and (4,0). The node A is the 169th node, not the 1st node. A, B, C, and D are defined through counter clockwise as FEM in Fig. 2.1. As discussed before 1st node is upper and most right so 1st element is the upper and the most right in the figure the mesh and global node numbers are seen in Table 3.2. Blue lines are element boundaries.
The equations of the element's edges are

```
yeqc = [2 -1/2*XC+2];
xeqd = [4-2*YD 0];
xeqb = [4 -sqrt(1-YB.^2)+4];
yeqa = [sqrt(1-(XA-4).^2) 0];
```

The left edge of the 1st element is adjacent to the up edge of the 2nd element. Because of that, the global numbers of the nodes on them are the same. The mesh generation is done for global node numbering will be handled differently. The connectivity matrix generated for N=2 is seen in Table 3.2.

Table 3.2 Global node numbering of Mesh seen in Fig. 3.14 for N=2

Local Node numbering	1	2	3	4	5	6	7	8	9
Element 1	1	2	3	4	5	6	7	8	9
Element 2	9	6	3	10	11	12	13	14	15

From Figure 4.15, it is seen that the left face of domain or 2nd element is nonhomogeneous Neumann boundary. It is a linear edge, but its 2D first order differentiation matrix can not be evaluated with the formulas for rectangular elements. Therefore, there are two choices. Evaluation of the surface Jacobian is done and it is added to the related rows of the related element's load vector or assembled load vector [2]. The related row should not be changed.

The second choice is the evaluation of the 2D first order differentiation matrix for quadrilateral element. This is discussed in following Section 3.3.5.1.

3.3.5.1 Differentiation within an elemental region

The second choice is the evaluation of the 2D first order differentiation matrix for quadrilateral element and replacing it into the element stiffness matrix. Equation (2.42) is implemented as

```
Dksi=diag(J11inv(:))*kron(I,Dxs)+diag(J21inv(:))*kron(Dxs,I);
```

The local and global node numbers of the nodes on the nonzero Neumann boundary (2nd element's left face seen in Fig. 3.14) is found as

```
localnodenum2ndelleft=find(yy==-1 ); %
for i=1:N+1
globalnodenum2ndelleft(i)=elnod(2,localnodenum2ndelleft(i));
end
```

Then it is replaced in the 2nd element stiffness matrix's related rows stored by `localnodenum2ndelleft`. It is implemented as

```
for i=1:13
delta(localnodenum2ndelleft,:) = Dksi(localnodenum2ndelleft,:);
end
```

After the Dirichlet boundary conditions have already been imposed. The sparsity plot of the global stiffness matrix is seen in Fig. 3.15.

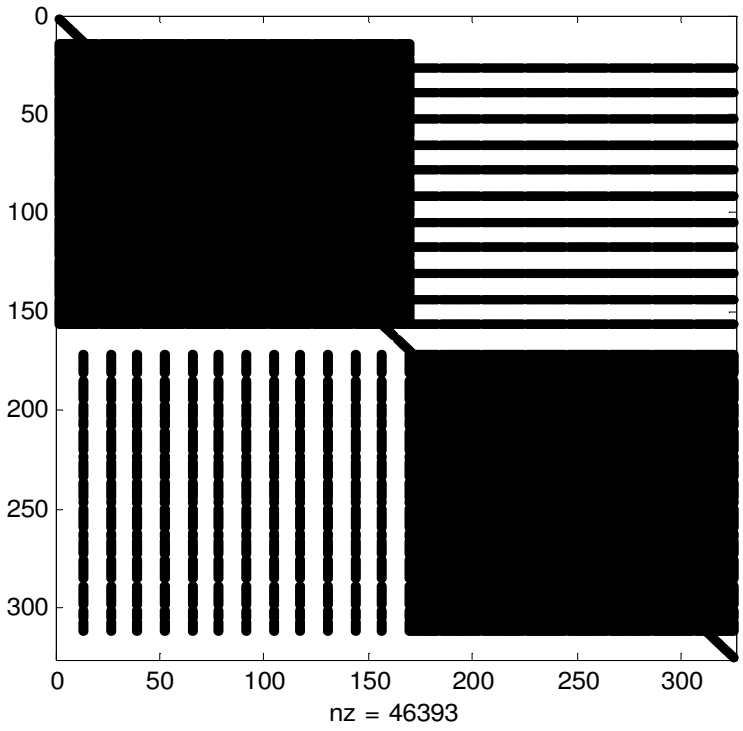


Figure 3.15 Sparsity plot of the same assembled (global) stiffness matrix that imposed all of the boundary conditions for the mesh of the stream function seen in Fig. 3.14.

3.4 Evaluation of the Helmholtz operator

Element stiffness matrices and mass matrices have been evaluated for quadrilateral and rectangular element. Mass matrix and Helmholtz operator of the quadrilateral element is evaluated with Equations (2.54) and (2.55). They are implemented as

```
for p=1:(N+1)^2
for q=1:(N+1)^2
if (p==q) MwJ(p,q)=1*Jab(p).*s(p,q);end
end
end
delta=delta+MwJ;
```

In this chapter, the implementation of spectral element method for the Poisson and Helmholtz equations using MATLAB on rectangular and deformed domains discretised with a single and multi-element are discussed including the element stiffness and mass matrices evaluation. Imposing the homogeneous or non-homogeneous Dirichlet and Neumann boundary conditions into the global stiffness matrix and load vector are also discussed for domains discretised with rectangular and isoparametric quadrilaterals. The global node numbering of the elements is discussed in Connectivity matrix generation. Assembly (Direct Stiffness Summation) of element stiffness matrices to form global element stiffness matrix is implemented. Assembly of element load vectors to form global load vector is implemented. The implementation procedure for the SEM solution with two quadrilateral elements to the inviscid flow around a cylinder is discussed for velocity potential equation including nonhomogeneous Neumann boundary conditions. Helmholtz operator of quadrilateral elements is implemented for the solution of Helmholtz.

The `elematrstmquad.m` written for the evaluation of isoparametric quadrilateral element stiffness matrix is first verified by evaluating rectangular element stiffness matrix.

CHAPTER 4

RESULTS AND DISCUSSIONS

Poisson and Helmholtz equations are solved with single and multi-elements including both of the rectangular and isoparametric quadrilaterals. Inviscid flow around a cylinder is also solved. Spectral element results will be compared with the the exact results to observe and appreciate the accuracy of the approach. Also, Single spectral element results will be compared with finite element or spectral collocation (pseudospectral) solutions.

4.1 Poisson Equation Solution with a single rectangular spectral element

Four different Poisson Equations including different types of boundary conditions (such as Dirichlet (zero or non-zero), Neumann and Robin) are solved.

4.1.1 Poisson equation with all zero Dirichlet boundaries

Our first program's purpose is to solve the Poisson Equation below

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=10\times\sin(8x\times(y-1)) \qquad \text{on} \qquad \Omega\in[-1,1]\times[-1,1] \tag{4.1}$$

$$u(-1, y) = 0, u(1, y) = 0 \tag{4.2}$$

$$u(x, -1)= 0, u(x, 1) = 0 \text{ on the Dirichlet(essential) boundaries} \tag{4.2a}$$

The problem is taken from [1]. It is on a rectangular domain. From Fig 3.2, the grid's middle part is sparser than equispaced grid, because the nodes are collocated near the boundaries as a feature of the roots of the orthogonal polynomials. For plotting, the problem is interpolated to more equispaced points. (Section 3.1.10)

Domain with a single rectangular spectral element is a problem that element stiffness matrix equals to the assembled stiffness matrix. Spectral method is used as benchmark for some problems especially domain with a single rectangular spectral element. Both of the methods have got a degree of N=24. For domains $x \in [1,-1]$ and $y \in [1,-1]$ the solution is calculated at same collocation points for both of the methods. The results must be same for same collocation points (Gauss Lobatto Legendre points). For N=24 degree of polynomials 625 collocation points are on the solution domain, 96 of the collocation points belong to the zero Dirichlet boundaries. U is calculated at 529 collocation points. The results are same up to 14 decimal digits.

If an even number N=24 was not chosen as the degree of the method, there wouldn't be any nodes (collocation points) on the midlines of the domain which are x=0 and y=0.

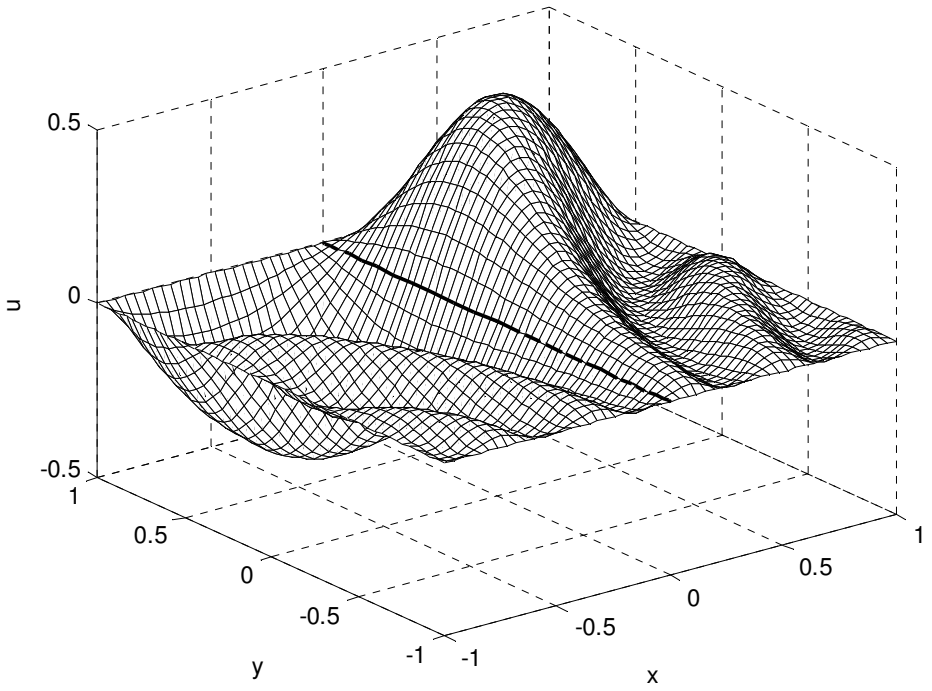


Figure 4.1 3D Solution graph of (4.1) for N=24 with a single rectangular spectral element. The bold line is u(0,y)=0 line.

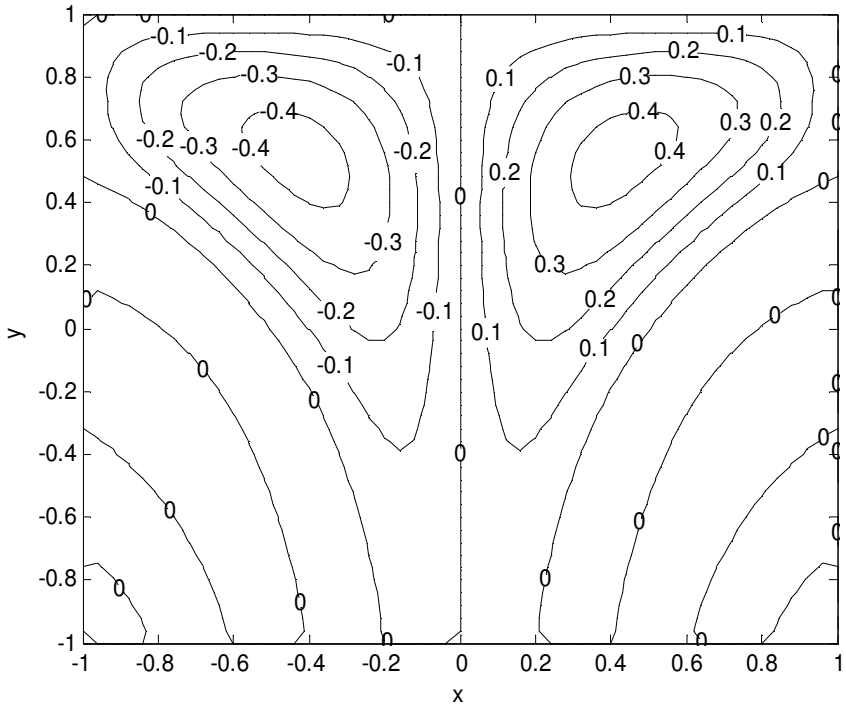
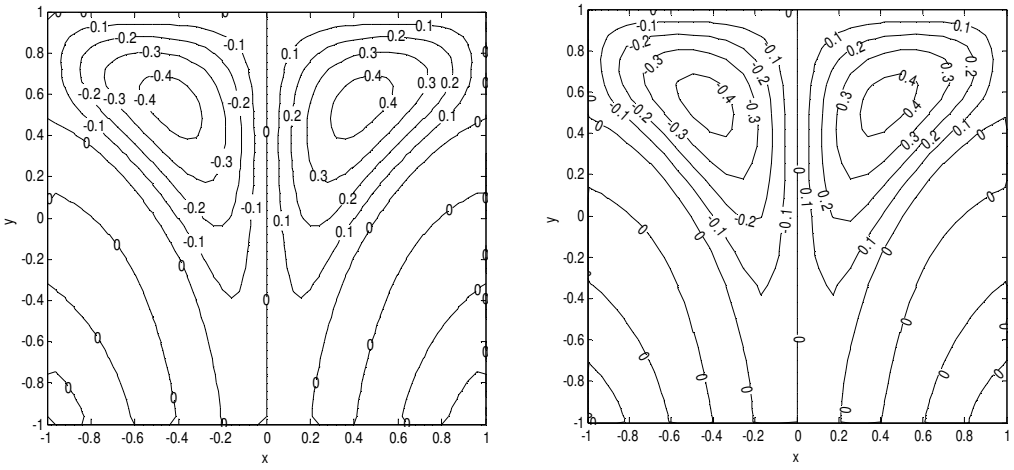


Figure 4.1b Contour plot of the solution of (4.1) for N=24 with a single rectangular spectral element. The midline is u(0,y)=0 line.



with interpolation without interpolation

Figure 4.1c Contour plot of the solution of (4.1) for N=24 with a single rectangular spectral element. The contour lines are less curvy.

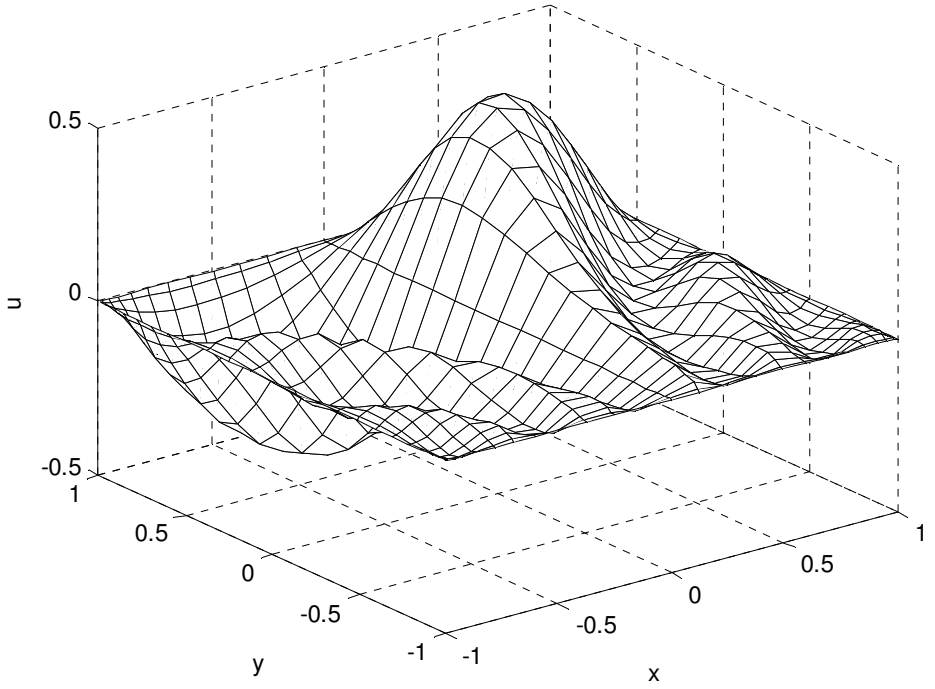


Figure 4.1d 3D Solution graph of (4.1) plotted on Gauss Lobatto Legendre grid for N=24 with a single rectangular spectral element. The u(0,y)=0 line is smooth.

The contours at the edges of the domain must not be broken. These errors seen in Fig. 4.1b are interpolation errors caused by interp2.m matlab built-in function. Because of that, the same graphs are plotted without interpolation. In Fig. 4.1c none of the contour lines are broken at y=0 line. On the midline u(0,y)=0 line the computation zeros change within the interval of -1.8788×10^{-15} and near the boundary -1.4084×10^{-15} .

4.1.2 Poisson equation with zero Neumann boundaries

Our second purpose is to solve the problem from [4]

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=-1 \tag{4.3}$$

u (1,y)=0, u(x,1)=0, 0<x, y<1 on the Dirichlet(essential) boundaries (4.3a)

$u_x(0, y)=0$ $u_y(x, 0)=0$ as shown in the FEM solution of the graph (4.3b)

4.1.2.1 The comparison of FEM and SEM

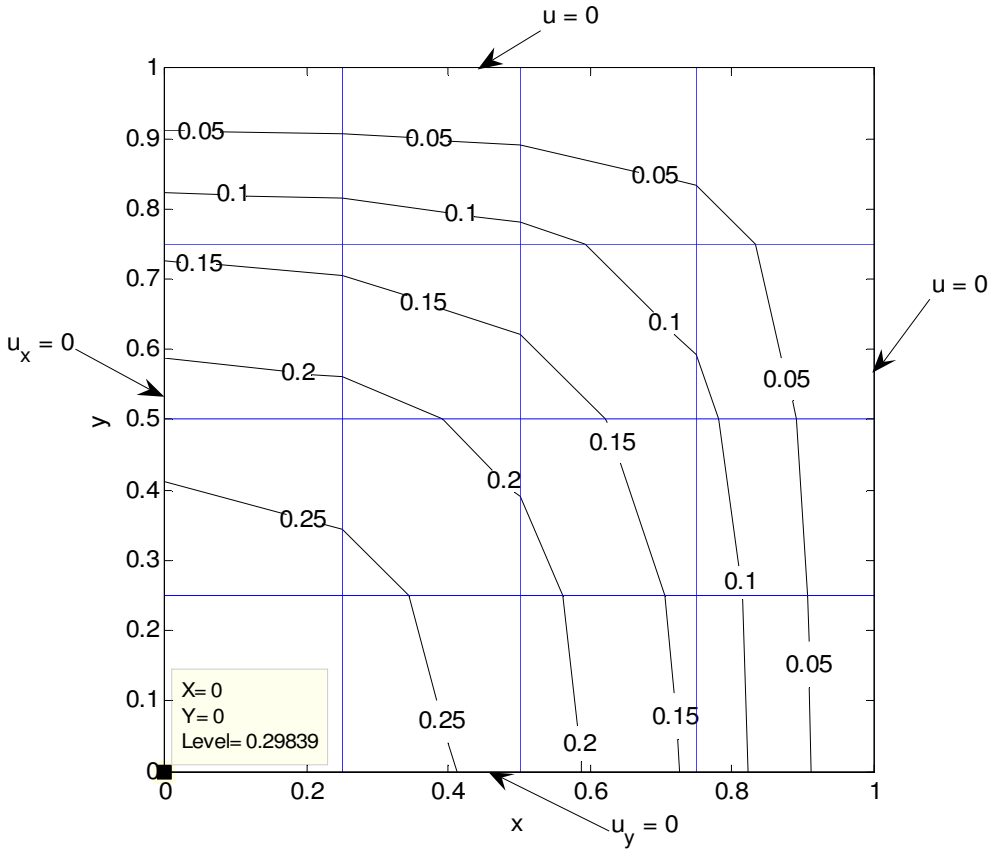


Figure 4.2 Finite element solution of Equation (4.3) with 16 linear rectangular element

Low order finite element method solutions can be used as a preconditioner for SEM. The vertical and horizontal lines are boundaries (edges) of the linear elements and the intersections of the lines are the nodes of the elements. From Figure 4.2, the plot of the solution has one hill at the coordinate of (0, 0). The value of u increases from (1, 1) to (0, 0) through the diagonal. There are five nodes on this diagonal and the increase is from 0 to 0.3125.

In this example, FEM has achieved the success of finding the characteristic of the solution. However, there is a problem at zero Neumann boundaries. The contours of 0.25, 0.2 and 0.15 are not perpendicular to x=0 and y=0 line. As a result of using first order FEM, the contours are made up of linear lines. It's like a light ray has broken while changing its environment.

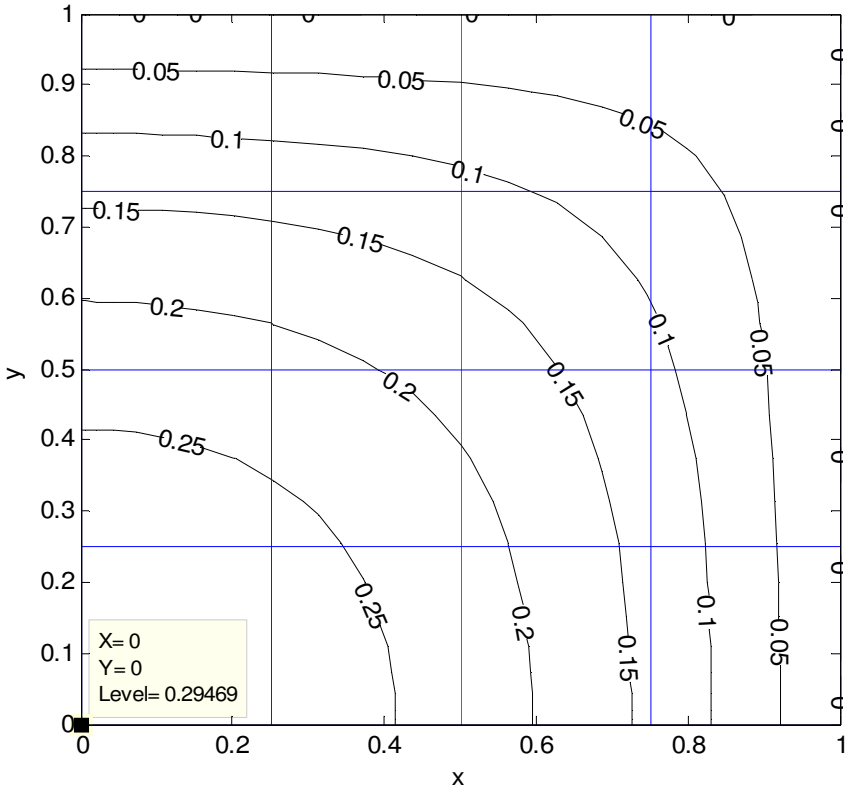


Figure 4.3 Contour graph of the spectral element solution of (4.3) with a single rectangular element for the order of N=24.

All of the contours are perpendicular to the zero Neumann boundaries at the solution graph of domain with a single spectral element. The graphs of SEM and FEM looks like same except this feature. The vertical and horizontal lines were drawn to show the success of FEM as a preconditioner.

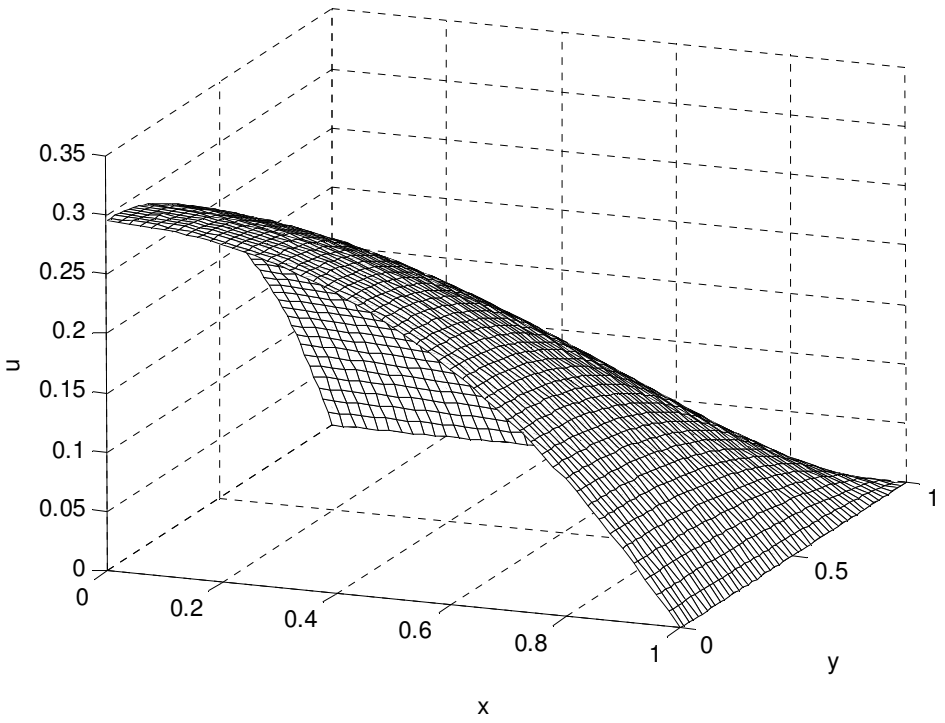


Figure 4.3b 3D graph of the solution of (4.3).

In Fig. 4.3b, the solution at (0, 0) is parallel to the x-y plane at the intersection of zero Neumann boundaries because the derivative of u to x and y is equal to zero. From both of the Figures 4.2 and 4.3, the solution of u is symmetric to the diagonal between (0, 0) and (1, 1) so this line is line of symmetry. On lines of symmetry the normal derivative of the solution is zero [4] as a result the solution contours are perpendicular to the line of symmetry in both of the Figures 4.2 and 4.3.

4.1.3 Steady Conduction Equation with a Convection face (Robin BC)

Consider the following two-dimensional steady conduction equation with a thermal conductivity k=20 W/mK and heat generation $q = 5\times10^{11}\times\sin(8x\times(y-1))\ W/m^3$.

$$-k\nabla^2 T = q \quad \text{on } \Omega \in [-1,1]\times[-1,1] \tag{4.1.3.1}$$

$$\text{T(-1,y) = 300 K} \quad \text{on } \Gamma \text{ (left face)} \tag{4.1.3.2a}$$

$$T(1,y) = 400\ K \quad \text{on } \Gamma \text{ (right face)} \tag{4.1.3.2b}$$

$$\frac{\partial T(x,-1)}{\partial y} = 0 \quad \text{on } \Gamma \text{ (down face is insulated)} \tag{4.1.3.2c}$$

Robin (mixed) BC is up face with convection heat transfer and written as

$$k\frac{\partial T(x,1)}{\partial y} = h(T(x,1) - T_{\infty}) \quad \text{on} \quad \Gamma \text{ (up face)} \tag{4.1.3.3}$$

Up face is exposed to a convection process with T_{∞}=300K and h=100 W/m^2 K. Domain's area is 4 mm^2. The steady conduction equation will be

$$\nabla^2 T = -2000 \times \sin(8x \times (y-1)) \quad \text{on } \Omega \in [-1,1] \times [-1,1] \tag{4.1.3.1a}$$

Convected face forming Robin BC Equation (4.1.3.3) combines the Neumann and Dirichlet boundary conditions as a result its implementation will be a modification and connection of them. Poisson equations with Robin boundary conditions are implemented like

```
upface=find(yy==1);
upface=upface(2:N);
DNBC = kron(D,I);
for i=1:23
DNBC(upface(i),upface(i))=DNBC(upface(i),upface(i))-5;
end
delta(upface,:)=DNBC(upface,:);
RHS(upface)=-1500;
```

The temperature contours are called as isotherms. The angle between the up edge and the isotherms of T=380, 360, 340 and 320 K over hill are less than right angle as seen in Fig. 4.4a. This feature is similar to the Table 2.1 of Incropera and DeWitt [37]. The isotherms are turning around themselves at the local maximum (heat generation) and minimum (heat sink) values of the q heat generation function to form a bottom at (-0.4649, 0.6264) and a hill by closing as seen in Fig. 4.4. The directions of the heat

fluxes at T=420 K isotherm are from hill to up face and bottom in the hollow of the T=240 K isotherm. The direction of the heat flux is always from hot places to cold places. Therefore, all of the heat flux vectors are perpendicular (normal) to isotherms. The ends of the isotherms at the down face are perpendicular to the y = 0 line (down face) as a feature of homogeneous Neumann boundary conditions.

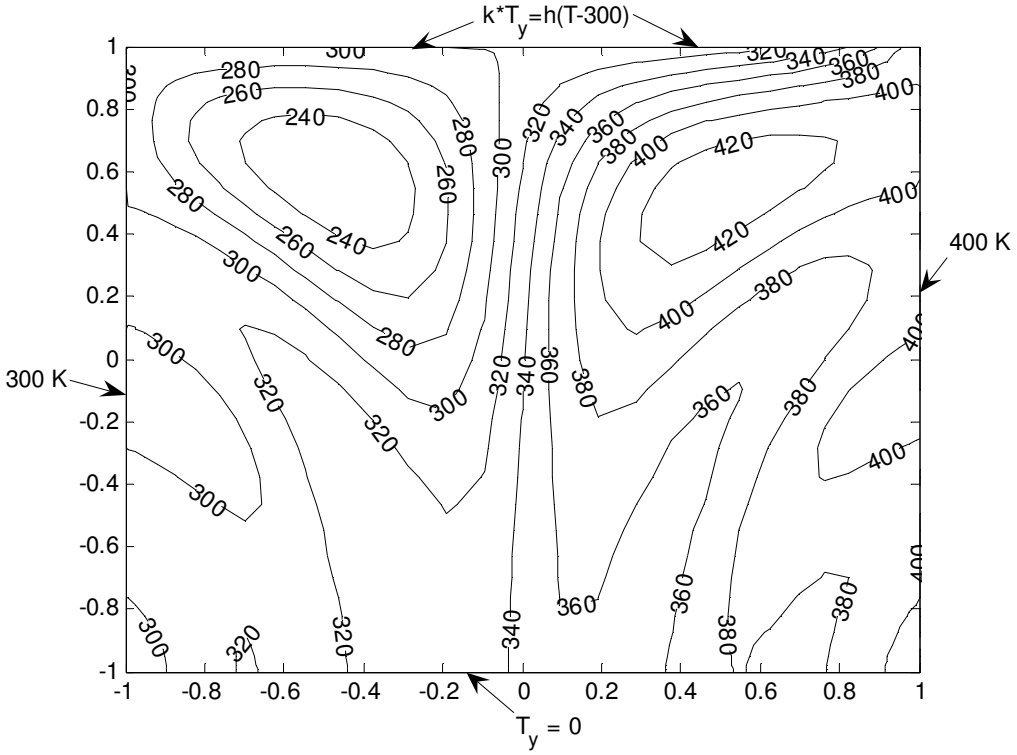


Figure 4.4a Isotherms of the spectral element solution of Equation (4.1.3.1a) with a single rectangular element for the order of N=32. The solution is plotted without interpolation.

The negative values of q function are called as heat sink where thermal energy is being consumed. The positive values of q heat generation function are also called as energy source term where thermal energy is being generated. The plot of the q/k function is anti-symmetric about x=0 line as seen in Fig. 4.4b.

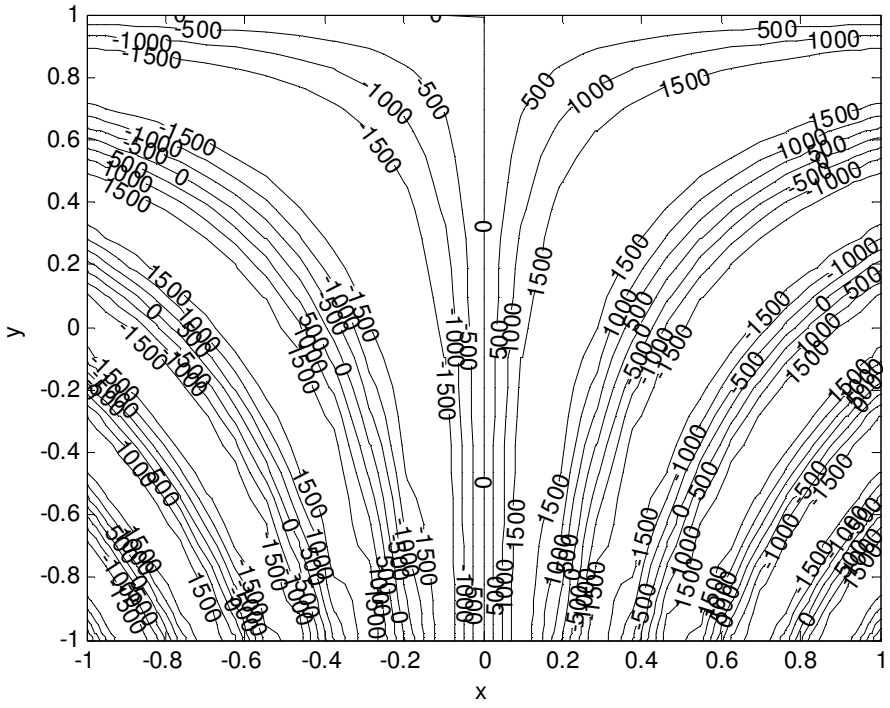


Figure 4.4b Contour graph of the RHS of Equation (4.1.3.1a) with a single rectangular element for the order of N=32. The solution is plotted without interpolation.

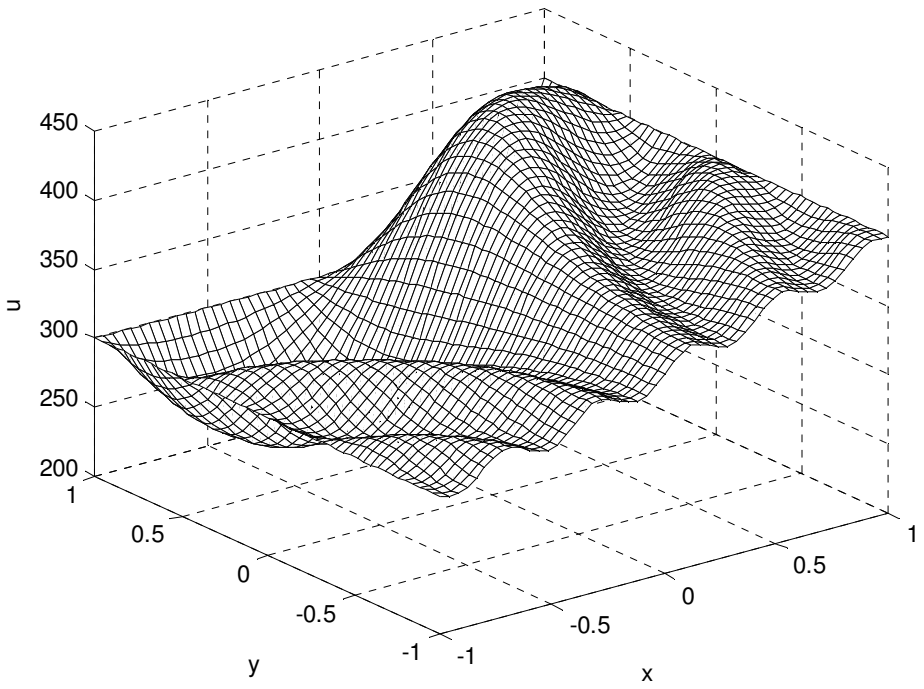


Figure 4.4c 3D spectral element solution graph of Equation (4.1.3.1) for N=32 with a single rectangular spectral element.

4.1.4 Poisson Equation with all nonzero Dirichlet boundary conditions

Consider the following two-dimensional steady Poisson equation with Dirichlet boundary conditions from [5]

$$-\nabla^2 u = f \qquad \text{on} \qquad \Omega \in [0,1]\times[0,1] \tag{4.4a}$$

$$u = u_{exact} \qquad \text{on} \qquad \Gamma \tag{4.4b}$$

The following force function

$$f = 4\pi \sin(4\pi A)\left(\left(\frac{\partial A}{\partial x}\right)^2 + \left(\frac{\partial A}{\partial x}\right)^2\right) + 4\pi \cos(4\pi A)\left(\frac{\partial^2 A}{\partial x^2} + \frac{\partial^2 A}{\partial y^2}\right) \tag{4.4c}$$

Results in the exact solution of

$$u_{exact} = \sin(4\pi A) \tag{4.4d}$$

where

$$A = \sqrt{(x-2)^2 + (y-2)^2} \tag{4.4e}$$

From Equation (4.4d), the exact solution is a sine function so it is periodic as a result periodicity is seen at the diagonal line between (0, 0) and (1, 1) in Fig. 4.5.

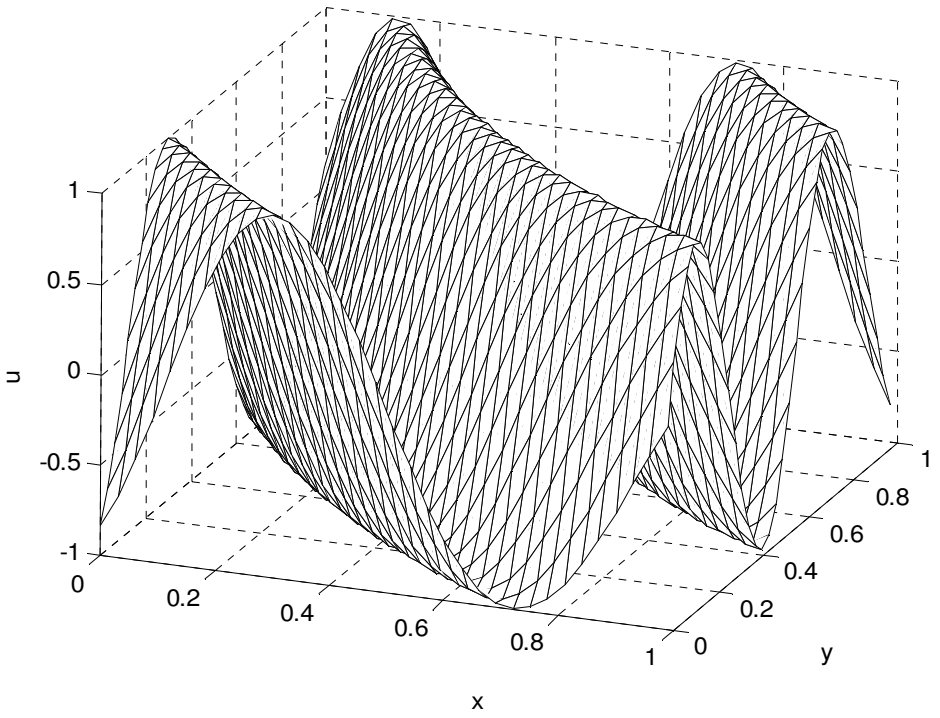


Figure 4.5 SEM solution for a single element with a degree of N = 24.

4.2 Poisson Equation solution with multi elements

Three meshes are used as seen in Fig. 3.5, 3.10 and 3.12. All of the global stiffness matrices of the three meshes are made up of assembling same size of matrices but Rectangular element stiffness matrices are sparser than quadrilateral element stiffness matrices. If a stiffness matrix has a value or values smaller than 1.0×10^{-15} this value won't be assembled to the assembled stiffness matrix as a result Mesh1's assembled stiffness matrix is sparser than the assembled quadrilateral element stiffness matrices. For N=12 local stiffness matrices have got a size of 169×169. After assembling local stiffness matrices and local load vectors, the size of the assembled stiffness matrix is 481×481 for all of them.

Point per wavelength (ppw) is an important issue for spectral methods. At first mesh there are three same rectangular elements. The number of nodes per area is all same for both of the elements of Mesh 1. On the line of x=0.75 three half wavelength is

seen in Fig. 4.6 and for degree of N=12 or larger the largest error is seen at the 2nd element. That's why the errors at x=0.75 have been plotted. In mesh2 right face of the 1st element is drawn at the center of (0, 0) with a radius of $\sqrt{2}/2$. While decreasing the area of 2nd element, Node per area of 2nd element increased. $NPA_{EL2} < NPA_{EL1} < NPA_{EL3}$

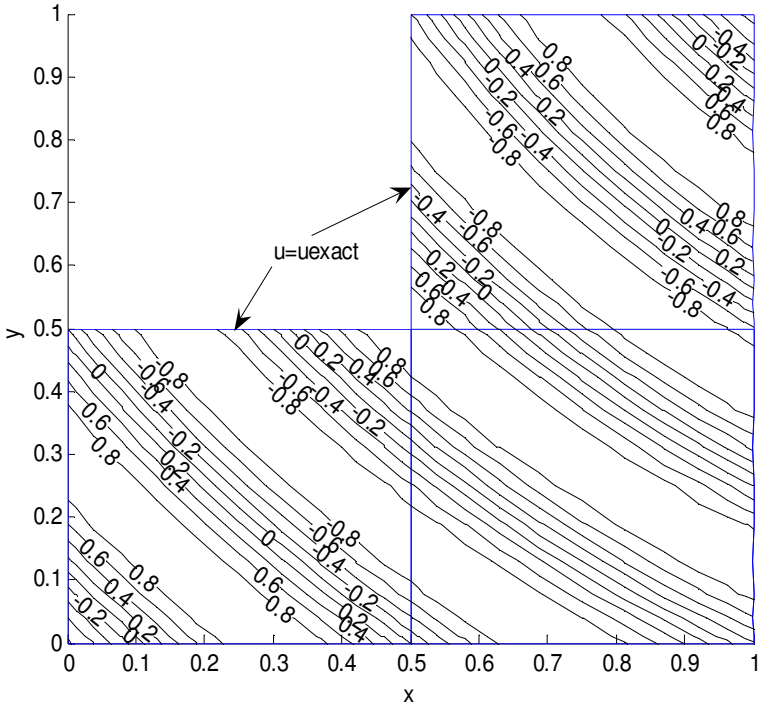


Figure 4.6 The results of the solution for the Mesh 1 made up of three elements with a degree of 12. Blue lines are element boundaries.

4.2.1 Arc edged Curvilinear Quadrilateral Element

Jacobi matrix is used to find quadrilateral element's local stiffness matrix. By using the determinant of the Jacobi matrix the exact area of the element must be found.

$$A = \int_{-1}^{1}\int_{-1}^{1} J(r)drds$$

Here J(r) is the determinant of Jacobi matrix. Dividing rectangular domain to rectangular elements is not an obligation. Changing size or the type of the element like quadrilateral element is a way to achieve or obtain better. Rectangular domain is divided to complex elements to control the success of the Jacobi mapping.

Figure 4.7, the contour graph of the solution is smoother than the Figure 4.8. The 2nd element with two arc face has the same area as the rectangular one. Arc upper face is needless because of the accuracy decreased by Jacobi mapping. Calculating element matrices will consume much more time if more face needs Jacobi mapping.

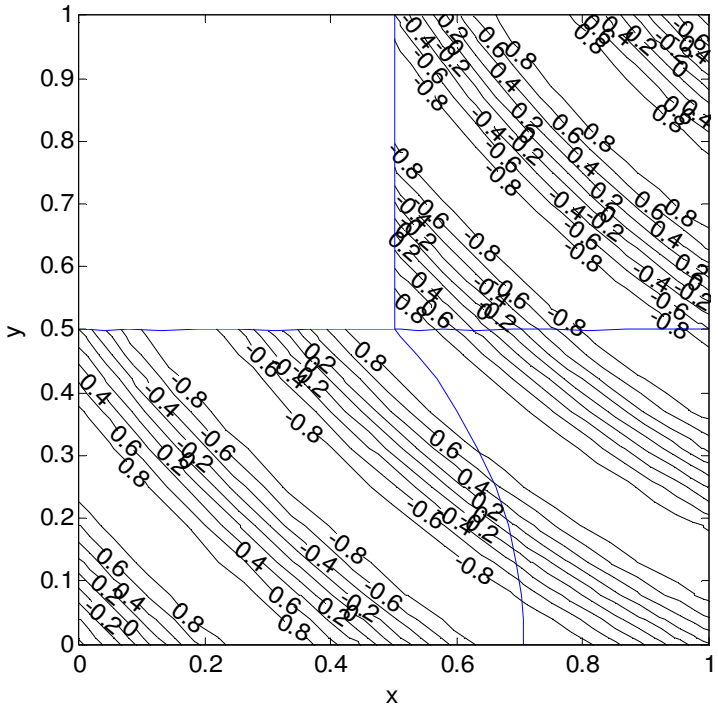


Figure 4.7 The results of the solution for the Mesh 2 made up of three elements with a degree of 12 as seen in Fig. 3.10. Blue lines are element boundaries.

The results are better and Fig 4.9 looks like 1D SEM solution in Fig. A.1. The Jacobi mapping is tested by dividing the domain to arc faced elements as a benchmark.

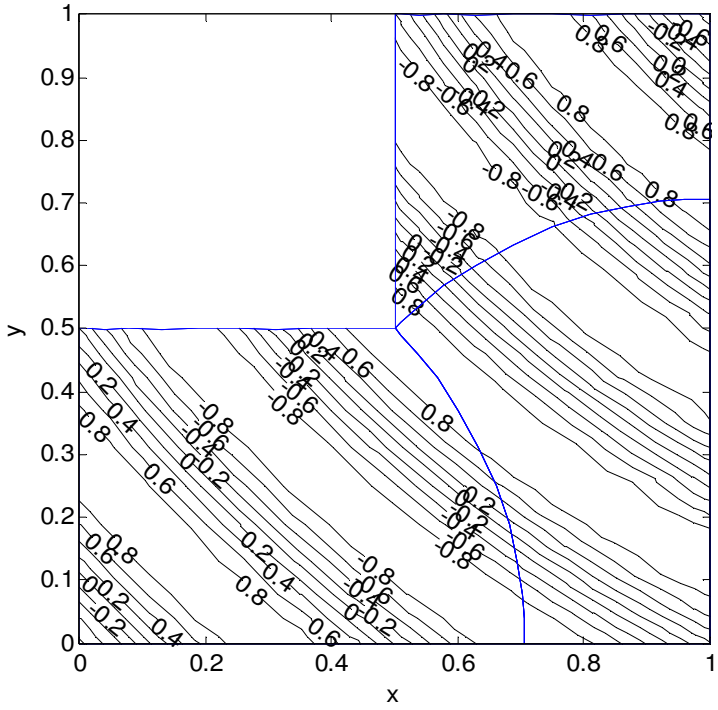


Figure 4.8 The results of the solution for the Mesh 2 made up of three elements with a degree of 12 as seen in Fig. 3.12. Blue lines are element boundaries.

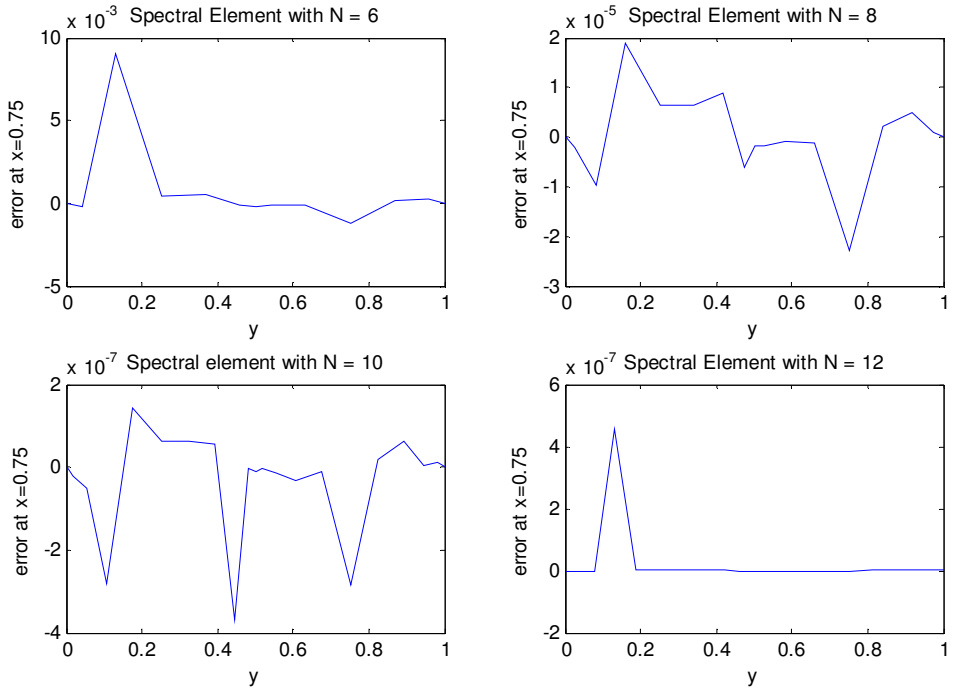


Figure 4.9 The errors of the results at x=0.75 line for mesh seen in Fig.3.5.

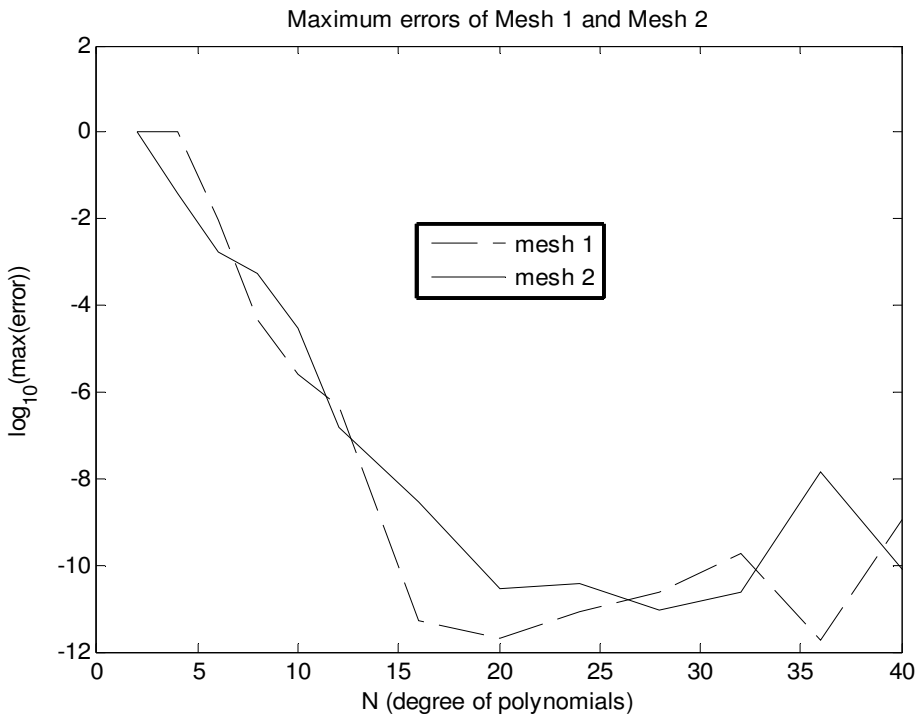


Figure 4.10 Spectral convergence obtained for Mesh 1 and 2 seen in Fig. 3.5 and 3.10

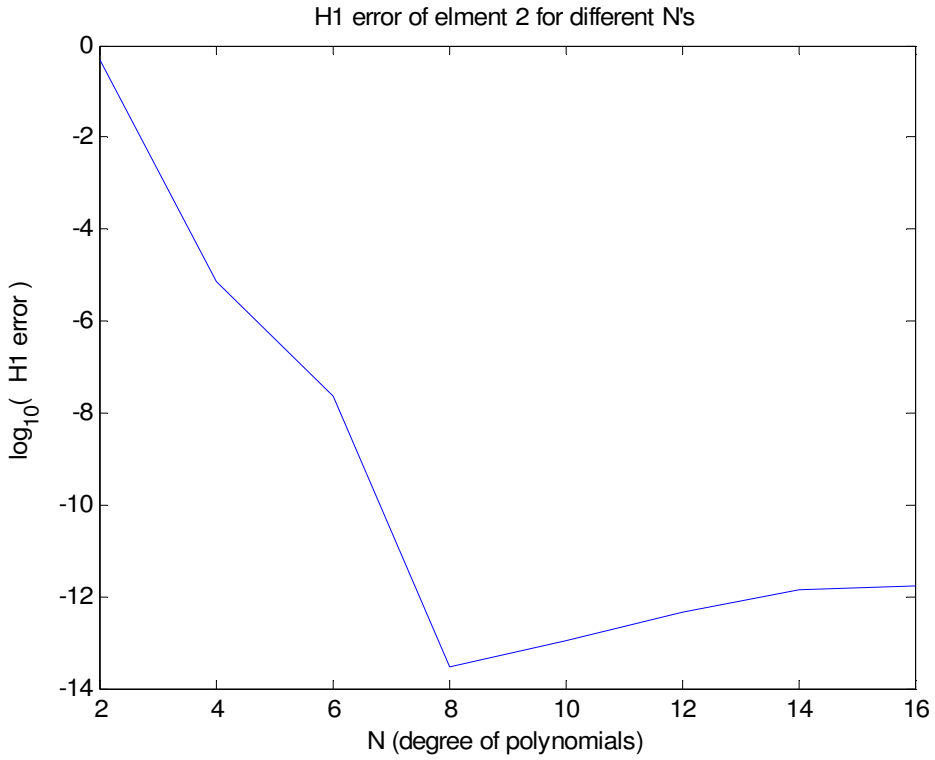


Figure 4.11 H1 errors for mesh 2 seen in Fig. 3.10.

The area of the domain of the single element and the other three meshes are different, so the Table 4.1 is not drawn for degree of freedom. The mesh of Fig. 4.5 with a degree of N=24 is as same as Fig. 3.2 except the domain of Fig. 4.5 spans the area $\Omega \in [0,1]\times[0,1]$ and the mesh of Fig. 4.6 is the Fig. 3.5 with an element degree of N=12. Both of the meshes of the Fig 4.5 and Fig. 4.6 have same number of nodes per unit area. The accuracy achieved with less nodes for a single element. This is not amazing, 1D SEM solution with a single element is also better than domain with more elements for 1D periodic problem. It is seen in Fig. A.1 in Appendix A. It is normal because it is first designed to solve channel expansion. Its capacity is seen a problem like in Fig. A.2 or discontinuous problems like 1D shock. SEM is not invented for steady diffusion problems.

Table 4.1 Maximum absolute relative error of Poisson Equation (4.4) for the entire domain

N	Single element	Mesh 1 (Fig. 3.5)	Mesh 2 (Fig. 3.10)	Mesh 3 (Fig.3.12)
2		1.1403	1.5663	1.6623
4	0.4404	1.2798	1.4389	0.2263
6	0.0724	0.0089	0.0069	0.1404
8	0.0548	1.8059×10^{-4}	5.6197×10^{-4}	0.0572
10	0.0029	2.4784×10^{-6}	3.0437×10^{-5}	0.0148
12	1.2073×10^{-4}	4.5781×10^{-7}	2.6920×10^{-6}	5.7047×10^{-6}
16	3.7225×10^{-7}	2.5270×10^{-11}	2.0046×10^{-8}	4.8677×10^{-8}
20	4.0880×10^{-10}	2.0249×10^{-12}	1.4206×10^{-10}	1.4862×10^{-10}
24	2.6689×10^{-11}	2.6145×10^{-10}	3.7651×10^{-11}	1.2995×10^{-9}
28	1.2045×10^{-11}	2.5300×10^{-11}	2.3118×10^{-11}	1.1142×10^{-10}
32	4.3815×10^{-12}	1.8417×10^{-10}	2.4370×10^{-11}	2.4050×10^{-11}
36	3.6459×10^{-11}	2.0805×10^{-8}	1.3703×10^{-8}	5.5512×10^{-11}
40	2.4293×10^{-12}	1.0802×10^{-9}	9.2790×10^{-11}	6.8234×10^{-10}

Needless curvilinear quadrilateral element usage causes extra time consumption. The comparison for the accuracy of the second element is seen in Table 4.2. Mesh1 reaches the lowest error for 2nd element with a degree of N=20. Mesh2 passes mesh1 only for N=12. Mesh1 is the first which reaches the computation zero for N=16. Curvy elements must only be used to discretize the curvy boundaries of the domain.

Table 4.2 Maximum absolute relative error for the entire of the second element

N	Mesh 1 (Fig. 3.5)	Mesh 2 (Fig. 3.10)	Mesh 3 (Fig.3.12)
2	0.5167	0.3723	0.2168
4	0.0606	0.0206	0.1006
6	0.0090	0.0016	0.0061
8	1.8824×10^{-5}	2.0467×10^{-5}	1.0748×10^{-4}
10	1.4186×10^{-7}	4.4625×10^{-7}	3.4713×10^{-5}
12	4.5781×10^{-7}	7.2291×10^{-8}	1.0107×10^{-6}
16	8.0173×10^{-14}	1.3504×10^{-10}	1.1937×10^{-9}
20	5.8280×10^{-14}	5.1680×10^{-13}	2.2909×10^{-11}
24	1.0122×10^{-12}	1.5530×10^{-13}	2.2068×10^{-13}
28	4.7380×10^{-13}	6.3631×10^{-13}	1.7309×10^{-13}
32	1.4405×10^{-12}	2.8633×10^{-12}	1.3065×10^{-13}
36	3.0318×10^{-11}	1.5288×10^{-11}	4.3030×10^{-13}
40	8.4921×10^{-11}	6.9665×10^{-12}	3.2235×10^{-12}

The accuracy of four element passes the accuracy of a single element for DOF=28 in Table 4.3. Four element reaches the computation zero for DOF=32. However, mesh1 and mesh with four elements have equal degree and node per area, mesh with four elements is better. The question is why mesh1 can not reach this accuracy and computation zero. Mesh1 in Fig.3.5 has an inner corner like lid-driven cavity problem. The effect of the geometric singularity is occurred at the inner corner of the three meshes of the Poisson Equation so it can not reach the computation zero. To increase the node per area, the area of the 2nd element is decreased by changing its

shape as seen in Fig. 3.10. Constructing mesh2 in Fig. 3.10 or mesh3 in Fig.3.12 causes higher errors like 2.4370×10^{-11} or 2.4050×10^{-11} as seen in Table 4.1 for the entire domain and the little fluctuations have seen at the contour lines of the Fig. 4.18, so usage of the curvilinear quadrilateral element is not a solution for the geometric singularity. It increases the band size and density as seen in the sparsity graphs of Fig. 3.11 and Fig. 3.13 as a result the evaluation time is increased.

Spectral and spectral element methods suffer more from the geometric singularities such as corners or discontinuities inherent in the solution such as shock waves. Here, all boundaries are continuous (u=uexact), therefore singularities are due to the inner corner in the domain of Fig. 4.6, 4.7 and 4.8. Pathria and Karniadakis [23] suggests auxiliary mapping to achieve exponential convergence.

Table 4.3 Maximum absolute relative error for DOF in the entire domain

Degree of freedom (DOF)	Single element DOF=N	Four element DOF=N/2
4	0.4404	1.0265
8	0.0548	0.0247
12	1.2073×10^{-4}	4.5853×10^{-4}
16	3.7225×10^{-7}	6.1216×10^{-6}
20	4.0880×10^{-10}	6.6509×10^{-8}
24	2.6689×10^{-11}	5.7290×10^{-10}
28	1.2045×10^{-11}	5.1903×10^{-12}
32	4.3815×10^{-12}	8.9262×10^{-14}
36	3.6459×10^{-11}	6.8945×10^{-14}
40	2.4293×10^{-12}	2.0461×10^{-13}

Meshes with same DOF have same number of nodes in each direction and the entire of the domain. Four element mesh seen in Fig. 4.11b have 625 number of nodes like

the single element with an element degree N = 24. For single element, element degree and DOF are equal to each other.

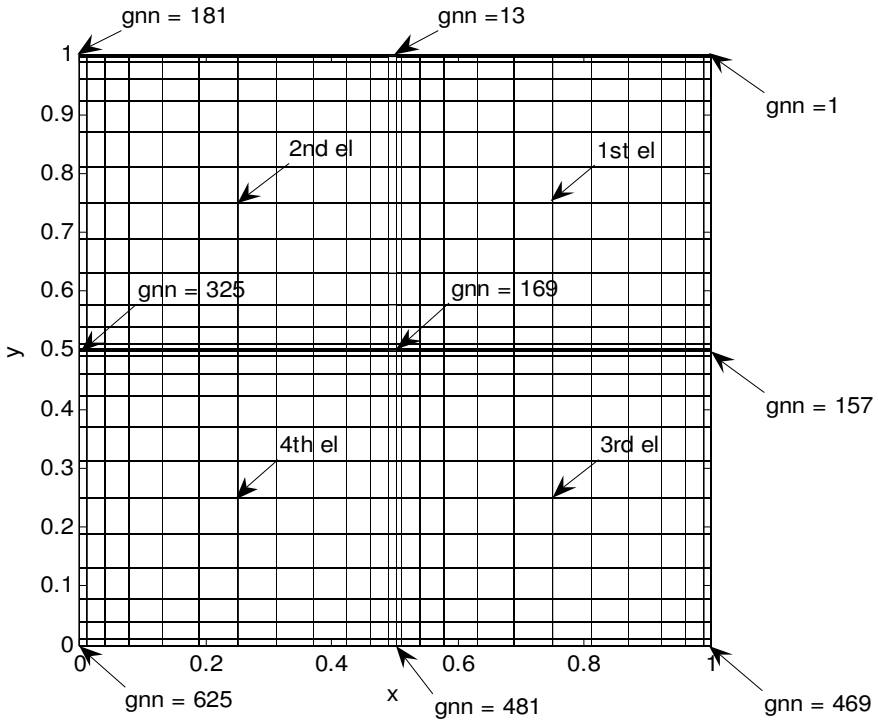


Figure 4.11b The grid of the Mesh for DOF = 24 made up of four rectangular elements. The degree of each element is N = 12.

4.3 Inviscid Flow Around a Circular Cylinder

The irrotational flow of an ideal fluid about a circular cylinder, placed with its axis perpendicular to the plane of the flow between two horizontal walls. [4]

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0 \tag{4.5}$$

$u(x,y)=0, \quad -1<x,y<1$ on the Dirichlet(essential) boundaries (4.5a)

The exact solution is found for the center of the circle as origin from [6].

$$\psi = U\left\{ y - \frac{\frac{H}{2\pi}\sinh^2(\frac{\pi b}{H})\sin(\frac{2\pi y}{H})}{\cosh^2(\frac{\pi x}{H}) - \cos^2(\frac{\pi y}{H})} \right\} \tag{4.6}$$

H is the vertical distance between the two plates and b is the radius. Chung [6] warned that this analytical solution is not accurate for large values b/H ratio. For this problem b/H is 0.25 and the error is 2.34%. SEM solution is far more accurate so the comparison is needless to have an idea.

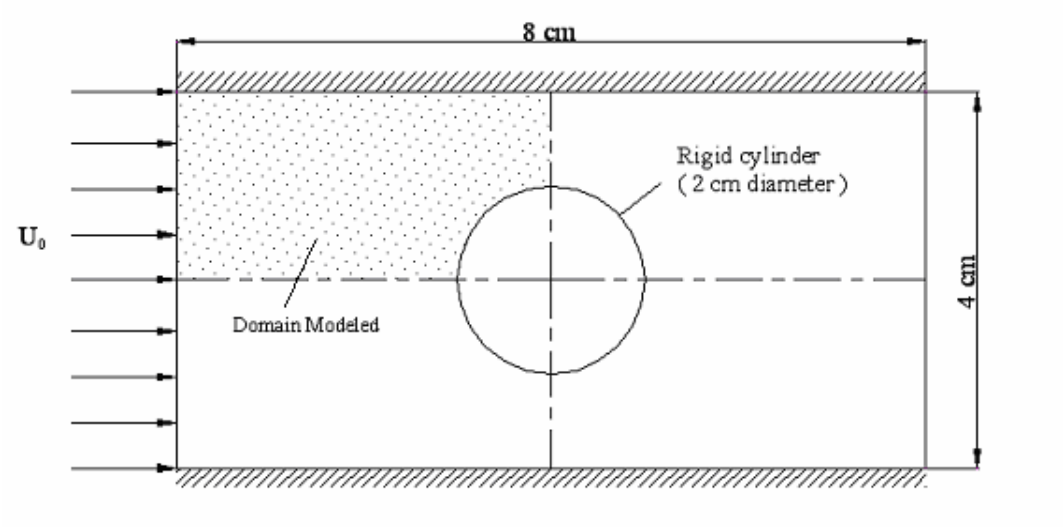


Figure 4.12 Domain for the stream function and velocity potential formulations of inviscid (irrotational) flow about a cylinder.

4.3.1 Stream function formulation

Because of symmetry about horizontal and vertical lines, a quadrant is enough to model the solution.

The boundary conditions on the stream function ψ can be determined as follows in Fig 4.13. Streamlines have the property that flow perpendicular to them is zero.

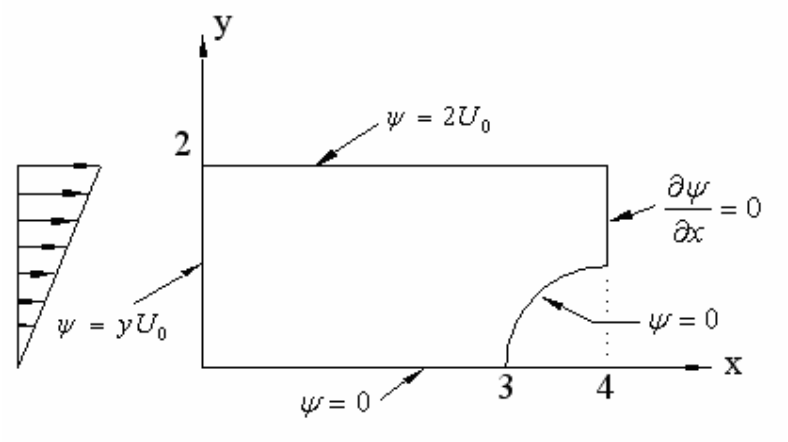


Figure 4.13 Computational domain and boundary conditions for the stream function formulation irrotational flow around a cylinder.

All contour lines should go to the up. Any line mustn't be seen as going down. It is seen at lower order FEM solution [4]. The domain is sliced two quadrilateral spectral elements from the diagonal passes through the center of the circle. From Fig. 4.14, none of the contour lines are seen broken down through the element boundary diagonal.

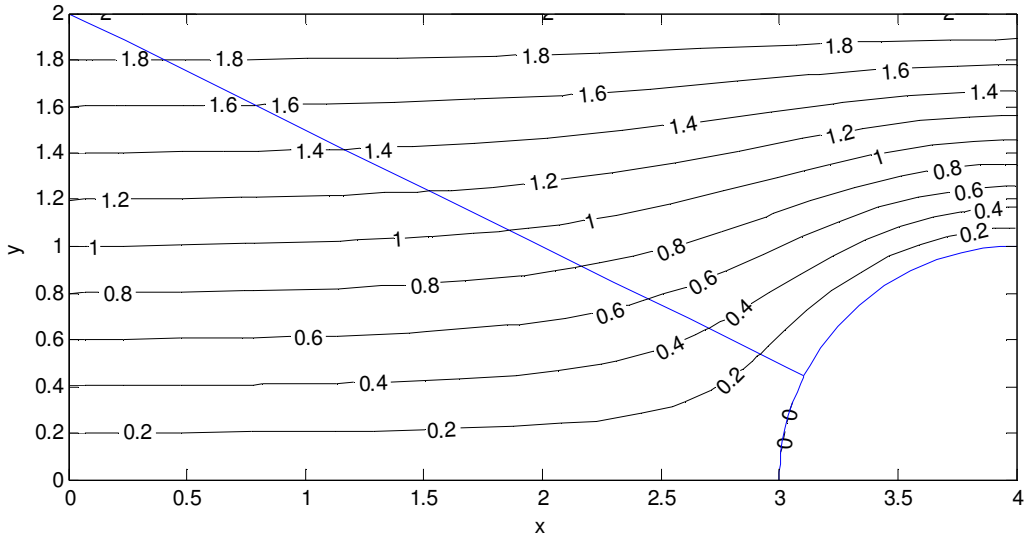


Figure 4.14 SEM solution of stream function with the BC's shown for two elements with the order of N=12.

From Fig. 4.14, all contour lines are perpendicular to zero Neumann BC at the x=4 line. All contour lines are smooth, no fluctuations are seen. Stream function is implemented and plotted by streamfunction.m as seen in Appendix E.

4.3.2 Velocity potential formulation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{4.7}$$

The boundary conditions are shown in Fig. 4.15. The nonzero Neumann boundary is the EA edge of the quadrant domain.

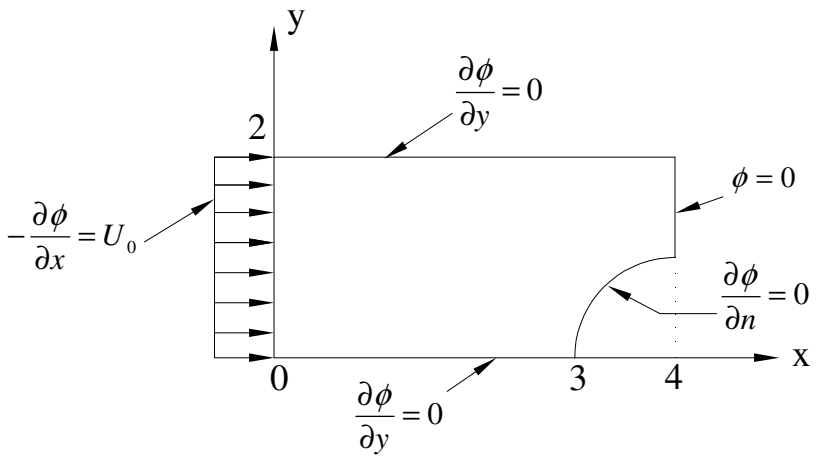


Figure 4.15 Computational domain and boundary conditions for the velocity potential formulation around cylinder.

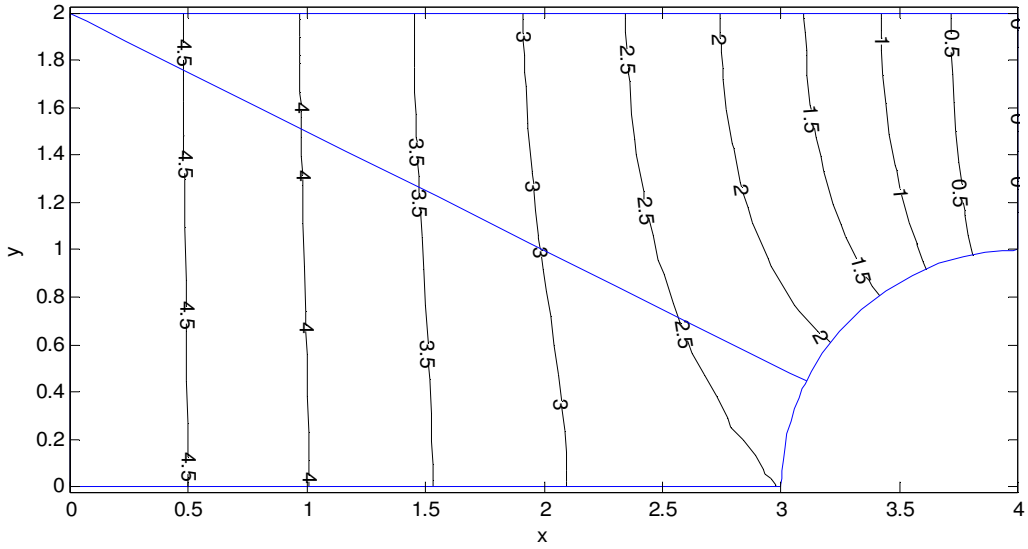


Figure 4.16 Velocity potentials are plotted for N=12. Same mesh is used in Fig. 3.15.

From Fig. 4.16, all contours are perpendicular to zero Neumann BC at the y=0 line and the arc line. At point (3, 0) there is a connection of zero Neumann BC's as a result the contour seems like that.

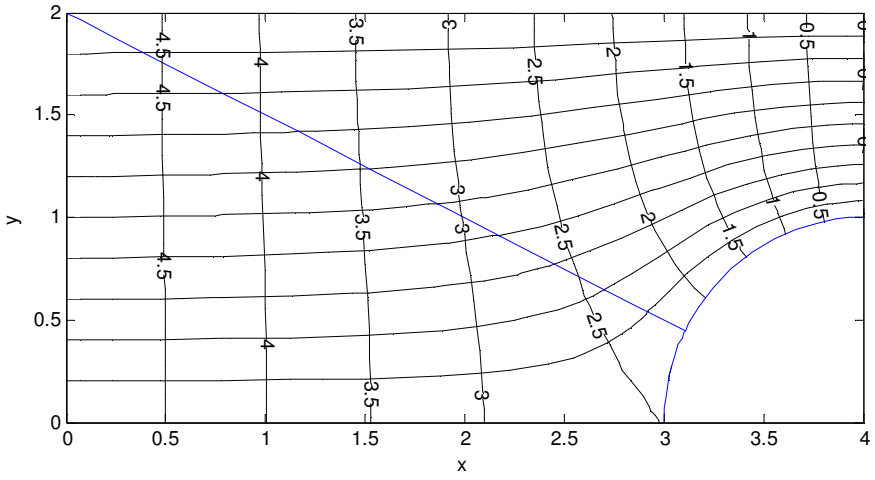


Figure 4.17 The figures of 4.14 and 4.16 plotted together to show the accuracy.

From Fig. 4.17, velocity potentials are perpendicular to the streamlines as expected.

4.4 Helmholtz Equation

4.4.1 Helmholtz Equation with a single element

A variation of the Poisson equation is the Helmholtz equation from [1].

$$u_{xx} + u_{yy} + k^2 u = f(x, y)\text{, -1 < x, y < 1, on the boundary u = 0 k=9} \tag{4.8}$$

The problem is on the rectangular domain. This equation arises in the analysis of wave propagation governed by the equation

$$-U_{tt} + U_{xx} + U_{yy} = e^{ikt} f(x, y) \tag{4.4.1.1}$$

after separation of variables to get $U(x, y, t) = e^{ikt} u(x, y)$. Program of Helmhholtz equation is a minor modification of Poisson Equation on the rectangular geometry to solve such a problem for the particular choices

$$\text{k = 9, } f(x, y) = \exp(-10[(y-1)^2 + \left(x - \frac{1}{2}\right)^2]) \tag{4.8a}$$

The modification on the code

```
delta=-(kron(RHS2,df2(2:N,2:N))+kron(df2(2:N,2:N),RHS2))+k^2*s;
```

Degree of GLL polynomials is chosen as even number to have odd number of nodes. Element with odd numbered nodes have a node at the midpoint of the element. We delete the rows and columns belong to the Dirichlet boundary condition in the differentiation matrix. Solution graphs are plotted as two types. One is 2D contour the other is 3D graph of the results on the solution domain. Our domain is x = [-1, 1] and y = [-1, 1]. Therefore, it's important what happened at x=0, y=0. Therefore, the even number is chosen because of this.

The solution appears as a 3D mesh plot in Fig. 4.18 and as a contour plot in Fig. 4.19. It is clear that the response generated by this forcing function f(x, y) for this value k=9 has approximately the form of a wave with three half-wavelengths in the x direction and five half-wavelengths in the y direction. This is easily explained. Such a wave is an eigenfunction of the homogeneous Helmholtz problem

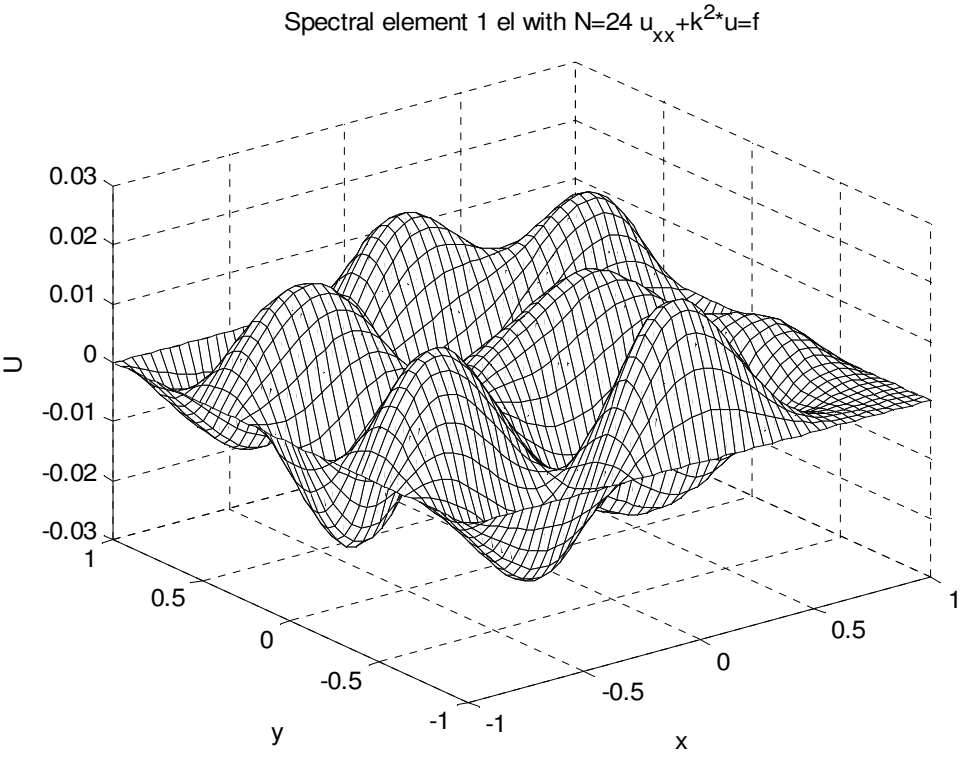


Figure 4.18 3D graph of the solution of $u_{xx} + u_{yy} + k^2 u = f(x, y)$

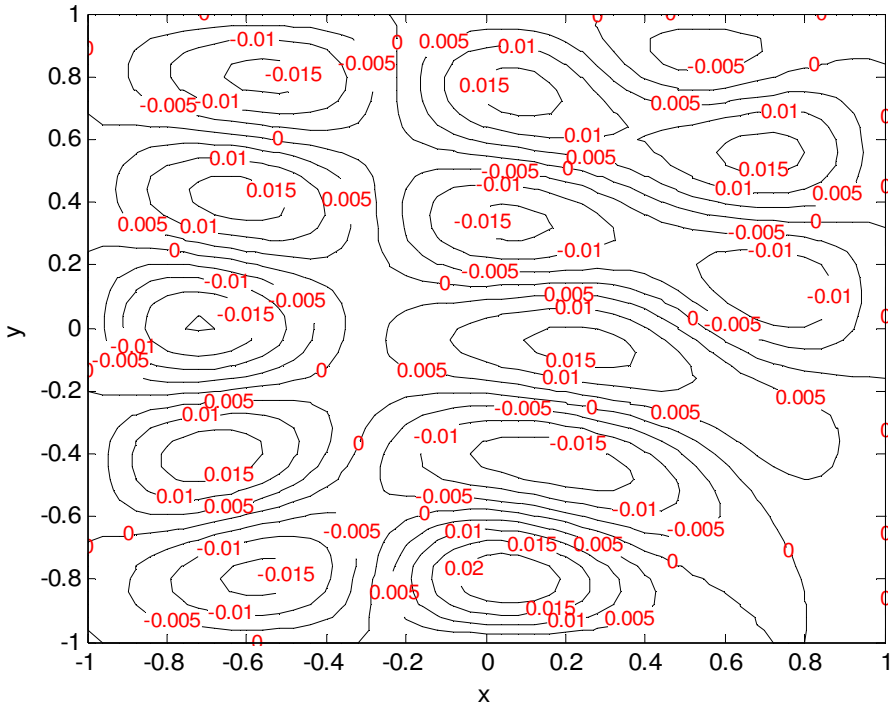


Figure 4.19 Contour graph of the solution of $u_{xx}+u_{yy}+k^2u=f(x,y)$

4.4.2 Helmholtz Equation with more elements

Consider the following two-dimensional Helmholtz equation with Dirichlet boundary conditions and $\lambda=1$

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+u=f \qquad \text{on} \qquad \Omega\in[-1,1]\times[-1,1] \tag{4.9}$$

$$u=u_{exact} \qquad \text{on} \qquad \Gamma \tag{4.9a}$$

Results in the exact solution of

$$u_{exact}=\sin(\pi x)\cos(\pi y) \tag{4.9b}$$

Helmholtz Equation (4.9) that we know its exact solution is a problem from Karniadakis's book [2]. Helmholtz equation's f function has been calculated from a

known u function $u(x, y) = \sin(\pi x)\cos(\pi y)$ by using symbolic differentiation feature of MATLAB as `f=-(diff(u,xx,2)+diff(u,yy,2))+u.`

The domain is sliced from x and y axis so four elements are used to represent the solution domain. From Fig. 4.20 the function's solution is symmetric to x axis.

The highest error is at x=0.5 and y=0.076826, and the second highest is at x=-0.5 and y=0.076826 in Fig. 4.21 for N=12. The highest error is found at the local x axis of the element and lower part. Examining the results from array editor of The Matlab, the largest errors are near the top of the hill and near the lowest part of the hollow. The peak of the hills and the bottom of the hollows are at the middle of the boundary (connection) line of the elements. The contour graphs seen in Figures 4.20, 4.21 and 4.22 are including four quarter wave-lengths and two half wave-lengths.

Also at the hill or at the bottom points of the contour graphs seen in Figures 4.20, 4.21 and 4.22 the derivative of the exact solution is equal to zero. When thinking the first element sliced from local x and y axis, lower left part looks like FEM example so the mesh is not dense enough at these hills and hollows.

From Fig. 4.20 it is seen that there are zero lines at y=0.5, y=-0.5 and x = 0. From Figures 4.21 and 4.22 it is seen that there is a gap between the zero lines for N=12 and N=18. The gaps of N=12 is wider than the gaps of N=18. At x=0 u=0, at y=0.5 and y=-0.5 u=0 so there must be a zero line. For N=32 the zero line is continuous in Fig. 4.20. For N=18 there is not an intersection plus at (0, 0.5) and (0, -0.5) but the error found is less than N=32.

Spectral methods have some difficulties to solve problems with sharp corners like hat function. For N=12 and N=18 the same difficulty has been seen at the coordinates of (0,-0.5) and (0, 0.5). These are the middle of the boundary line of the element.

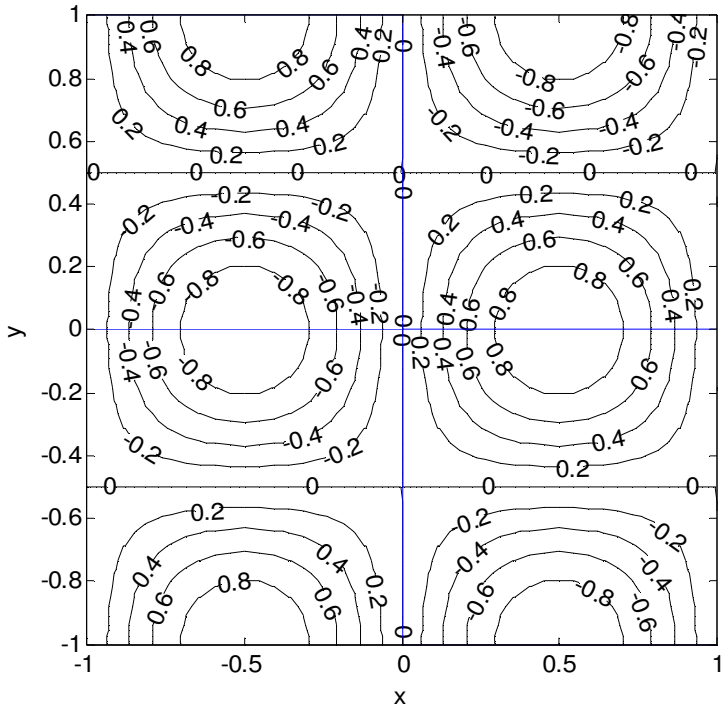


Figure 4.20 Spectral Element Solution graph of the Helmholtz equation $u_{xx} + u_{yy} + \lambda u = f(x, y)$ with 4 elements with the order of N=32. Blue lines are element boundaries.

u=0 lines must intersect with a right angle. It is seen at the Fig. 4.20 for N=32. The edges of the 0.8 contour circle couldn't intersect with a right angle for Fig 4.21 of N = 12 which means the solution is not enough. The smooth 0.8 curve contour is seen for Fig. 4.20 of N = 32. For N=32 the contour lines are seen clearly perpendicular to the x axis as a feature of line of symmetry.

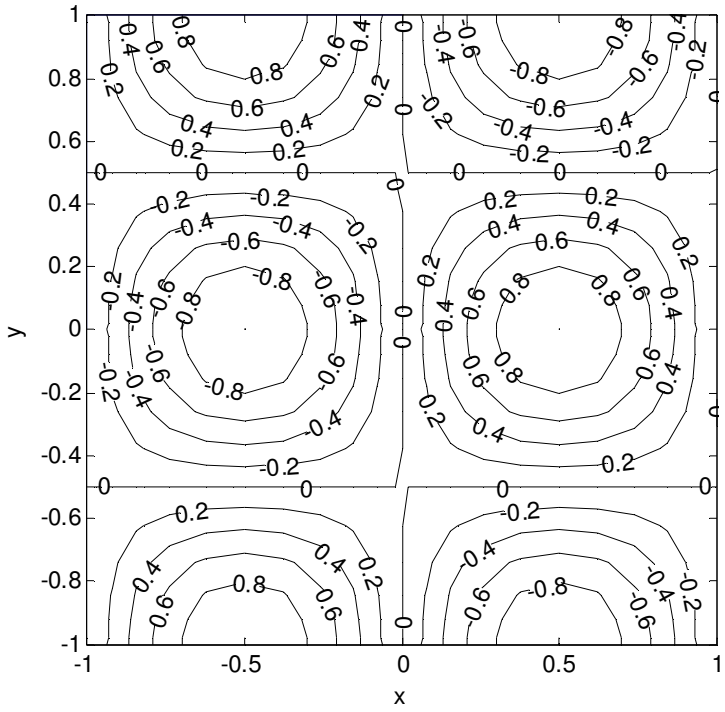


Figure 4.21 Spectral Element Solution graph of the Helmholtz equation $u_{xx} + u_{yy} + \lambda u = f(x, y)$ with 4 elements with the order of N=12. Element boundaries are same so they are not drawn to see the graph better.

For Fig. 4.21, at the coordinates of (-0.5, 0) and (0.5, 0) a point is seen as contour have a value of 1. From Fig. 4.22 the solution loses hill point at (0.5, 0) as for N=32. None of the hills can be seen in Fig. 4.20 for N=32.

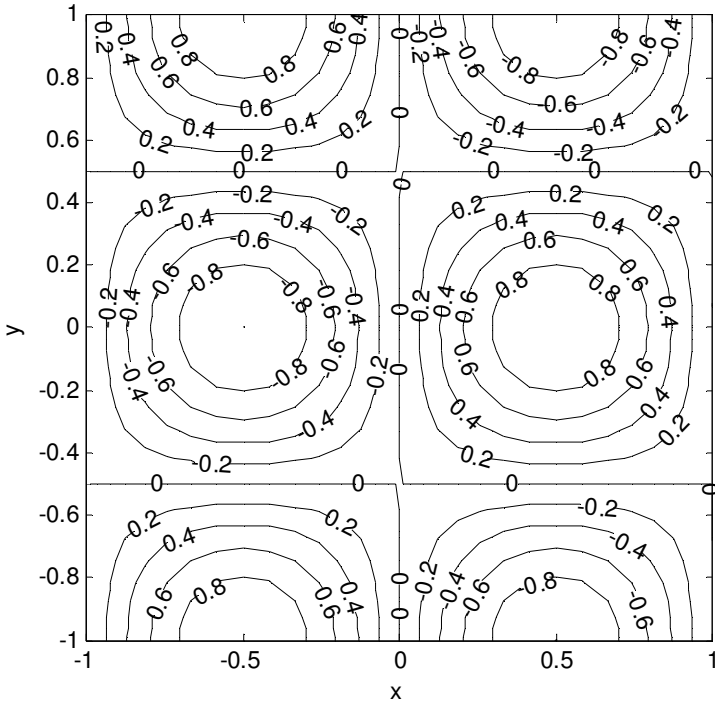
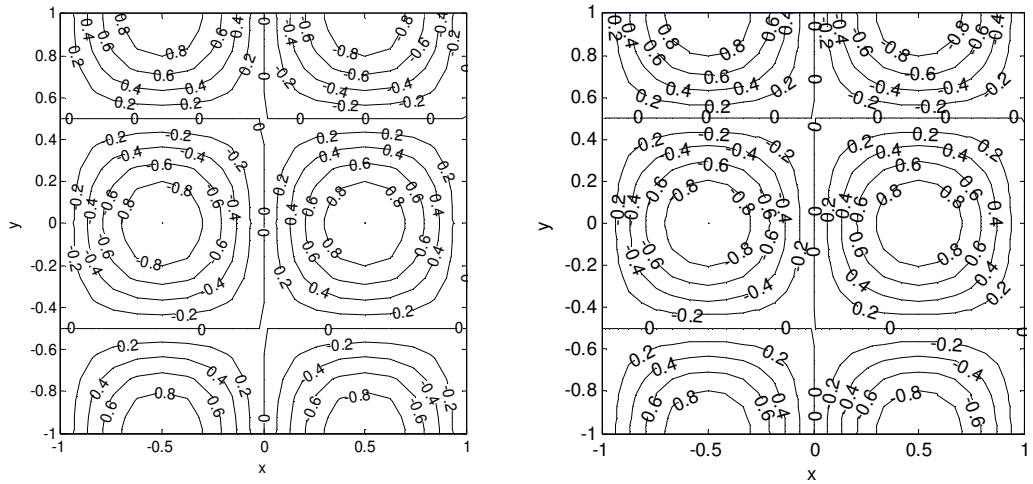


Figure 4.22 Spectral Element Solution graph of the Helmholtz equation $u_{xx} + u_{yy} + \lambda u = f(x, y)$ with 4 elements with the order of N=18. Element boundaries are same so they are not drawn to see the graph better.



N=12 N=18

Figure 4.23 The figures of 4.21 and 4.22 are drawn together to see the difference.

Another difference is zero lines surrounding the hill. For N=12 zero lines surround the negative hill at the negative x. For N=18 zero lines surround the positive hill at the positive x.

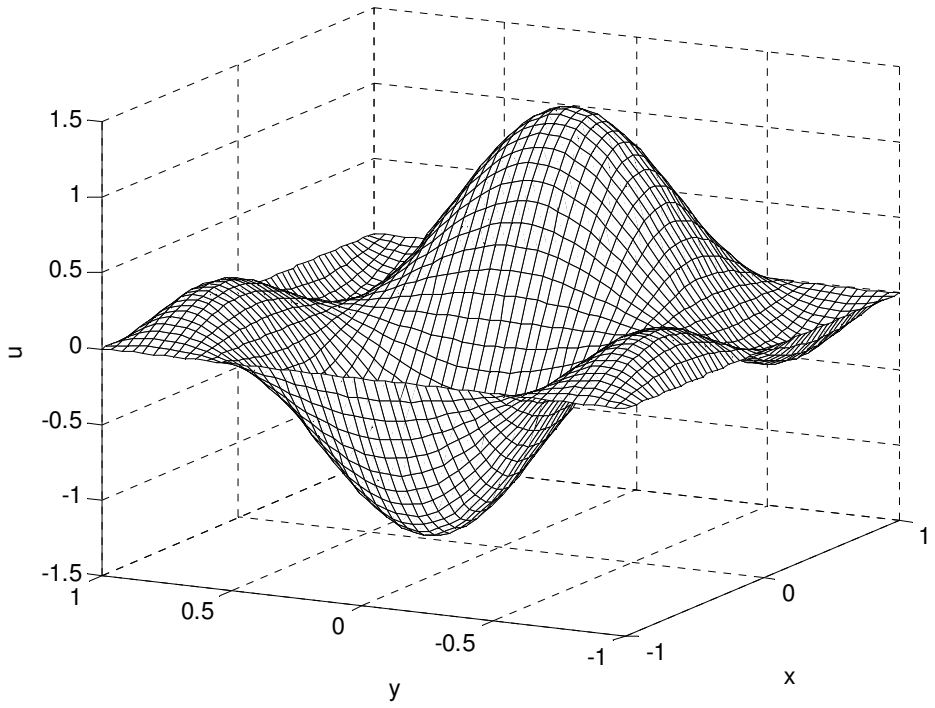


Figure 4.24 3D graph of the solution to the Equation (4.9), (4.9a) and (4.9b) to show the hills and hollows

The Dirichlet BC's are included in the solution by making the related row and column 1 and the others in the related row is zero. Therefore, the related rows of the RHS are equal to uexact.

CHAPTER 5

CONCLUSION

A spectral element solver program using MATLAB is developed for the solution of Poisson and Helmholtz equations. Element stiffness matrices and element mass matrices are evaluated for both of the isoparametric quadrilateral and rectangular elements to form steady diffusion and Helmholtz operators with this program.

The steady diffusion operator, which is equal to rectangular element stiffness matrix for a single element is used to solve the Poisson problems including both of the Dirichlet and Neumann boundary conditions which are homogeneous or nonhomogeneous. The steady diffusion operators of domains with more than one element are solved with the global stiffness matrix, which is the result of assembled element stiffness matrices.

The implementation of the quadrilateral element stiffness matrix evaluation is presented. Quadrilateral element formula is first tested on the mesh with rectangular elements. The results are discussed. The contour lines are smooth and there are no broken line is seen at the common element boundaries.

The local stiffness matrices of square element that are calculated with the quadrilateral element formulas are full matrices. When the rectangular element formulas are used, the matrices will have less band size and density. If a component of the matrix is less than a threshold value, it is declared as zero. With this threshold value of 1.0×10^{-13}, both of the element matrices will be the same and have the same sparsity shown in Fig 3.7. The computation zero is 1.0×10^{-13}.

The features, capacity and accuracy of curvilinear quadrilateral elements with single or two curvy edges investigated. As a result curvilinear quadrilateral elements have to be used to discretise curvy boundaries of the deformed domains.

The accuracy of the four element solution passes the single element for DOF=28. The four element reaches the computation zero for DOF=32 in Table 4.3. This means the assembly of the local stiffness matrices are done correctly.

Inviscid flow around a cylinder is a problem that the use of curvilinear quadrilateral is a must. From he contour graph of the stream function, it is seen that all contours are perpendicular to zero Neumann BC at the y=0 line and the arc line. The contour lines are smooth and no broken line is seen at the common element boundaries. Velocity potential formulation is a problem with nonhomogeneous Neumann boundary conditions. Here, two dimensional first order differentiation matrix is successfully evaluated and used. The obtained velocity potentials are perpendicular to the streamlines.

Helmholtz equation is solved with a single rectangular element. The formulas for the evaluation of Helmholtz operator are discussed. The implementation of Helmholtz equation is presented. Rectangular element formulas are used to evaluate Helmholtz operator. The results are plotted as both 3D and contour graphs.

The accuracy of the Helmholtz operator that is evaluated by quadrilateral element formulas for the domain with more elements is tested. All obtained u=0 lines intersect with a right angle for N=32. For N=32, the contour lines are seen clearly perpendicular to the x axis as a feature of the line of symmetry.

The element load vectors of Poisson and the Helmholtz equation are evaluated and assembled to construct the global load vector. This is also an important part of the solution of the Partial Differential Equations.

Developed spectral element code is benchmarked. For the implementation of spectral element methods, MATLAB built-in functions and MATLAB function libraries of the spectral methods in the literature are used. MATLAB is a powerful tool that

lowers the implementation work load. Especially, the implementations of matrix operations are easy with its built-in functions. The suitability of MATLAB for spectral element methods is investigated. MATLAB is ready for the spectral element method with its matlab built-in functions and its spectral methods library provided by scientists and researchers. Also, its plotting feature is useful for post-processing the results.

The main background of the spectral element methods is discussed. Therefore, this thesis can be easily developed and expanded for the solution of Stokes, Advection-diffusion and Navier-Stokes Equations.

REFERENCES

[1] L. N. Trefethen, Spectral Methods in Matlab, SIAM, Philadelphia, 2000.

[2] G. E. Karniadakis and S. J. Sherwin, Spectral/hp Element Methods for CFD, Oxford University Press, New York, 1999.

[3] M. O. Deville, P. F. Fischer and E. H. Mund, High-Order Methods for Incompressible Fluid Flow, volume 9 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2002.

[4] J. N. Reddy, An Introduction to the Finite Element Method, 2nd ed., McGraw-Hill, 1993.

[5] C. Sert, Nonconforming Formulations with Spectral Element Methods, Ph.D. Thesis, Mechanical Engineering Department, Texas A&M University, 2003.

[6] T.J Chung, Finite Element Analysis in Fluid Dynamics, 1st. ed., McGraw-Hill, New York, 1978.

[7] J. A. C Weideman and S. C. Reddy, A MATLAB Differentiation Matrix Suite, ACM Trans. Math. Software (TOMS), vol. **36**, pg. 465-519, 2000.

[8] H.I. Tarman, ES 702 Pseudospectral Methods Lecture Notes, METU, Ankara, 2005.

[9] C. A. Mavriplis, Nonconforming Discretizations and a Posteriori Error Estimators for Adaptive Spectral Element Techniques, Ph.D. Thesis, Department of Aeronautics and Astronautics, MIT, 1989.

[10] A. T. Patera, A spectral element method for fluid dynamics: Laminar flow in a channel expansion, J. Comp. Phys., vol. **54**, pg. 468-488, 1984.

[11] L. W. Ho, A Legendre Spectral Element Method for simulation of Incompressible Unsteady Viscous Free-Surface Flows, Ph.D. Thesis, Department of Aeronautics and Astronautics, MIT, 1989.

[12] B. D. Welfert, Generation of pseudospectral differentiation matrices I, *SIAM J. Numerical Analysis*, vol. **34**, pg. 1640–1657, 1997

[13] W.J. Gordon and C.A. Hall, Transfinite Element Methods: Blending Function Interpolation over Arbitrary Curved Element Domains. Num. Math., vol. **21**, pg. 109, 1973.

[14] G. Strang and G. J. Fix, An Analysis of Finite Element Method, Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[15] G.E. Karniadakis, E.T. Bullister, and A.T. Patera, A spectral element method for solution of two and three-dimensional time dependent NavierStokes equations, In Finite Element Methods for Nonlinear Problems, pg. 803, SpringerVerlag, Berlin, 1985.

[16] E.M. Rønquist, Optimal spectral element methods for the unsteady three dimensional incompressible Navier-Stokes equations, PhD thesis, Massachusetts Institute of Technology, 1988.

[17] M. Dubiner, Spectral methods on triangles and other domains, J. Sci. Comp., vol. **6**, pg. 345, 1991.

[18] S. J. Sherwin and G. E. Karniadakis, A triangular spectral element method; applications to the incompressible NavierStokes equations. Comp. Meth. Appl. Mech. Eng., vol. **123**, pg. 189-229, 1995.

[19] S. J. Sherwin and G. E. Karniadakis, A new triangular and tetrahedral basis for high-order finite element methods, Int. J. Num. Meth. Eng., vol. **38**, pg. 3775-3802, 1995.

[20] P. F. Fischer, Spectral element solution of the Navier-Stokes equations on high performance distributed-memory parallel processors, PhD thesis, Massachusetts Institute of Technology, 1989.

[21] L. M. Delves and C.A: Hall, J. Inst. Math. Appl., vol. **23**, pg. 223, 1979.

[22] L. M. Delves and C. Philips, J. Inst. Math. Appl., vol. **25**, pg. 177, 1980.

[23] D. Pathria and G. E. Karniadakis, Spectral element methods for elliptic problems in nonsmooth domains, J. Comp. Physics, vol. **122**, pg. 83-95, 1995.

[24] R. Pasquetti and F. Rapetti, Spectral element methods on triangles and quadrilaterals: comparisons and applications, J. Comp. Physics, vol. **198**, pg. 349-362, 2004.

[25] D. Wilhelm and E. Meiburg, Three-dimensional spectral element simulations of variable density and viscosity, miscible displacements in a capillary tube, Computers&Fluids, vol. **33,** Issue **3**, pg. 485-508, 2004.

[26] J. de Frutos and J. Novo, A postprocess based improvement of the spectral element method, Appl. Num. Math.,vol. **33,** pg. 217-223, 2000.

[27] O.Z. Mehdizadeh and M. Paraschivoiu, Investigation of a two-dimensional spectral element method for Helmholtz's equation, Journal of Comp. Physics, vol. **189**, pg. 111-129, 2003.

[28] G. E. Karniadakis, M. Israeli, and S. A. Orszag, High-order splitting methods for incompressible Navier–Stokes equations, *J. Computational Physics,* vol. **97**, pg. 414, 1991.

[29] S.J. Sherwin and M. Casarin, Low-Energy Basis Preconditioning for Elliptic Substructured Solvers Based on Unstructured Spectral/hp Element Discretization, Journal of Computational Physics, vol. **171,** pg. 394–417, 2001.

[30] I. Bica, Iterative Substructuring Algorithms for the p-version Finite Element Method for Elliptic Problems, Ph.D. thesis, Courant Institute of New York University, 1997.

[31] J. L. Guermond and J. Shen, A new class of truly consistent splitting schemes for incompressible flows, *J. Computational Physics,* vol. **192**, pg. 262-276, 2003.

[32] L. F. Pavarino and O. B. Widlund, Iterative Substructuring Methods for Spectral Elements: Problems in Three Dimensions Based on Numerical Quadrature, Computers & Math. with Applications, vol. **33**, No. 1/2, pg. 193-209, 1997.

[33] C. Evangelinos, S. J. Sherwin and G. E. Karniadakis, Parallel DNS algorithms on unstructured grids, Comput. Methods Appl. Mech. Engrg., vol. **184**, 401-425, 2000.

[34] S.J. Sherwin, G.E. Karniadakis, Tetrahedral hp finite elements: Algorithms and flow simulations, J. Comput. Phys., vol. **122**, pg. 191, 1995.

[35] S.J. Sherwin, Hierarchical hp finite elements in hybrid domains, Finite Elements in Analysis and Design, vol. **27**, pg. 109-119, 1997.

[36] Warburton, T. C., Spectral/hp methods on polymorphic multi-domains: algorithms and applications, PhD Thesis, Brown University, 1998.

[37] F. P. Incropera and D.P. DeWitt, Fundamentals of heat and mass transfer, 4th edition, John Wiley & Sons, New York, 1996.

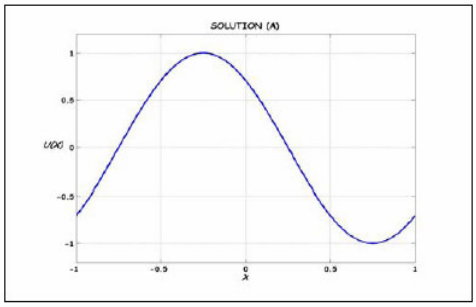
[38] C. Sert and A. Beskok, Spectral element formulations on non-conforming grids: A comparative study of pointwise matching and integral projection methods, Journal of Comp. Physics, vol. **211**, pg. 300-325, 2006.

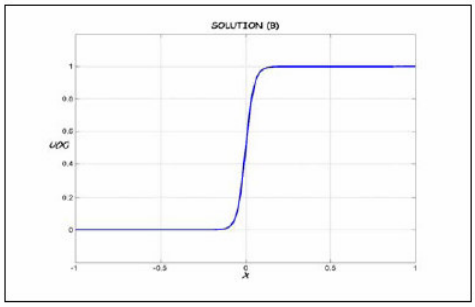
Appendix A: Results of 1D SEM solution

1. Consider the problems on $[-1,1]$:

(A) $u'' = -\pi^2 \cos\pi(x+\frac{1}{4})$ with $u(-1) = u(1) = -\sqrt{2}/2 \quad \Rightarrow \quad u(x) = \cos\pi(x+\frac{1}{4})$

(B) $u'' = -20^2 \tanh(20x)\,\mathrm{sech}^2(20x)$ with $u(-1) = 0$ and $u(1) = 1 \quad \Rightarrow \quad u(x) = \frac{1}{2}[\tanh(20x)+1]$





Use the following decomposition cut-off points at :

Problem (A) : **(i)** $[-1,1]$, **(ii)** $[-1,0,1]$, **(iii)** $[-1,-0.2,0.2,1]$, **(iv)** $[-1,-0.5,0,0.5,1]$

Problem (B) : **(i)** $[-1,1]$, **(ii)** $[-1,0,1]$, **(iii)** $[-1,-0.2,0.2,1]$, **(iv)** $[-1,-0.1,0.1,1]$

in an application of Spectral Element method.

This problem is the 5th homework of Tarman [8]. Figures A.1 and A.2 are my homework results of problem (A) and (B).

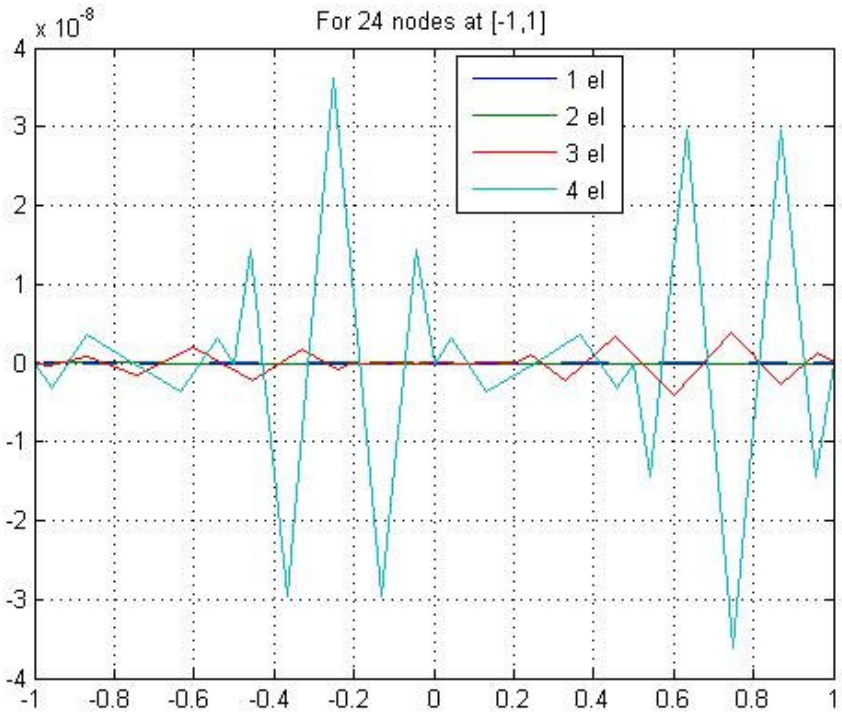


Figure A.1 Errors of the 1D spectral element solution of problem (A) for same number of nodes in the interval [-1, 1]. 1 elm for N=24, 2 elm for N=12, 3 elm for N=8 and 4 elm for N=6.

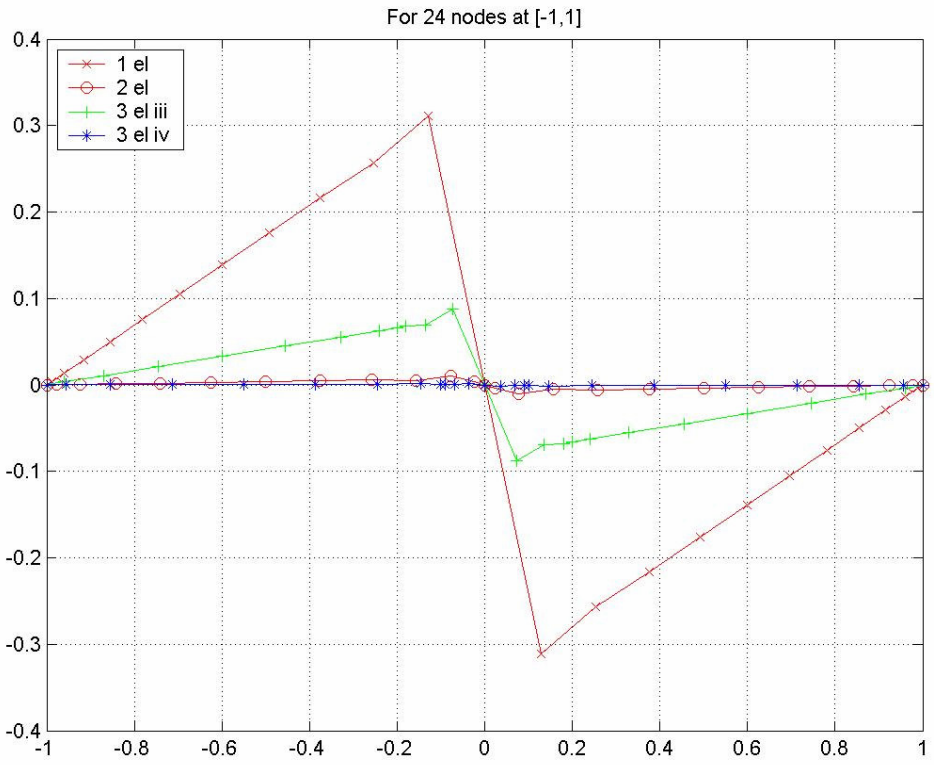


Figure A.2 Errors of 1D spectral element solution of problem (B) for same number of nodes in the interval [-1, 1]. 1 elm for N=24, 2 elm for N=12, 3 elm for N=8 and 4 elm for N=6.

Appendix B: 1D Expansion Bases

Most of the Appendix B belongs to Karniadakis and Sherwin [2]. Some of the parts and equations of Appendix B.2.2 are from Tarman [8] and cited in the Appendix B.2.2.

B.1 Expansion Bases: h-p type approximation

Essential part of constructing different expansion bases will be to introduce a standard elemental region within which the standard expansions will be defined. Assembly procedure of the global expansion bases from these local definitions. This type of elemental construction also provides an efficient way to numerically implement the spectral h-p element technique once integration and differentiation of polynomial functions have been understood.

B.1.1 Elemental Decomposition: h-type extension

In the h-type method, a fixed order polynomial is used in every polynomial and convergence is achieved by reducing the size of the elements. This is the so-called h-type extension, where h represents the characteristic size of an element (subdomain) means its length in one dimension. This type of extension aids in geometric flexibility, especially in high dimensions.

u is constructed from a class of functions which are C^0 continuous in the domain Ω, however at the boundary between the elements the derivative may be discontinuous. It can be shown that, upon convergence to the exact solution when h become smaller (as h→0) the jump in the in the derivative across inter element boundaries become zero and that the Neumann condition is exactly satisfied.

B.1.2 Polynomial expansion: the p-type extension

In the p-type method, a fixed mesh is used, it means h kept fixed and convergence is achieved by increasing the order of the polynomial expansion in every element. This is the so-called p-type extension where p represents the expansion order in the elements.

If the whole solution domain is treated as a single element, then the p-type method becomes a spectral method. [8, 16]

Recalling the definition of the standard element, Ω_{st} and the coordinate mapping $x(\xi_1, \xi_2)$ from Ω_{st} to an elemental region Ω^e, if $\mathcal{P}_{N_{\deg}}(\Omega_{st})$ denotes the space of all polynomials of degree $N_{\deg}$ defined on the standard element Ω_{st}. The discrete h-p expansion space χ_N is the set of all functions u_N which exists in H^1 and that are polynomials in ξ_1, ξ_2 within every element which is written as

$$\chi_N = \{ u_N \mid u_N \in H^1,\ u_N(x^e(\xi_1, \xi_2)) \in \mathcal{P}_{N_{\deg}}(\Omega_{st}),\ e=1,\ldots,N_{el} \} \qquad \text{(B.1.2.1)}$$

This definition allows both the mapping $x(\xi_1, \xi_2)$ and the polynomial order $N_{\deg}$ to vary within each element e thereby permitting both h-type refinement, which alters $x^e(\xi_1, \xi_2)$ capability and N_{el}, p-type refinement, which alters $N^e_{\deg}$. This is known as h-p type approximation.

Our main concern is polynomial expansion. Traditionally, the finite element method has always used polynomial expansions. This may be attributed to the historical use of Taylor series expansions which allow analytical functions to be expressed in terms of polynomials.

B.1.2.1 Modal and nodal expansions

To illustrate the difference between a modal and nodal polynomial expansion, three expansion sets are introduced which are denoted by $\Phi_p^A(x)$, $\Phi_p^B(x)$ and $\Phi_p^C(x)$ $(0 \le p \le \mathrm{P})$, in the region $\Omega_{st} = \{\mathrm{x} \mid -1 \le x \le 1\}$. All of these expansions represent a complete set of polynomials up to order P and are mathematically defined as

$$\Phi_p^A(x) = x^p, \ \mathrm{p} = 0, \ldots, \mathrm{P} \tag{B.1.2.2}$$

$$\Phi_p^B(x) = \frac{\prod_{q=0, q \neq p}^{P} (x - x_q)}{\prod_{q=0, q \neq p}^{P} (x_p - x_q)}, \ \mathrm{p}=0, \ldots, \mathrm{P} \tag{B.1.2.3}$$

$$\Phi_p^C(x) = L_p(x), \ \mathrm{p}=0, \ldots, \mathrm{P} \tag{B.1.2.4}$$

The first expansion set simply increase the order of the x in a monomial fashion. It is called moment expansion (each order contributing an extra moment the expansion). This basis is referred to as a modal or a hierarchical expansion because the expansion set of order *P*-1 contained within the expansion set of order *P*. There is a notion of hierarchy in the sense that higher-order expansion sets are built from the lower-order expansion sets.

The second polynomial $\Phi_p^B(x)$ is a Lagrange polynomial which is based on a series of *P*+1 nodal points x_q which are chosen beforehand and could be, for example, equispaced in the interval. The Lagrange polynomial is a non-hierarchical basis (that is, $\chi_P^N \not\subset \chi_{P+1}^N$) because it consists of P+1 polynomials of order P. This can be contrasted with hierarchical expansion $\Phi_p^A(x)$ which consists of polynomials of increasing order. The Lagrange basis has the notable property that $\Phi_p^B(x_q) = \delta_{pq}$, where δ_{pq} represents the Kronecker delta. This property implies that

$$u^N(x_q) = \sum_{p=0}^{P} \hat{u}_p \Phi_p^B(x_q) = \sum_{p=0}^{P} \hat{u}_p \delta_{pq} = \hat{u}_q \tag{B.1.2.5}$$

where it is seen that the expansion coefficient $\hat{u}_p$ can be defined in terms of the approximate solution at the point x_q.

The coefficients, therefore, have a physical interpretation in that represent the approximate solution at the points x_q. The point x_q are referred to a node and the Lagrange expansion basis is referred as a nodal expansion, linear finite elements are an example of a nodal expansion where the nodal points are at the ends of the domain.

A distinction between a nodal expansion and the collocation method (or collocation projection) is drawn. In the collocation method, the equation being solved is exactly satisfied at the collocation points where as in a nodal expansion the expansion coefficients represent the approximate solution at a given set of nodes. However, a nodal expansion can be used in different types of methods such as the Galerkin or collocation method. It must be remembered that an approximate solution using a nodal expansion base does not satisfy equation exactly at the nodal points.

The final expansion, $\Phi_p^C(x)$ is also a hierarchical or modal expansion however, in this case the expansion is the Legendre polynomial $L_p(x)$. By definition, this polynomial is orthogonal in the Legendre inner product

$$(L_p(x), L_q(x)) = \int_{-1}^{1} L_p(x) L_q(x) dx = \frac{2}{2p+1} \delta_{pq} \tag{B.1.2.6}$$

Orthogonality has important numerical implication for the galerkin method.

B.1.2.2 Choice of an expansion set

Choice of an expansion set is influenced by its numerical efficiency, conditioning, and the linear independence of the basis, as well as its approximation properties. To illustrate some of these factors, the three expansion set $\Phi_p^A(x)$, $\Phi_p^B(x)$ and $\Phi_p^C(x)$ will be considered in Galerkin projection.

The Galerkin or L^2 projection of a smooth function f(x) in the domain onto the polynomial expansion $u^N(x)$ is the solution to the following problem find $u^N \in \chi^N$ such that

The moment expansion $\Phi_p^A(x)$ produces a mass matrix which has components $(0 \le p, q \le P)$ of the form

$$M[p][q]=(\ \Phi_p^A, \Phi_q^A) = \int_{-1}^{1} x^p x^q dx = \left[\frac{x^{p+q+1}}{p+q+1} \right]_{-1}^{1}$$

$$= \begin{Bmatrix} \frac{2}{p+q+1}, & p+q \quad even \\ 0, & p+q \quad odd \end{Bmatrix} \qquad \text{(B.1.2.7)}$$

Therefore, when constructing M using this basis we need only calculate half of the components. However, the inverse will still be full and the cost of inverting the matrix is typically the dominant operation.

The second expansion $\Phi_p^B(x)$ is the Lagrange polynomial and so it is associated with a set of nodal points x_q. As a common feature of finite element the nodes are defined equispaced in the domain Ω_{st}, and so in the interval x_q=2q/*P*-1. If Gaussian quadrature is used, the mass matrix will be full. The reason discussed later. It causes twice more time and memory consumption. Therefore, the construction of the mass matrix $\Phi_p^B(x)$ is twice as expensive as $\Phi_p^A(x)$, although the matrix inversion won't cause any change. [2]

The third expansion $\Phi_p^C(x)$ is the Legendre polynomial. If the gauss Legendre quadrature nodes and weights are used for the numerical integration, the mass matrix will be diagonal. The components of the matrix are

$$M[\mathrm{p}][\mathrm{q}] = (\Phi_p^C, \Phi_q^C) = \int_{-1}^{1} L_p(x) L_q(x) dx = \frac{2}{2p+1} \delta_{pq} \qquad \text{(B.1.2.8)}$$

The construction and inversion of the diagonal mass matrix is the easiest as a result $\Phi_p^C(x)$ is the best of the three expansions if only it is thought from the numerical point of view. It is noted, however, that the basis can not be easily extended to an elemental decomposition which is globally continuos since the continuity constraints destroy the orthogonality of the global matrix structure.

If the local expansions were constructed so that only a few expansion modes have magnitude at an elemental boundary then the matching condition can be imposed far more easily. This type of decomposition is known as *boundary and interior decomposition*. Boundary modes have magnitude at one of elemental boundaries and are zero at all boundaries as a known feature of FEM. Interior modes sometimes known as bubble modes, only have magnitude in the interior of the element are zero along all boundaries. The equispaced Lagrange expansion $\Phi_p^B(x)$ already satisfies these conditions. If the nodal points did not include the end points then the boundary interior decomposition would not be possible.

The poor conditioning of the basis $\Phi_p^B(x)$ can be attributed to the high level of oscillations towards the end of the region, which can be seen at the graph Lagrangian interpolating polynomials for equispaced points in modes p=4 and p=6. These oscillations can be prevented by a better choice of nodal points so to obtain independently shaped modes with well-behave bounds, as shown by the modes of $\Phi_p^C(x)$ will be possible.

B.2 Nodal Polynomial expansions

Polynomial nodal expansions are based upon the Lagrange polynomials which are associated with a set of nodal points. The nodal points must include the ends of the domain if the expansion is to be decomposed into boundary and interior modes. The choice of these points, however, plays an important role in the stability of the approximation.

B.2.1 Lagrange polynomials

Given a set of P+1 nodal points, denoted by x_q ($0 \le p \le$ P), the Lagrange polynomial $h_p(x)$ is the unique polynomial of order *P* which has a unit value at x_p and is zero at x_q ($p \neq q$). This definition can be written as

$$h_p(x_q) = \delta_{pq} \tag{B.2.1.1}$$

where δ_{pq} is the Kronecker delta. Implementation clues will be in the implementation chapter. The Lagrange polynomial can also be written in product form as

$$h_p(x) = \frac{\prod_{q=0, q\neq p}^{P} (x - x_q)}{\prod_{q=0, q\neq p}^{P} (x_p - x_q)} \tag{B.2.1.1a}$$

If the polynomial of order *P*+1 with zeros at the *P*+1 nodal points x_q is denoted *g*(x), so $h_q(x)$ can be written as

$$h_p(x) = \frac{g(x)}{g'(x_p)(x - x_p)} \tag{B.2.1.2}$$

The Kronecker delta property shown in eqn (B.2.1.1) makes the Lagrange polynomial particularly useful as an interpolation basis. The Lagrange interpolant through the *P*+1 nodal points x_q is written as

$$\mathcal{I}h_p(x) = \sum_{p=0}^{P} \hat{u}_p h_p(x) \tag{B.2.1.3}$$

The interpolation approximation requires that $\mathcal{I}$u(x_q) = u(x_q) and, because of the property of $\Phi_p^B(x)$ and Equation (B.2.1.1), this means that $\hat{u}_p = u(x_p)$. The interpolation approximation can be written as

$$\mathcal{I}h_p(x) = \sum_{p=0}^{P} u(x_p) h_p(x) \tag{B.2.1.4}$$

If u(x) is a polynomial of order P then the relationship is exact.

B.2.2 Nodal p-type basis: spectral elements

A class of nodal p-type elements which is known and called as 'spectral element', due to Patera [10], use the Lagrange polynomial the zeros of the Gauss Lobatto polynomials. In the early version of the spectral element method the polynomials were Chebyshev type [15,10], but in later versions Legendre polynomials were selected for more accurate numerical quadratures [16]. This type of integration will be discussed.

The elemental mass matrix using this expansion is full if the inner product is evaluated exactly.

This type of integration is discussed in section 2.4.1, where using *P*+1 points give nodal points at the roots of the polynomial g $(\xi) = (1-\xi)(1+\xi)L'_P(\xi)$. Substituting this into Eqn. (B.2.1.2)the nodal p-type expansion in the standard element Ω_{st}.

$$\phi_p \mapsto h_p(\xi) = \begin{Bmatrix} 1 & \xi = \xi_p \\ (\xi-1)(\xi+1)L_P^{'}(\xi) & otherwise \end{Bmatrix} \quad 0 \le p \le P \qquad \text{(B.2.2.1)}$$

where Eqn (B.2.2.2) is used to deduce . The derivative of the Legendre polynomial $L_p^{'}(\xi)$ can be related to the Jacobi polynomial $P_{P-1}^{1,1}(\xi)$ using the following equation

$$\frac{d}{dx}P_n^{\alpha,\beta}(x) = \frac{1}{2}(\alpha+\beta+n+1)P_{n-1}^{\alpha+1,\beta+1}(x) \qquad \text{(B.2.2.2)}$$

A class of nodal p-type elements, which are known as spectral elements due to Patera[10], use the Lagrange polynomial through the zeros of the Gauss-Lobatto polynomials, i.e

$$g(\xi) = (1-\xi)(1+\xi)\frac{d}{d\xi}(P_{N_{\deg}}^{\alpha,\beta}(\xi)) \qquad \text{(B.2.2.3)}$$

where $P_{N_{\deg}}^{\alpha,\beta}(\xi)$ is the $N_{\deg}$ th order of Jacobi polynomial.

In the early version of the spectral element method, the polynomials were Chebyshev type ($\alpha = \beta = -1/2$) but in later versions Legendre polynomials ($\alpha = \beta = 0$) were selected for more accurate numerical quadrature. [8]

Now, that a nodal p-type expansion within each elemental domain can be constructed using the Lagrange interpolants through the zeros of the Gauss-Lobatto polynomials [8],

$$\phi_p(\xi) \ \rightarrow \ L_p(\xi), \ 0 \le p \le N_{\deg} \qquad \text{(B.2.2.4)}$$

the local differentiation [8]

$$\left.\frac{du_N(\xi)}{d\xi}\right|_{\xi=\xi_q} = \sum_{p=0}^{N_{\deg}} u(\xi_p)\frac{d}{d\xi}L_p(\xi)\Big|_{\xi=\xi_q} = \sum_{p=0}^{N_{\deg}} D_{qp}u(\xi_p) \qquad \text{(B.2.2.5)}$$

via the differentiation matrix D, and the local integration [8]

$$\int_{-1}^{1} u_N(\xi)d\xi = \sum_{p=0}^{Q-1} w_p u(\xi_p) + R(u_N) \tag{B.2.2.6}$$

via the Gaussian quadrature with the number of quadrature points Q(= $N_{\deg}$ +1), can be performed.Here, R(u_N)=0 if $u_N \in \mathsf{P}_{2N_{\deg}-1}$ for Gauss-Lobatto integration with appropriate weights w_p.

In this case, the elemental mass matrix will be [8]

$$\begin{aligned} M^e(p)(q) = (L_p, L_q) &= \int_{-1}^{1} L_p(\xi)L_q(\xi)d\xi \\ &\simeq \sum_{i=0}^{P} w_i L_p(\xi_i)L_q(\xi_i) = \sum_{i=0}^{P} w_i \delta_p \delta_q = w_p \delta_{pq} \end{aligned} \tag{B.2.2.7}$$

and turns out to be diagonal. [2] and [8]

Recall [8] that Legendre-Lobatto points $\{x_j\}_{j=0}^{N}$ are the roots of

$$\text{q(x)}=\left(1-x^2\right)\frac{d}{dx}P_N(x) \tag{B.2.2.8}$$

where $P_N(x)$ is the Legendre polynomial of degree [8]. Gauss Lobatto Legendre weights can be calculated as

$$w_j = \left\{ \begin{array}{ll} \dfrac{2}{N(N+1)} & j = 0, N \\ \dfrac{2}{N(N+1)}[P_N(x_j)]^{-2} & j = 1,\ldots N-1 \end{array} \right\} \tag{B.2.2.9}$$

may be used to set up a Gaussian quadrature approximation [8]

$$\int_{-1}^{1} f(x)dx \approx \sum_{j=0}^{N} w_j f(x_j) \tag{B.2.2.10}$$

which is exact for f(x)$\in P_{2N-1}$ (the space of polynomials of degree $\leq$2N-1)

A Newton iteration [8] can be set up

$$x_j^{(k+1)} = x_j^{(k)} - \frac{q(x_j^{(k)})}{q'(x_j^{(k)})} \tag{B.2.2.11}$$

to compute the Legendre-Lobatto points from the initial guesses $x_j^{(0)}$=cos(jπ/N) (Chebyshev points). It can be shown from [8] that

$$\mathrm{q}(x_j) = P_{N-1}(x_j) - P_{N+1}(x_j) = 0 \tag{B.2.2.12}$$

which follows from the recurrence relation [8]

$$(1-x^2)\frac{d}{dx}P_n(x) = \frac{n(n+1)}{2n+1}\left[P_{n-1}(x) - P_{n+1}(x)\right] \tag{B.2.2.13}$$

$$q'(x_j) = -(2N+1)P_n(x_j) \tag{B.2.2.14}$$

which follows from the recurrence relation [8]

$$P_n(x) = -\frac{1}{2n+1}\left[P'_{n-1}(x) - P'_{n+1}(x)\right] \tag{B.2.2.15}$$

The mathematical formulation of `Lobatto.m` of [8] is discussed.

Appendix C: Lobatto.m

```
function [w, y] = Lobatto(n)

% The Legendre-Lobatto points y(1:n+1) are the roots of f(x) = (1-
x^2) dP_n/dx,
% with the weights w(1:n-1) = 2/n(n+1) P_n(y(1:n-1))^-2 and w([0 n]
= 2/n(n+1).
% where P_n(x) is the nth degree Legendre polynomial.
%
% This routine uses Newton iteration to find the roots to 10 digit
accuracy.
% Only symmetric half is computed.
% The initial estimates are the Chebyshev points cos((pi/n)*(0:n)).
%
% By using the two relations
% (1-x^2)(dP_n/dx) = n(n+1)/(2n+1) (P_(n-1) - P_(n+1))
% P_n = -1/(2n+1) (dP_(n-1)/dx - dP_(n+1)/dx)
% we identify
% f = P_(n-1) - P_(n+1) and df/dx = -(2n+1)P_n.

if n==1, w = [1 1]; y = [1 -1]; return, end
if n==2, w = [1/3 4/3 1/3]; y = [1 0 -1]; return, end

s = 2/(n*(n+1));
m = ceil(n/2) - 1; % # of half-internal points except zero.
for i=1:m
z = cos((pi/n)*i); d = 1;
while abs(d) >= 5e-11
    Lnm1 = legendre(n-1,z); Lnp1 = legendre(n+1,z); Ln =
legendre(n,z);
    fz = Lnm1(1,1) - Lnp1(1,1); fpz = -(2*n+1)*Ln(1,1);
    d = -(fz/fpz); z = z + d;
end
yh(i) = z;
wh(i) = 1/Ln(1,1)^2;
end
if 2*ceil(n/2)==n,
    Ln = legendre(n,0); w0 = 1/Ln(1,1)^2;
    w = s*[1 wh(:)' w0 fliplr(wh(:)') 1];
    y = [1 yh(:)' 0 -fliplr(yh(:)') -1];
else
    w = s*[1 wh(:)' fliplr(wh(:)') 1];
    y = [1 yh(:)' -fliplr(yh(:)') -1];
end
```

Appendix D: elematrstmquad.m

```
function [delta,Xesit,Yesit,Jab]=elematrs(N,xna,xnb,xnc,
xnd,yna,ynb,ync,ynd,orderofedge,yeqa,yeqc,xeqd,xeqb)

[w, xs] = Lobatto(N); xs = xs'; w = w'; ys = xs;
[XS, YS] = meshgrid(xs,ys);
Dxs = poldif(xs,1); Dxs = Dxs(:,:,1); Dys = Dxs;
LXS1 = (1-XS)/2; LXS2 = (1+XS)/2;
LYS1 = (1-YS)/2; LYS2 = (1+YS)/2;
%  >> xna=0;xnb=0.5;xnc=0.5;xnd=0;
%  >> yna=0;ynb=0;ync=0.5;ynd=0.5;
% the vertices of a rectangular element for benchmark
if(orderofedge==0)
XA = xna*LXS1 + xnb*LXS2; YA = yna; % down face
XC = xnd*LXS1 + xnc*LXS2; YC = ync; % up face
YD = yna*LYS1 + ynd*LYS2; XD = xnd; % left face
YB = ynb*LYS1 + ync*LYS2; XB = xnb; % right face
else
XA = xna*LXS1 + xnb*LXS2; YA = subs(yeqa);
XC = xnd*LXS1 + xnc*LXS2; YC = subs(yeqc);
YD = yna*LYS1 + ynd*LYS2; XD = subs(xeqd);
% The value in the subs( ) paranthesis can
% be a symbolic equation or a number
YB = ynb*LYS1 + ync*LYS2; XB = subs(xeqb);
end
X = XA.*((1/2)*(1 - YS)) + XC.*((1/2)*(1 + YS)) + ...
XD.*((1/2)*(1 - XS)) + XB.*((1/2)*(1 + XS)) - ...
(xna)*((1/2)*(1 - XS))  .*  ((1/2)*(1 - YS)) - ...
(xnb)*((1/2)*(1 + XS))  .*  ((1/2)*(1 - YS)) - ...
xnd*((1/2)*(1 - XS))  .*  ((1/2)*(1 + YS)) - ...
(xnc)*((1/2)*(1 + XS))  .*  ((1/2)*(1 + YS));
```

```
Y = YA.*((1/2)*(1 - YS)) + YC.*((1/2)*(1 + YS)) + ...
YD.*((1/2)*(1 - XS)) + YB.*((1/2)*(1 + XS)) - ...
(yna)*((1/2)*(1 - XS))  .*   ((1/2)*(1 - YS)) - ...
(ynb)*((1/2)*(1 + XS))  .*   ((1/2)*(1 - YS)) - ...
(ynd)*((1/2)*(1 - XS))  .*   ((1/2)*(1 + YS)) - ...
(ync)*((1/2)*(1 + XS))  .*   ((1/2)*(1 + YS));
% XA = 0*LXS1 + 0.5*LXS2; YA = 0;
% XC = 0*LXS1 + 0.5*LXS2; YC = 0.5;
% YD = 0*LYS1 + 0.5*LYS2; XD = 0;
% YB = 0*LYS1 + 0.5*LYS2; XB = 0.5;
% X = XA.*((1/2)*(1 - YS)) + XC.*((1/2)*(1 + YS)) + ...
% XD.*((1/2)*(1 - XS)) + XB.*((1/2)*(1 + XS)) - ...
% (0)*((1/2)*(1 - XS))  .*   ((1/2)*(1 - YS)) - ...
% (0.5)*((1/2)*(1 + XS))  .*   ((1/2)*(1 - YS)) - ...
% 0*((1/2)*(1 - XS))  .*   ((1/2)*(1 + YS)) - ...
% (0.5)*((1/2)*(1 + XS))  .*   ((1/2)*(1 + YS));
% Y = YA.*((1/2)*(1 - YS)) + YC.*((1/2)*(1 + YS)) + ...
% YD.*((1/2)*(1 - XS)) + YB.*((1/2)*(1 + XS)) - ...
% (0)*((1/2)*(1 - XS))  .*   ((1/2)*(1 - YS)) - ...
% (0)*((1/2)*(1 + XS))  .*   ((1/2)*(1 - YS)) - ...
% (0.5)*((1/2)*(1 - XS))  .*   ((1/2)*(1 + YS)) - ...
% (0.5)*((1/2)*(1 + XS))  .*   ((1/2)*(1 + YS));
J = (Dxs * X')' .* (Dys * Y) - (Dys * X) .* (Dxs * Y')';
Area = w' * J * w
Xesit=X';
Xesit=Xesit(:);
I=eye(N+1);
Dx = kron(Dxs,I);
Dy = kron(I,Dys);
Yesit=Y';
Yesit=Yesit(:);
Dy = kron(Dys,I);
Dx = kron(I,Dxs);
Jlast=[(Dxs * X')' (Dys * X);(Dxs * Y')' (Dys * Y)];
Jab=J';
J11y=Jlast(1:(N+1),1:(N+1))';
J12y=Jlast(1:(N+1),N+2:2*N+2)';
```

```
J21y=Jlast(N+2:2*N+2,1:(N+1))';
J22y=Jlast(N+2:2*N+2,N+2:2*N+2)';
Jinv=[J22y(:)./Jab(:) -J12y(:)./Jab(:);-J21y(:)./Jab(:)
J11y(:)./Jab(:)];
J11inv=Jinv(1:(N+1)^2,1);
J12inv=Jinv(1:(N+1)^2,2);
J21inv=Jinv((N+1)^2+1:2*(N+1)^2,1);
J22inv=Jinv((N+1)^2+1:2*(N+1)^2,2);
Gk11=J11inv.*J11inv.*Jab(:)+J12inv.*J12inv.*Jab(:);
Gk12=J11inv.*J21inv.*Jab(:)+J12inv.*J22inv.*Jab(:);
Gk21=J21inv.*J11inv.*Jab(:)+J22inv.*J12inv.*Jab(:);
Gk22=J21inv.*J21inv.*Jab(:)+J22inv.*J22inv.*Jab(:);
D=poldif(xs,1);
w1=w(1:N+1);
I=eye(N+1);
for p=1:N+1
for q=1:N+1
L=I(:,q).*I(:,p);
RHS2(p,q)=dot(w1,L);end % RHS2 equals to 1D mass matrix.
end
s=sparse(kron(RHS2,RHS2));
G11=sparse(s*diag(Gk11));
G12=sparse(s*diag(Gk12));
G22=sparse(s*diag(Gk22));
G21=sparse(s*diag(Gk21));
D1=sparse(kron(I,D));
D2=sparse(kron(D,I));
DekT=[D1;D2]';
G=[G11 G12;G21 G22];
delta=DekT*G*[D1;D2];
```

Appendix E: streamfunction.m

```
clear
N=12;
syms XA XC YD YB
yeqc = [2 -1/2*XC+2];
xeqd = [4-2*YD 0];
xeqb =[4 -sqrt(1-YB.^2)+4];
yeqa = [sqrt(1-(XA-4).^2) 0];
elnod=elnode(N); % elnode.m is seen in Appendix E.
elnod=elnod([1 2],:); % we have two elements
stiff2=zeros(2*(N+1)^2-1*(N+1),2*(N+1)^2-1*(N+1));
% second element last node is 325
elnod(2,1:13)=[169:-13:13];
elnod(2,14:(N+1)^2)=[(N+1)^2+1:2*(N+1)^2-1*(N+1)];
% elnod is the connectivity matrix generated.
f3=zeros((N+1)^2,1);
load=zeros(2*(N+1)^2-1*(N+1),1);
[w,x]=Lobatto(N);
w1=w(1:N+1);
I=eye(N+1);
for p=1:N+1
for q=1:N+1
L=I(:,q).*I(:,p);
RHS2(p,q)=dot(w1,L);end % RHS2 equals to 1D mass matrix.
end
s=sparse(kron(RHS2,RHS2));
uex=[];allxx=[];allyy=[];
xna=[4-2/sqrt(5) 0];
xnb=[4 3]; % which one is first don't forget
xnc=[4 4-2/sqrt(5)]; % which one is first don't forget
xnd=[0 0]; % which one is first don't forget
yna=[1/sqrt(5) 0];
ynb=[1 0];
ync=[2 1/sqrt(5)];
ynd=[2 2];
for e=1:2
[delta,xx,yy,Jab,Jlast,Dxs]=elematrstmquad(N,xna(e),
xnb(e),xnc(e),xnd(e),yna(e),ynb(e),ync(e),ynd(e)...
,2,yeqa(e),yeqc(e),xeqd(e),xeqb(e));
u=sin(4*pi*sqrt((xx-2).^2+(yy-2).^2));uex=[uex u];
allxx=[allxx xx]; allyy=[allyy yy];
for p=1:(N+1)^2
for q=1:(N+1)^2
if (p==q) RHS(q)=Jab(p).*s(p,q).*f3(q);end
end
end
```

```
for i=1:(N+1)^2
for j=1:(N+1)^2
if(abs(delta(i,j))>=1e-15)
stiff2(elnod(e,i),elnod(e,j))=stiff2(elnod(e,i),
elnod(e,j))+delta(i,j);
end
end
end
for j=1:(N+1)^2 load(elnod(e,j))=load(elnod(e,j))+RHS(j);
end
end
[w,x]=Lobatto(N);
y=x;
[xx,yy] = meshgrid(x(1:N+1),y(1:N+1));
xx = xx(:); yy = yy(:);
nodenum1steldown = find( xx==-1);
% local xx's of the down edge.
% nodenum1steldown = find( xx==-1);
% 1steldownedge of the element
for i=1:13
stiff2(nodenum1steldown,:) = zeros(N+1,2*(N+1)^2-
1*(N+1)); stiff2(nodenum1steldown,nodenum1steldown) =
eye(N+1);
end
nodenum1stelup = find( xx==1); % local xx must be found.
for i=1:13
stiff2(nodenum1stelup,:) = zeros(N+1,2*(N+1)^2-1*(N+1));
stiff2(nodenum1stelup,nodenum1stelup) = eye(N+1);
end
localnodenum2ndelleft=find(yy==-1 ); for i=1:N+1
globalnodenum2ndelleft(i)=elnod(2,localnodenum2ndelleft(i
));
end
for i=1:13
stiff2(globalnodenum2ndelleft,:) = zeros(N+1,2*(N+1)^2-
1*(N+1));
stiff2(globalnodenum2ndelleft,globalnodenum2ndelleft) =
eye(N+1);
end
localnodenum2ndelright = find( yy==1); % local yy.
for i=1:N+1
globalnodenum2ndelright(i)=elnod(2,localnodenum2ndelright
(i));
end
for i=1:13
stiff2(globalnodenum2ndelright,:) = zeros(N+1,2*(N+1)^2-
1*(N+1));
stiff2(globalnodenum2ndelright,globalnodenum2ndelright) =
eye(N+1);
end
localnodenum2ndeldown = find( xx==-1); % local xx.
```

```
for i=1:N+1
globalnodenum2ndeldown(i)=elnod(2,localnodenum2ndeldown(i
));
end
for i=1:13
stiff2(globalnodenum2ndeldown,:) = zeros(N+1,2*(N+1)^2-
1*(N+1));
stiff2(globalnodenum2ndeldown,globalnodenum2ndeldown) =
eye(N+1);
end
for i=1:13
load(nodenum1stelup(i))=2;
end
for i=1:13
load(globalnodenum2ndelleft(i))=allyy(localnodenum2ndelle
ft(i),2);  end
un=stiff2\load;
min(un)
ert=un(elnod(2,:));
ert13e13el2=reshape(ert,N+1,N+1);
xx1=reshape(allxx(:,1),N+1,N+1);
xx2=reshape(allxx(:,2),N+1,N+1);
yy1=reshape(allyy(:,1),N+1,N+1);
yy2=reshape(allyy(:,2),N+1,N+1);
contour(xx2,yy2,ert13e13el2)
un169_13e13el1=reshape(un(1:169),N+1,N+1);
hold on
contour(xx1,yy1,un169_13e13el1)
axis([0 4 0 2])
line(xx2(1:13,1),yy2(1:13,1))
ert=un(elnod(2,:));
ert13e13el2=reshape(ert,N+1,N+1);
xx1=reshape(allxx(:,1),N+1,N+1);
xx2=reshape(allxx(:,2),N+1,N+1);
yy1=reshape(allyy(:,1),N+1,N+1);
yy2=reshape(allyy(:,2),N+1,N+1);
contour(xx2,yy2,ert13e13el2)
un169_13e13el1=reshape(un(1:169),N+1,N+1);
hold on
contour(xx1,yy1,un169_13e13el1)
axis([0 4 0 2])
line(xx2(1:13,1),yy2(1:13,1))

% command lines more than one row must be connected
because of the margins of the paper aren't enough. The
second row must be added to the end of the first row. Or
Use three blue points.
```