APPLICATION OF THE FEKETE SPECTRAL ELEMENT METHOD IN STRESS ANALYSIS OF PLATES INVOLVING HOLES AND COMPLEX GEOMETRIES

By

Justin Kyle Shannon

Arash Ghasemi
Director, Civil Infrastructure Laboratory & Research
(Chair)

James C. Newman III
Department Head and Professor of Mechanical Engineering
(Committee Member)

William H. Sutton
Professor of Mechanical Engineering
(Committee Member)

APPLICATION OF THE FEKETE SPECTRAL ELEMENT METHOD IN STRESS ANALYSIS OF PLATES INVOLVING HOLES AND COMPLEX GEOMETRIES

By

Justin Kyle Shannon

A Thesis Submitted to the Faculty of the University
of Tennessee at Chattanooga in Partial
Fulfillment of the Requirements of the
Degree of Master of Science:
Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

December 2020

ABSTRACT

In an effort to prove the effectiveness of a methodology that computes more efficiently than traditional FEM methods, this paper details a method that allows for the accurate and efficient solution of an elasticity problem using an *hp*-SEM code. The code can interchange basis functions to allow us to compare Lagrange with Fekete basis functions, which have the capability of high accuracy and efficiency. The code is applied to solve a traditional elasticity problem with an analytical solution to judge the accuracy on a course mesh. This gives us a method that provides greater accuracy than traditional FEM when comparing the same mesh, and higher efficiency compared to Lagrange bases comparing matrix condition numbers. This produces a flatter curve when varying *p*-order, which shows that higher orders than the 9th-order methods tested here are easily achievable with this technique. The flexibility of our code is shown by solving solutions with complex geometry and holes.

ACKNOWLEDGEMENTS

I would like to thank the many teachers at UTC who have supported me and my education at this institution since 2011. In particular, I would like to thank Dr. Will Sutton and Dr. Arash Ghasemi for their advising, mentorship, and teaching over the last several years, which made this thesis possible. Thanks must also be given to Volkswagen Chattanooga, for financing this endeavor, and my wife Allyson, who supported me during the many hours invested into this work and my education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Finite Element Method (FEM) is used in engineering to solve problems of complexities that are too complicated to be solved analytically. They work by approximating their way toward an accurate solution, however this can come at a large cost to computational overhead (time and computer capabilities) when using traditional FEM techniques. To reduce overhead, the refinement of the mesh can be reduced which will subsequently reduce the computational intensity. These traditional techniques, however, lose accuracy when decreasing the meshing refinement (h-refinement). This can be overcome by increasing the refinement of the basis functions which drive the approximation accuracy (p-refinement). However, increasing the p-order on these traditional Lagrangian hp-FEM techniques increases the complexity of the underlying matrix at extreme rates. By further modifying the basis functions to spectral element method (SEM) polynomials, better matrix conditioning, accuracy, and efficiency is gained, further increasing the complexity allowed with the same resources.

It is a common theme in literature that SEM is generally only applied to problems with smooth solutions, such as those found in fluid dynamics [2, 23, 38]. This paper details a method to allow for accurate, efficient solution of SEM at a higher range of p-orders and lower h-refinement. We will show the application of SEM to solid mechanics with elasticity solutions. We achieve this using a traditional, but portable hp-SEM code, which allows us to vary p-orders and also interchange basis functions. The ability to swap basis functions allows us to compare traditional Lagrange basis functions with Fekete basis functions, which have the capability of allowing for high p-order, low h-refinement SEM while maintaining

accuracy and efficiency.

Overview

Here, we describe how the Fekete basis functions are derived and then later will be applied to the solution of a two-dimensional elasticity problem. Next, we give a background on the formulation of elasticity equations for spectral/finite element methods. Then, the equations of elasticity are solved using Finite Element formulation. Then, the implementation of the theory into code is detailed. We then apply this code to solve a traditional elasticity problem with an analytical solution to judge the accuracy of the solution on a course mesh, where the solution method is validated for a thin rectangular plate under tensile stress. There is a hole at the middle of the plate. The solution is then reevaluated using the Fekete spectral element method and compared to the results of the original solution using traditional techniques where there is good agreement with the solutions. We vary the p-order and compare to Lagrange FEM by comparing the matrix condition numbers. We also show the flexibility of our code by proving that it can also solve solutions with complex geometry and additional holes. Then, we discuss the results of the solution validation and provide a conclusion and recommendations for future research.

Literature Review

Many researchers have worked to improve upon the fundamental concepts of FEM to improve the efficiency of simulations to require less resources, so larger problems can be solved. To go about this, researchers have developed many novel solutions to the solution of finite element problems. Generally this is done by moving to high order elements in some form or fashion. High order elements involve increasing the p-order, also known as the polynomial order, of each element, basis, and shape functions, therefore increasing the number of both

interior and boundary nodes per element, and for those elements on the boundary of an object, such boundary nodes should be curvilinear to match such a surface. Going to a high order element alone will not reduce the computation required, reducing the h-order, or elemental mesh refinement, should also be done to reduce the number of elements necessary to represent the geometry involved. Combining these methodologies create what we know as hp-FEM, however further technologies have been developed as well. Extending the hp-FEM technique by adopting spectral polynomials, we get SEM. In the following section, we will discuss the various background works of FEM, hp-FEM, SEM, and also other techniques and technologies related to these methodologies.

Regarding the necessity of curvilinear boundaries, Bassi and Rebay in their work discuss the application of FEM to the solution of the 2D Euler equations. Their technique uses a course unstructured triangular grid structure upon which they found that "the method is inaccurate at curved solid walls if a piecewise linear approximation of the geometry of the boundary is employed." To achieve proper levels of high order accuracy, a high order boundary representation is utilized with "a dramatic improvement in the accuracy of the numerical approximation"[3].

Poya, Sevilla, and Gil discuss a method for the a posteriori generation of high-order curvilinear meshes. This a posteriori method of mesh generation is what we use in our work, albeit in a basic implementation comparative to the method they describe. The general a posteriori mesh generation method works basically by generating an initial low order mesh, then applying a high order meshing routine over this by adding the boundary nodes, curving the boundary nodes according the the part geometry, and then applying the interior nodes within the element. Poya, et al, describe this method as advantageous over direct methods which generate high order meshes directly from the CAD model to take advantage of the maturity of low-order meshing algorithms. They expand on this by describing a technique

which, after the generation of the initial low order mesh, applies all of the high order nodes to the mesh, then deforms the boundary nodes of the mesh according to the CAD model. The displacements necessary to deform the mesh boundaries are then used in a solid mechanics technique to subsequently deform the interior nodes. This technique deforms the nodes such that the interior nodes curve and achieve the proper distribution according to the geometry and basis functions utilized [27]. Xie, et al, in their work on the generation of arbitrary order curved meshes for use in 3D FEM, again use the a posteriori method for mesh generation. One of their goals is to correctly eliminate and low quality or invalid elements made in the curving process of the linear mesh. They utilize a similar linear elastic model, but employ a Fekete nodal distribution in their algorithm, as we do in our technique, since they "substantially reduce the interpolation error compared to equally-spaced nodal distributions when very high-order approximations are considered, say $p \geq 4$" [39].

Looking over the history of the further development of finite element style methods, we find several novel approaches to the solution of these problems. One of which is the Virtual Element Method (VEM). According to Beirão da Veiga, Brezzi, and Marini, VEM is an FEM based solution that starts off with much of the same parts as FEM, a mesh with numerous elements, basis functions, quadratures, etc. with some additional functions and selected such that the local stiffness matrices are able to be solved without the solution of the local PDEs. To explain the term Virtual Element Method, they say, "The label *virtual* depends on the fact that some of the basis functions are not explicitly known." The authors, in their paper "Virtual Elements for Linear Elasticity Problems", then go on to describe the application of VEM to linear elasticity and provide some test cases with their results [4].

Artioli, Beirão da Veiga, and Dassi, then go on to expand VEM to curvilinear elements, as we see in high order FEM methods. The authors explain that the adaptation of VEM to curvilinear elements is straightforward, describing the modification as, "By changing the definition of the edge-wise space, the method is able to accommodate for

general boundaries described by a given parametrization (for instance obtained by CAD)." In code, this is only a change to the quadrature rules. This is advantageous over isogeometric analysis (IGA) and isoparametric FEM requiring no further processing (parameterization or positioning) to adapt the curvilinear elements to the analysis framework. After presenting the theory, the authors describe the test cases which involved testing multiple quadrature rules, as well as several application based problems and results that show good convergence with only minimal extra computational expense over the standard VEM method [1].

Another concept for the solution of FEM problems is the Finite Cell Method (FCM). FCM is a hybridization of the fictitious domain method and high order FEM, in which the domain is generalized in such a form that a structured mesh can be generated that greatly simplifies the meshing process over traditional techniques. The use of a structured mesh is significantly different from the unstructured techniques we use. It is less flexible than an unstructured mesh and can be harder to generate, but that comes with the advantages of generally easier algorithms when it comes to computation. The implementation of the structured mesh in FCM is highly rasterized compared to the curvilinear meshes we use as well, which does not conform to the boundary exactly due to the standard element shape used. This gives generally less accuracy at the boundary comparatively unless significant refinement is applied at the boundary. Düster, Parvizian, Yang, and Rank describe a method that applies FCM to 3D solid mechanics. After describing and deriving the FCM process, the authors showcase several examples of the application of FCM in solid mechanics. Of particular note is the thick wall plate with hole example, which is similar to the example we use for validating our results. They perform multiple tests from $p = 1...8$ to which they see good agreement with the analytical solution. They showcase another example applying FCM to Computed Tomography (CT) scan data, which shows the power of their voxel based meshing routine on highly complex geometries which they compared with the commercial software *Abaqus* with good agreement. These examples showcased the flexibility of this

method and the versatility of its applications [8]. Schillinger, Kollmannsberger, Mundani, and Rank then extend this method to nonlinear FCM, which to this point have only been applied to linear elasticity problems. They point out that the traditional methods of p-FCM, as previously reviewed, do not perform well in nonlinear analyses, generally due to the rasterized boundaries causing oscillations. To get around this, the authors adapt B-splines to the boundaries to reduce the oscillatory effects. The use of B-splines still maintains the primary advantage of FCM, the ease of meshing, but adds generally curvilinear boundaries and sets the method up for greater success on the solution of nonlinear problems [30]. Schillinger, Cai, Mundani, and Rank then take FCM further into nonlinear analysis by applying T-splines in place of B-splines and exhibit its application with the modal analysis of a ship's propeller and with stress analysis of an open cell foam structure. They prove that an increase in p-order "improves the reproduction of the geometrically nonlinear behavior of the foam." These examples well exhibit the flexibility of this technology for complex geometry [29].

Isogeometric Analysis (IGA) is another such technique for the solution of FE-like problems. IGA modifies FEM by utilizing the same basis functions for both geometry and analysis, which gives exact representation of the geometry instead of the approximated representation seen in classical FEM. Rank, et al, adapted IGA to FCM to create a hybrid technique. When adapting CAD models to IGA, not all NURBS (Non-Uniform Rational B-Splines) surfaces adapt well to the IGA approach, being that, while there are multiple ways of constructing a geometry in CAD, not all methods are adaptable and parts should be modeled to be aware of the analysis technique for successful IGA. But as previously discussed, FCM is adaptable to nearly any geometry, so it can work well as an interface to work with the IGA approach by creating a fictitious boundary that IGA can then represent easily. The authors give some examples demonstrating this technique and state that the

expansion of the technique to other than Neumann boundary conditions will be the subject of future research in this topic [28].

Looking further at IGA purely, Hughes, Cottrell, and Bazilevs explain the method as it was developed. They explain it as having commonality with both FEM and meshless methods, but it's primary goal is to be mostly geometrically based so as to retain exactness with the geometry no matter the density of the mesh. As such, it is intended to tie closely with CAD. The authors claim that roughly 80% of the time it takes for analysis is involved in the mesh creation process, as such, creating a technique that simplifies this process was an important necessity for performing analysis in industries where CAD changes quicker than meshes can be created. Adding to this, the approximation in the meshes can be less than perfect, which can compound into accuracy problems in the analysis. To achieve this goal and get around the deficiencies of traditional techniques, the authors developed IGA to utilize NURBS to match the CAD geometry and then construct a course mesh of NURBS based elements from that. The authors then detail the derivation of NURBS and the analysis framework built upon it from there, before showcasing the application of IGA, including the example with a hole in thin plate, like we use in our technique detailed in this paper. They tested this solution through multiple levels and techniques of refinement to find the best strategy for IGA refinement, which they determine to be k-refinement, a unique parameter for IGA which refers to the elevation of curve order coupled with insertion of unique knot value (knots being the analog for mesh nodes/edges in classical FEM), therefore increasing the continuity of the functions used. They also demonstrated some examples showing how this technique can be applied for fluid analysis as well, showcasing the versatility of this technique [16].

As a segue into the advanced FEM and Spectral Element Methods (SEM) that we will be discussing for the remainder of this text, we review that authors Gervasio, Dedè, Chanon, and Quarteroni have carried out a comparison of IGA and SEM in their article, *A*

Computational Comparison Between Isogeometric Analysis and Spectral Element Methods: Accuracy and Spectral Properties. They explain that these methods "can be seen as two different paradigms for high order approximation of partial differential equations," although the intent of IGA was not for that. It can be used as such with the use of specific basis functions. The authors explain further that both can be applied to the Galerkin method and in both "the induced approximation error decays more than algebraically fast with respect to the local polynomial degree." After performing extensive testing of the two methods, the authors conclude that although the two methods are comparable when examining their accuracy, the computational costs are much different, matrix assembly for SEM required much less resources than IGA when comparing CPU time and memory. When examining the condition number of the matrices over p, SEM again outperforms IGA, with algebraic versus exponential growth respectively [11]. It can be seen from this that the utilization of SEM over IGA, at least from a computational perspective, has advantages.

Explaining further FEM and SEM, Babuška explains that "there are three versions of the finite element method," h, p, and hp. h-FEM is the classical approach to FEM which uses mesh refinement alone to achieve accuracy. p-FEM uses a fixed mesh and refines the p-order of the mesh to achieve accuracy. hp-FEM is the combination thereof. Regarding the implementation, the author explains that h- and hp-FEM gives great flexibility in the change of degrees of the elements and that this has to be implemented in the program's architecture, generally through the use of hierarchical shape functions. p-FEM, with its larger elements also requires attention to the curved boundaries. Babuška suggests that it is advantageous to describe the boundary exactly using polynomials and not approximated, and for this reason he states that "the architecture of the p version finite element program has essential differences from the h version." Judging the computational cost and complexity of the methods, it can be seen in his results that increasing p increases the computational time, but after about $p = 4$, it increases somewhat linearly. The author concludes from this that

"the conventional h version with $p = 1, 2$ will practically never lead to an accuracy of order of 1 percent for any reasonable computational effort if the problem is not very simple." Speaking to how robust this method is, he states that better accuracy is achieved with the same mesh by increasing the order p. Babuška then shares the industrial experience with p- and hp-FEM. He explains that in industry, users are generally more concerned with the human time factor than computational time, but overall time elasped to give a correct solution is still important as deadlines can be critical in industrial settings. In addition to this, the author claims that easy learning, flexibility of mesh design, and rapid convergence and flexibility are important factors for industry for such a program. The author also addresses the similarity of p- and hp-FEM to spectral methods (SEM). He defines SEM as "expanding the solution of the problem in high order Fourier or polynomial series, the coefficients of which are determined by weighted residual projections." He says that the use of polynomial approximation in SEM closely matches that of p-FEM. He notes that SEM (at the time of publication) had mostly only been applied to fluids and problems with smooth solutions [2]. Later in this thesis, it will be demonstrated how SEM can be applied to solid mechanics problems with complex geometry, far from the smooth type solutions Babuška describes.

Netz, Düster, and Hartmann provide a comparison between high order and low order mixed FEM formulations. They compare two trains of thought when it comes to solving problems. They claim there is an "h-community", who uses linear mixed element formulations to solve small strain problems, and a "p-FEM community", who utilizes fewer elements with increased polynomial order and employ "Jacobi polynomial-based hierarchical shape functions" for these same problems. The authors compare the methods describing that error is removed much faster in the p-FEM approach, but this approach comparatively has a much higher amount of implementation due to the coupling to neighboring elements. They conclude their comparison by describing the results of their research, stating for finite strain hyperelasticity that traditional linear methods perform poorly converging to the correct

results. Linear mixed elements improve upon this, but only for stresses on problems that are bending dominated. They have early failure when considering large deformations. p-FEM overcomes these difficulties for these situations and are generally applicable to any smooth constitutive model [22].

Expanding on the use of high order FEM and SEM in solid mechanics, Bittencourt, Vazquez, and Nogueira describe a method of applying high order FEM to nonlinear structural elastic problems. In their paper, the authors compare two types of high order bases, the Sherwin-Karniadakis and the fully tensorial bases for triangles (which the authors refer as Bittencourt basis), by their condition numbers, then apply the Bittencourt basis to nonlinear elasticity and compares that to an ANSYS solution. The high order mesh uses 23 $p = 5$ elements which showed superior rates of convergence compared to the low order ANSYS mesh, which uses 303 elements. The two solutions had good agreement, so the authors concluded that the high order technique is valid for use in nonlinear elasticity [5].

Tews and Rachowicz explore an automatic method for hp-FEM to generate adaptive meshing for 3D linear elasticity. The method they propose is intended to reduce the meshing process to just a one-click approach of applying the CAD model and having a solution that converges to zero approximation error. The authors discuss that this is generally possible with hp-FEM, but there is large amounts of memory required to process the high order elements. The authors also conclude that the generation of coarse elements for complex geometry may not be easy [36].

Of high interest in the realm of the application hp methods is the question of how high order is actually needed for the efficient and accurate solution of FEM/SEM problems. Mitchell explains that high order FEM is more efficient than low order FEM for the solution of partial differential equations (PDEs), but how high is measureably beneficial when examining accuracy versus computational resources? He explores three different problems with three different techniques each, smooth solutions, steep gradients, and problems with singularities

and applies h-, p-, and hp-FEM techniques to determine which has advantages for each type of problem and to what degree of refinement is necessary. The author explains that at lower orders, increasing the order drastically changes the convergence rates, but when the order is high, increasing the order sees much lower change in convergence rates. Increasing the order of FEM also increases the computational requirements by introducing higher degree basis functions, high order quadrature, and denser global matrix. Mitchell concludes that traditional h-FEM is suitable enough for low accuracy requirements near 1% relative error, the singular problems had no need for higher than $p = 2$ and nonsingular problems only beneficial up to $p = 4$. He continues that for high accuracy, higher orders are justified, but only up to roughly $p = 10$, or $p = 8$ when looking solely at optimal computational time, except in cases of smooth problems, where p-refinement is most effective [21]. In this thesis, we examine p-refinement the most, and only have demonstrated up to $p = 9$, though the method shows capabilities of much higher p, without significant resource requirements due to the use of specular elements and the Fekete basis functions which was not examined in this article.

Melenk, Gerdes, and Schwab describe hp-FEM as flexible in the combination of h- and p-refinement, which makes it superior to h- and p-FEM individually as well as SEM, but it is perceived as computationally expensive due to stiffness matrix generation and slow solution algorithms. They go on to state that h-FEM has simple stiffness matrix generation and efficient solvers available and that SEM has a reputation of speed due to several techniques including sum factorization, preconditioning, and weighted collocation in place of quadrature. To combat this perception, the authors developed a fast stiffness matrix generation technique using a hybridization of hp-FEM and SEM, which they call hp-spectral Galerkin FEM. This method utilizes decoupled quadrature rules from the elemental p-order, localized mesh refinement using hanging nodes, parallelization, decoupled geometric representation from the finite element space (which allows exact geometric representation)

and *hp*-isoparametric elements. From SEM, they integrate to *hp*-FEM the use of sum factorization and the adaptation of the shape functions to the quadrature rule. They are able to prove that these efforts show a speed up from orders of $p = 5$ through $p = 10$. This comes at the expense of worse matrix condition numbers compared to SEM alone, but with comparable CPU times [20]. Overall this generates a technique similar to that which we use, though in this thesis, we demonstrate its application to elasticity.

So far in describing the different methods used to further develop traditional FEM techniques we have looked at VEM, FCM, IGA, and high order FEM techniques. We will finish this discussion by focusing on SEM, which is the predominant technology we utilize in this thesis. Pavarino, Zampieri, Pasquetti, and Rapetti explain that Spectral and *hp*-FEM are several of the most successful high order methods used for the solution of PDEs. The authors note that generally SEM is using quadrilateral element structured meshes, but researchers have made using triangular element unstructured meshes feasible. Like us, they are using Fekete basis functions coupled with triangular elements on the unstructured meshes for analysis. The authors explain that the discrete matrices produced by these methods generally have poor conditioning which makes the implementation of efficient iterative solvers difficult. To get around this difficulty, the authors have developed preconditioners that give generous overlap between domains. This gives convergence rates that are independent of the number of subdomains and, because of the overlap, the *p*-order. This comes at the cost of being limited to planar regular structured meshes [25]. Pasquetti and Rapetti have also analyzed the differences in SEM between the use of triangular and quadrilateral meshes. They state that the primary advantage of SEM is the exponential convergence of smooth solutions, but that SEM doesn't handle complex geometries well. To combat this, there have been numerous techniques to work around this drawback. One being the use of quadrilateral elements. These elements, however, still cannot handle the highly complex geometry, so the use of triangles are necessary in these situations. Past research had attempted to transform

the quadrangle technique for triangles by using coordinate shifting or mesh restrictions, but the author explains their use of Koornwinder–Dubiner polynomials and Fekete points to implement triangles efficiently. The authors perform a variety of tests to check the convergence capabilities of triangle based SEM. They show similar performance (though with slightly higher condition numbers) compared with quadrilateral based SEM. The authors conclude that although triangle based SEM has better capabilities of handling complex geometry, their usage of Fekete points gives several drawbacks including: a lack of Gauss quadrature rules and the computation of derivatives is such that it is more computationally expensive [23]. Pasquetti and Rapetti continue their research of unstructured SEM meshes with Fekete based elements with an analysis of four different adaptations to this routine. They propose TSEM-1 through TSEM-4. TSEM-1 they describe as the general Fekete triangular SEM technique with Lagrange polynomials. TSEM-2 extends TSEM-1 using a semi-analytical integration rule. TSEM-3 uses a Gauss point integration rule. TSEM-4 is TSEM-3 with mass lumping. Their testing shows that TSEM-1 and TSEM-4 do not yield satisfactory results in regards to the collocation method with a sample PDE with Dirichlet boundary conditions applied, but TSEM-2 and TSEM-3 do. TSEM-2 however had a longer computational time. Continuing their study with TSEM-3, they determine the method has similar conditioning behavior as *hp*-FEM. The authors conclude that this method with Fekete approximation points and Gauss quadrature points with derivatives gives a computational efficiency not usually seen with SEM [24].

Taylor and Wingate continue with the use of Fekete triangle based SEM. Again, the authors explain that traditional SEM uses quadrilateral elements. They tend to give a diagonal mass matrix, which allows for computational efficiency. They explain that the diagonal mass matrix has only been achievable on structured quadrilateral meshes due to the Gauss-like quadrature rules which are believed to not exist with triangles. To get around this deficiency, the authors propose a technique for diagonal mass matrix SEM based on Fekete

points. They conclude with a comparison of the Fekete method to the Dubiner method, and while the Fekete does not have as much accuracy, it converges at the same rate with lower computational cost due to not having the need of inverting the mass matrix [35].

Konovalov, Vershinin, Zingerman, and Levin discuss the implementation of SEM into a computer aided engineering (CAE) system for the solution of elasticity problems on curvilinear meshes. The authors detail that SEM is advantageous over FEM due to its ability to maintain high approximation accuracy with small numbers of elements. SEM has no need to rebuild or refine meshes to check convergence either since the mesh can remain in its initial state by changing the order of the elements. They describe that its efficiency in parallelization makes it attractive for use in industry, describing a case where SEM was applied to a complex solution with quick solution by GPU computations. The authors claim that SEM's adoption in industry is hindered by lack of proper mesh generation. For 3D, SEM was initially developed for hexahedral elements (quadrilaterals in 2D), but meshing requirements of industrial problems with more complex geometry require a mixture of both hexahedral and tetrahedral elements (triangular elements in 2D). This is known as a mixed element formulation, which is what the authors implement for SEM in this article. The authors detail that implementing elasticity theory to SEM is much like that of FEM, "such as discretization of equilibrium equations in the integral form; selection of quadrature for calculation of integrals; building of local matrices of stiffness, mass, and damping for each element; assembling global matrices of stiffness, mass, and damping." They go on to state that most mesh generation tools, however, are only capable for generating first or second order meshes, which requires post processing to increase the order for SEM. One advancement that they make is the implementation of a fast mesh connectivity algorithm for SEM, which uses the initial low order connectivity graph to generate the high order version more efficiently. Their results show that mesh convergence is much faster when compared to FEM, and

application to an elasticity problem shows only an error of 1% with SEM compared to 10% with first order FEM and 2% for second order FEM [18].

Vos, Sherwin, and Kirby describe methods of determining the optimal order of high order SEM to apply for greatest efficiency in terms of computational cost. The authors describe several different communities of thought in the FEM community. The traditional h-FEM community, who may consider up to 4th polynomial order as high order. The spectral/hp community, who regularly use orders as high as 15, and global spectral method community who may consider 16th order to be low. Besides the definition of high versus low order, these communities have different approaches to the efficient solution of problems; the h-community favoring sparse matrix techniques and the spectral community favoring elemental operators versus global. And the global spectral community "often make use of the tensor-product approximations where products of one-dimensional rules for integration and differentiation can be applied." The authors note that these techniques often perform poorly when used outside their typical environments, strengthening the perceived incompatibility between the techniques. The authors look into each of these techniques to explore which are the most versatile to move up and down orders of p and which are most optimal for order p discretization regarding CPU times. After analyzing the different methods, the authors conclude that moving between orders of p from 1 to 15, it is most optimal to switch between three efficiency strategies. For low orders, use of global matrices is most efficient. For high orders, use of sum factorization. For the middle ground between low and high order, local matrix evaluation is most efficient. The exact boundary between these points, however, is dependent on element shape and computers used. Evaluating the most optimal run times, the authors concluded that a $p = 6$ solution on smooth problem provided the minimum CPU time with a 10% error. For non-smooth, the authors determined that "$p = 5$ was optimal, thereby promoting the use of high-order expansions for problems with corner-type singularities, at least when using a radical mesh distribution" [38].

CHAPTER 2

MATH FORMULATION

In this chapter, we will discuss the formulation and proof of the Fekete spectral element method and the application of this method to the solution of elasticity problems. The following theory on the Fekete spectral element methods and derivation was written by Dr. Arash Ghasemi and is included here for the theory to be self contained [14, 13].

Derivation of d-dimensional spectral basis functions over an arbitrary convex/concave hull

The space of d-variate polynomials obtained by product of one-dimensional monomials is represented by

$$f_Q = \prod_{i=1}^{d} {x_i}^{d_i}, \quad d_i = 0, \ldots, (n_i - 1), \tag{2.1}$$

which has the dimension $N = \prod_{i=1}^{d} n_i$ defined in $\Omega = \prod_{i=1}^{d} \mathbb{R}_i \subset \mathbb{R}^d$ where $\dim \mathbb{R}_i = n_i$. The constrained subspace of Eq. (2.1) for $n_i = n$ is the space of polynomials with the maximum degree (n-1) denoted by

$$f_P = \prod_{i=1}^{d} {x_i}^{d_i}, \quad \sum_{k=1}^{d} d_k \leq (n-1), \tag{2.2}$$

where $N = \dim(f_P) = 1/d! \prod_{k=1}^{d}(n + k - 1)$. For sufficiently large number of *interpolation points* in Ω

$$X = \{\hat{x}_i\} \subset \Omega, \quad 1 \leq i \leq M, \quad M \gg N, \tag{2.3}$$

the transpose of the rectangular Vandermonde matrix can be constructed columnwise by using f defined in either in Eq. (2.1) or Eq. (2.2) as follows

$$\mathcal{V}^T = [[f(\hat{x}_1)], [f(\hat{x}_2)], \ldots, [f(\hat{x}_M)]] \in \mathbb{R}^{N \times M}. \tag{2.4}$$

Definition 1. *The large set of points $X = \{\hat{x}_{i=1\ldots M}\}$ in Eq. (2.3) are the candidate points. The approximate Fekete points $X_F = \{\hat{x}_{i=1\ldots N}\}$ are selected amongst these points such that they mimic the Gauss-Lobatto quadrature points inside Ω. For more details on construction of these points, refer to § 5.*

We are interested to find X such that the Lagrange polynomials constructed by

$$(\hat{x}_i) = \delta_{ij}, \quad 1 \leq i, j \leq N, \quad \hat{x}_i \in X, \tag{2.5}$$

have small Lebesgue constant defined by the operator norm of the d-dimensional interpolation

$$\Lambda_N(T) = \max \left(\sum_{i=1}^{N} |\psi_i(x)| \right), \quad x \in \Omega, \tag{2.6}$$

which determines an upperbound for the interpolation error

$$\|u - I(u)\| \leq (1 + \Lambda_N(T)) \, \|u - p^*\| \tag{2.7}$$

where $\|u - I(u)\|$ is the norm of the difference between the exact value of u and an interpolated value $I(u)$ given as

$$I(u)(x) = \sum_{i=1}^{N} \ _i(x) u_i, \tag{2.8}$$

at any point $x \in \Omega$ and p^* is the optimum interpolant, yielding the best approximation of u at x. The existence of a polynomial interpolant for u is guaranteed by Weierstrass approximation theorem* and it can be shown that the optimal interpolant also exists and is unique†. For one-dimensional equally distributed points, which forms popular FEM

*One of many proofs to this theorem can be obtained considering u as the initial condition u_0 for multidimensional diffusion $\partial_t u = \nabla^2 u$. Since the time dependent solution is obtained by Gaussian convolution integral of initial function and is analytic, then as $t \to 0$, the initial function can be represented to arbitrary term (polynomial) in the Taylor series of the convolution integral. Please see Chapter 6 of [37] for more details and references. Also please see Theo. (2.7) for a new proof using orthonormal hull basis functions.

†Please see Theorem 10.1 of [37] for proof.

Lagrange basis, it is possible to show that the Lebesgue constant grows exponentially as $\Lambda_N(T) \sim 2^{N+1}/(e\,N\log N)$ [32]. Therefore, according to Eq. (2.7), for very high-order approximations, these points result in significant deviation from the optimal result. It is possible and practical to decrease Lebesgue constant by choosing Chebyshev points (with $\Lambda_N(T) < \frac{2}{\pi}\log(N+1)+1$). However, in higher dimensions, Chebyshev points are determined by Kronecker product of 1D distribution and hence Ω must be limited to quadrilateral and hexagonal shapes. A quad/hex tessellation of the domain significantly violates optimum partitioning of a complex shape domain and hence extra elements and hence extra DOF will be induced which is not desirable.

Yet there is still another elegant approach to decrease the Lebesgue constant by selecting a set of quadrature points as the interpolation points in Eq. (2.3) [34]. In this method, which is applicable to arbitrarily-shaped domains, the best linear approximation for the j^{th} moment

$$m_j = \int_\Omega f_j(x) d\mu, \tag{2.9}$$

can be obtained by solving the following underdetermined system

$$\sum_j w_j f_i(\hat{x}_j) = \mathcal{V}^T w = m_i, \quad 1 \le i \le N, \quad 1 \le j \le M. \tag{2.10}$$

or simply

$$\mathcal{V}^T w = m. \tag{2.11}$$

The moments in Eq. (2.9) can be obtained analytically for simple geometries or they can be computed using various methods including sub-triangulation and composite integration, the polygonal Gauss-like method [33] or an efficient method introduced below.

Introducing

$$F_k = \frac{1}{d}\prod_{i=1}^{d} \frac{{x_i}^{(d_i+\delta_{ik})}}{d_i+\delta_{ik}}, \tag{2.12}$$

satisfies $\nabla.F = \partial F_k/\partial x_k = f$ where variants f are given in (2.1) and (2.2). Therefore for the volume Ω enclosed by boundary $\partial\Omega$ we have

$$m = \int_\Omega f d\Omega = \int_{\partial\Omega} F_k \hat{n}_k d\partial\Omega \approx \sum_{\Delta\Omega_l} J_l \sum_m \sum_k F_k\left(\tilde{x}_{lm}\right) \hat{n}_{kl} W_{lm}, \tag{2.13}$$

where it reduces the computational complexity of moment calculations to one dimension smaller and can be easily computed in a computer program.

Iterative Process Reduced Using Singular Value Decomposition

An iterative procedure similar to the QR-based method by Sommariva, et al [34], can be written using Singular Value Decomposition of the Vandermonde matrix as follows.

Data: $\mathcal{V}_{k=0}$ is the Vandermode matrix defined in Eq.(2.4)
Result: Matrix P_s and well-conditioned Vandermonde matrix $\mathcal{V}_{s+1}$
for $k = 0 \ldots s$ **do**
  $\mathcal{V}_k = U_k S_k V_k^T$, **Note** : Economy size SVD ;
  $P_k = V_k S_k^{-1}$;
  $\mathcal{V}_{k+1} = \mathcal{V}_k P_k$ **Note** : $\mathcal{V}_{k+1} = U_k$ so is well-conditioned since U_k is unitary ;
end

Algorithm 1: The process of reducing the condition number of the Vandermonde matrix using only one iteration $s = 0$ which results in an explicit relation for the approximation of Fekete points in Eq. (2.16)

where the initially ill-conditioned $\mathcal{V}$ is forced to mimic the orthonormal matrix U_k by multiplication with $P_k = \text{inv}(S_k V_k^T)$. Note that the inverse is as costly as a matrix multiplication, since S_k is diagonal and V_k is orthonormal as well, hence P_k is readily known. In fact, this algorithm can be considered efficient if only one iteration is needed to reduce the condition number of $\mathcal{V}$. For a moment, let us consider the case $s = 0$

$$\mathcal{V}_1 = \mathcal{V}_0 P_0 = \mathcal{V} V S^{-1} \tag{2.14}$$

where $\mathcal{V}_0$ is the *unaltered* Vandermonde matrix in Eq.(2.11) and V and S are the right unitary matrix and the diagonal singular value matrix of the SVD of the *unaltered* Vandermonde matrix. The experience shows that the above system is well-conditioned and in contrast to QR-based algorithm where at least two iterations are necessary, Eq. (2.14) generates an explicit relation [14].

Left multiplying Eq. (2.11) with P^T yields

$$P^T \mathcal{V}^T w = P^T m = \mu \tag{2.15}$$

But from Eq. (2.14), $P^T \mathcal{V}^T = \mathcal{V}_1^T$ which is very well-conditioned since $\mathcal{V}_1 = U_1$ according to line 4 in Alg. (1) and hence its singular values are all close to unity. [‡] Therefore, Eq. (2.15) yields

$$\mathcal{V}_1^T w = \mu \tag{2.16}$$

which is a very tall underdetermined $N \times M, M \gg N$ system of equations. Equations (2.16) should be solved using a greedy algorithm that senses non-zero weights w of the quadrature points. The final result is $\mathcal{V}_1^T (w \neq 0) = \mu$ or

$$U_0^T w = \mu, \quad if \;\; w \neq 0, \tag{2.17}$$

where U_0 is the left unitary matrix of the SVD of the original Vandermonde matrix $\mathcal{V}_0$. Solving Eq. (2.17) yields a set of weights that are mostly positive and all nonzero. Such a solution algorithm can be obtained by performing a QR factorization of $\mathcal{V}_1$ and performing appropriate sorting to find the first N strong columns among M initial columns. The

[‡]We later use this property in Eq. (2.45) to show that the Lebesgue constant remains very small in this case and very accurate interpolation can be achieved. We will also show that such interpolation if used in the framework of DG spectral elements, leads to the superior accuracy compared to conventional FEM Lagrange basis functions of the same order of accuracy since the Lebesgue constant (and hence interpolation error) is smaller.

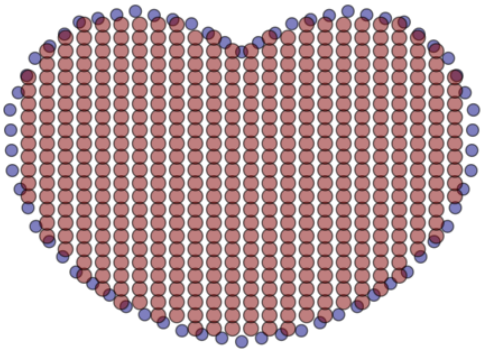


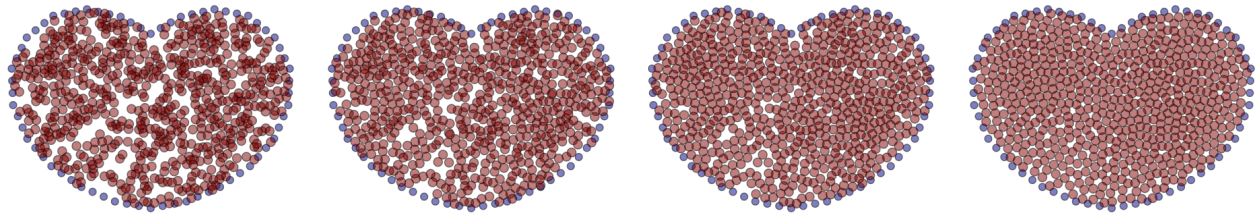
Figure 2.1 The generation of the candidate points using the fill pattern method

quadrature points $\hat{x}$ corresponding to these columns are then selected as the interpolation points. Since these are good estimate of Fekete points they are called Approximate Fekete points [34]. Another method used in this work to solve Eq. (2.16) is the method of Orthogonal Matching Pursuit (OMP) [19, 6], which has the advantage that no QR factorization is needed.

Generating candidate points and procedures to find approximate Fekete points

In order to find approximate Fekete points using Eq. (2.17), in the first step, the given arbitrary hull needs to be filled with a sufficiently *dense* set of candidate points. There are two methods in order to achieve this goal as described below.

1. *Fill pattern method* is a fast way to fill an arbitrary polyhedral subset of $\mathbb{R}^d$ with a large set of equally distance points inside the bounding box of the polyhedral. Then, each point is explicitly checked using polygonal point inclusion test [9] to see if it is inside the polyhedral and points that are outside are eliminated form the set. The result of this algorithm is very close to a uniform distribution except near the boundaries of the polyhedral where a gap is generated (See Figure 2.1).

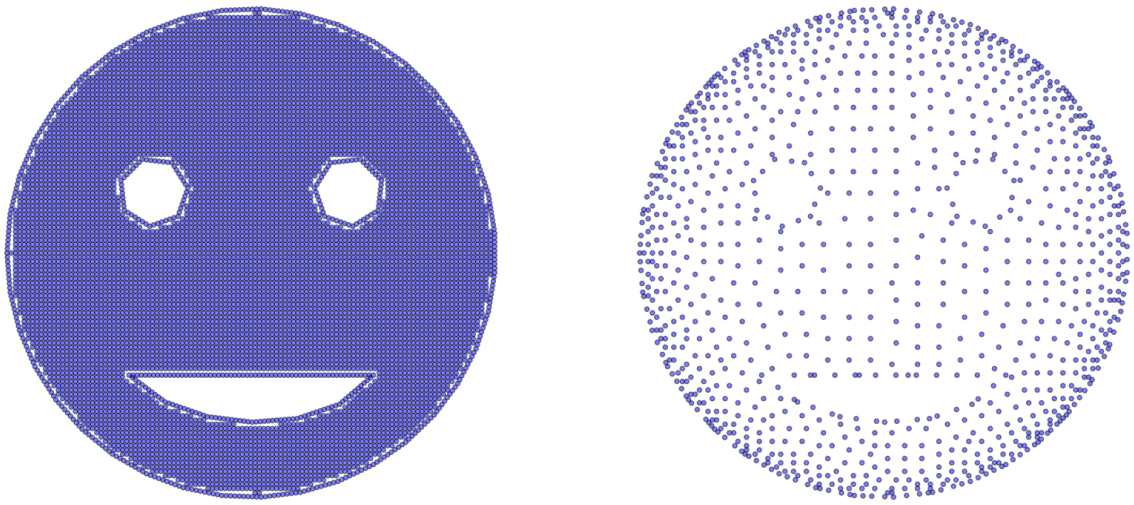


(a) $itr = 1$ (b) $itr = 2$ (c) $itr = 8$ (d) $itr = 32$

Figure 2.2 The generation of the candidate points using a pseudo time iterative gravitational method

2. The bounding box of the polyhedral is first filled with a random set of points. Then a gravitational equilibrium approach with an artificial time dependency algorithm similar to [26] is used to smoothen the distribution of these points. This algorithm is called the *iterative gravitational method* and can resolve the issues of the fill pattern method in generating uniform distribution near the boundaries. An example of this iterative procedure is illustrated in Figure 2.2.

Once the candidate points are generated, we are ready to perform SVD Alg. (1) and solve for Eq. (2.17) to find the approximate Fekete points. The result of this process is shown in Figure 2.3. As seen, in the first step, candidate points are generated using the fill pattern method in Figure 2.3-a. Subsequently, the approximate Fekete points are selected from this point set by solving Eq. (2.17). The selected points demonstrate stretching near boundaries which is a typical sign of Fekete-like points distribution. It should be noted that the selected points are not uniformly symmetric and the pattern has small deviations. However, these points generate very accurate interpolations as will be discussed in the following sections (see Figure 2.11).



(a) Candidate Points (b) Approximate Fekete Points

Figure 2.3 The procedure of generating candidate points (a) and selecting approximate Fekete points (b) on a concave hull

Numerical comparison between SVD and QR approaches

The QR algorithm of Sommariva, et al [34], needs at least two iterations to yield well-conditioned approximation of Fekete points. This is mentioned as the rule of "twice is enough", see Ref. [15]. However SVD based Alg. (1) only needs one iteration. In other words, the SVD algorithm results in a closed form relation Eq.(2.16) for approximation of the Fekete points. In order to validate this, two-dimensional function $u = \cos(3\pi x)\cos(3\pi y)$ is reconstructed on $\Omega = [-1,1] \times [-1,1]$ by evaluating the vandermonde matrix on two different set of points obtained by SVD and QR algorithms. In both cases, the function is reconstructed using nodal basis obtained by using Eq. (2.32). The number of iterations is fixed to $s = 1$ for both methods.

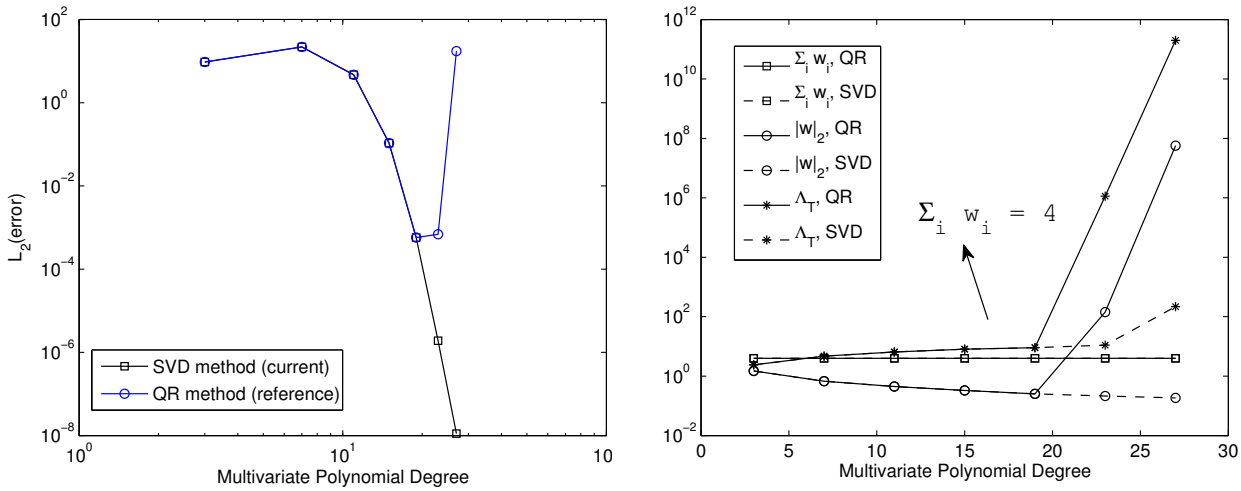


Figure 2.4 The comparison between SVD and QR approaches to find approximate Fekete points using only one iteration. Left) The interpolation error versus polynomial degree, Right) Various measures

As shown in Figure 2.4, with increasing the polynomial order, the Lebesgue constant of the QR-based algorithm increases rapidly and the reconstructed function exhibits unacceptable error. Interestingly, in contrast with QR-based method, the sum of the absolute values of the weights of the quadratures, i.e. w_i, obtained using SVD algorithm is monotonically decreasing. For both methods, $\sum_i w_i \approx 4$, which shows that the QR-based algorithm generates negative weights with increased polynomial degree.

Nodal Spectral Hull Basis Functions

A polynomial of degree at most N in d-dimensional space can be represented by

$$_j\,(x_1, x_2, \ldots, x_d) = \underbrace{\left[1, x_1, x_1^2, \ldots, x_2, x_2^2, \ldots, x_d, x_d^2, \ldots\right]_{(1,N)}}_{\text{either } f_P \text{ or} f_Q} \left[a\right]_{j(N,1)} = f a_j. \tag{2.18}$$

When evaluated at selected interpolation points $\hat{x}_i$ obtained using either the QR algorithm [34] or the SVD Alg. (1), for N-possible variation of constant vector a it yields

$$
\begin{aligned}
[\psi_1, \psi_2, \ldots, \psi_N]_{(1,N)} &= \left[[f(\hat{x}_i)]_{(1,N)}[a_1]_{(N,1)}, [f(\hat{x}_i)]_{(1,N)}[a_2]_{(N,1)}, \ldots, [f(\hat{x}_i)]_{(1,N)}[a_N]_{(N,1)} \right] \\
&= [f(\hat{x}_i)]_{(1,N)} \left[[a_1]_{(N,1)}, [a_2]_{(N,1)}, \ldots, [a_N]_{(N,1)} \right] = [f(\hat{x}_i)]_{(1,N)}[a]_{(N,N)} \qquad (2.19)
\end{aligned}
$$

Evaluating Eq. (2.19) at all $X_F = \hat{x}_i, i = 1 \ldots N$ yields the basis functions

$$
\Psi = [[\psi_1], [\psi_2], \ldots, [\psi_N]]_{(N,N)} = \begin{bmatrix} [f(\hat{x}_1)]_{(1,N)} \\ [f(\hat{x}_2)]_{(1,N)} \\ \vdots \\ [f(\hat{x}_N)]_{(1,N)} \end{bmatrix}_{(N,N)} [a]_{(N,N)}. \qquad (2.20)
$$

By using the definition of the Vandermonde matrix, Eq. (2.20) can be represented in the following compact form.

$$
\Psi = \mathcal{V}a, \qquad (2.21)
$$

where $\mathcal{V}$ is a $N \times N$ subsection of the tall $M \times N$ Vandermonde matrix defined in Eq.(2.4) such that $\mathcal{V}\left(\hat{x}_i \forall w_{i=1\ldots N} \neq 0\right)$ according to the solution of Eq.(2.16). In Eq. (2.21), the numerical value of the j^{th} basis function evaluated at point $\hat{x}_i$ is located at entry $\Psi(i,j)$. Applying Eq. (2.5) to Eq. (2.21) yields

$$
\mathcal{V}a = I, \qquad (2.22)
$$

and hence, the coefficients of the nodal basis functions can be determined by inverting the Vandermonde matrix. Here we introduce a new nodal/modal basis function by replacing the Vandermonde matrix in Eq. (2.22) with a *complete* singular value decomposition

$$USV^T a = I, \tag{2.23}$$

which yields the unknown coefficients of the *nodal* Approximate Fekete Basis (AFB) as

$$a = V\ S^{-1}\ U^T. \tag{2.24}$$

The coefficient matrix a can be calculated *once* and tabulated for arbitrary order of approximation and variety of hull shapes. Then, any point $x \in \Omega$, the value of the j^{th} basis function can be efficiently calculated with $\mathcal{O}(N)$ operations by evaluating f_P or f_Q at that point and performing the vector product in Eq.(2.18). In order to obtain the modal form of AFB, let us substitute Eq. (2.24) into Eq. (2.19) to obtain

$$\psi_j = [\psi_1, \psi_2, \ldots, \psi_N]_{(1,N)} = [f(x)]_{(1,N)} V\ S^{-1}\ U^T \tag{2.25}$$

where $x \in \Omega$ is not necessarily an interpolation point. An alternative evaluation of the nodal basis functions is to start with Eq. (2.22) and then define $\hat{a}$ such that

$$P_0 \hat{a} = a \tag{2.26}$$

where P_0 is defined in the Alg. (1) for $P = P_{k=0}$. Substituting Eq.(2.26) in Eq. (2.22) yields

$$\mathcal{V} P_0 \hat{a} = \mathcal{V}_0 P_0 \hat{a} = I, \tag{2.27}$$

Using Eq.(2.14), Eq. (2.27) can be written as

$$\mathcal{V}_1 \hat{a} = I, \tag{2.28}$$

which is already calculated and has better condition number compared to $\mathcal{V}\,(\hat{x}_i \forall w_{i=1\ldots N} \neq 0)$ because it is forced to mimic the unitary matrix U according to line (4) in Alg. (1). Now, replacing $\mathcal{V}_1$ with its SVD yields

$$U_1 S_1 V_1^T \hat{a} = I, \tag{2.29}$$

or

$$\hat{a} = V_1 S_1^{-1} U_1^T. \tag{2.30}$$

Substituting Eq. (2.30) in Eq. (2.26) yields the coefficients of the nodal basis functions as follows.

$$a = P_0 V_1 S_1^{-1} U_1^T \tag{2.31}$$

and hence the nodal basis functions can also be represented by inserting the coefficients a from Eq. (2.31) into Eq.(2.18). The final result is given below

$$[\psi_1(x), \psi_2(x), \ldots, \psi_N(x)] = [f(x)]_{(1,N)}\; P_0 V_1 S_1^{-1} U_1^T \tag{2.32}$$

Modal Spectral Hull Basis Functions

Right multiplying Eq. (2.25) with the unitary matrix U yields

$$\bar{\psi_j} = \psi_j U = [f(x)]_{(1,N)} V\; S^{-1}, \tag{2.33}$$

where according to Theo. (2.3), $\bar{\psi_j}$ is the j^{th} modal basis function since $\bar{\psi_j}$ are orthogonal. Since the singular values are naturally ordered from *large values* to *small values* in the

standard SVD, the $\bar{\psi}_j$ modes also start from lowest frequency mode for $j = 1$ to the highest frequency mode for $j = N$. The important point is that a given function u can be *approximated* as the sum of $\bar{\psi}$ with the converging property in a sense that the higher modes included, the more accuracy is obtained. To show this, let us project u into a subset of the modal space

$$u = \sum_{k=1}^{k_m} \bar{\psi}_k w_k, 1 \leq (k_m = k_{max}) \leq N. \tag{2.34}$$

Substituting $\bar{\psi}_j$ from Eq. (2.33) in Eq. (2.34) yields

$$u = [f(x)]_{(1,N)} V_{(N,k_m)} \; S^{-1}_{(k_m,k_m)} w_{(k_m,1)} \tag{2.35}$$

Definition 2. *Inspired by the terminology "Fourier Coefficient" given in (Ref. [10]- Page 27) for the polynomial approximation of one-dimensional functions, we consistently extend it to the Generalized Fourier Coefficient/Amplitude (GFC) of $\mathbb{R}^d$ approximation $u \approxeq \bar{\psi}_k w_k$ where w_k is the GFC.*

Now we have the following theorem:

Theorem 2.1. *Let $u(x)$ be defined for $x \in \Omega \subset \mathbb{R}^d$. Then the GFC w_k in the series expansion $u = \sum_{k=1}^{k_m} \bar{\psi}_k w_k, 1 \leq (k_m = k_{max}) \leq N$ decays.*[§]

In practice, the values of GFCs can be computed in a surprisingly efficient manner by utilizing the unitary matrix U. This is discussed in the following theorem.

Theorem 2.2. *For any bounded u defined on $\Omega \subset \mathbb{R}^d$, the GFCs of orthogonal hull expansion Eq. (2.34) are given by*[¶]

$$w = U^T u \tag{2.36}$$

[§]Proof of Theorem 2.1 is described in detail in Theorem 4.1 in [13].
[¶]Proof of Theorem 2.2 is described in detail in Theorem 4.2 in [13].

Remark 1. *The reader may have noticed that Eq. (2.36) constitutes a Generalized Discrete Transform similar to Discrete Fourier Transform by multiplying the given function u with the unitary matrix U. While the basic mechanism of $w = U^T u$ is similar to DFT, Eq. (2.36) generalizes DFT to any arbitrarily shaped and non-periodic hulls (domains). This important result reveals that the unitary matrix U is in fact can be regarded as a very generalized convolution operator (matrix).*

Remark 2. *The truncated GFCs using $w_{1\ldots k_m} = U^T_{(N,1\ldots k_m)} u$ yields a generalized a posteriori error estimator. In the classical Fourier analysis, the tail of the Fourier series (higher frequencies) can be eliminated to smoothen the solution. This filtering strategy can be done here on arbitrary shaped hull by just using the first modes in the series expansion $u = \bar{\psi}_k w_k$ with the coefficients $w_{1\ldots k_m} = U^T_{(N,1\ldots k_m)} u$. Strictly speaking, the norm of the eliminated tail, i.e. $\left\| U^T_{(N,k_{m+1}\ldots N)} u \right\|_2$ is an error estimator of the sum of the eliminated energy according to Parseval theorem (2.4).*

Calculating the Lebesgue constant

The general linear interpolation $I(u) = \mathcal{L}_N u$ can be written using the modal expansion Eq. (2.34) as follows

$$I(u) = \mathcal{L}_N u = \sum_{k=1}^{k_m} \bar{\psi}_k w_k. \tag{2.37}$$

It is our interest to study the error generated by such interpolation. In particular, the conventional concept used in the Approximation Theory community is to show that

$$E\left(I(u)\right) = \left\| u - \mathcal{L}_N u \right\|, \tag{2.38}$$

is small enough. Equation (2.38) can be written as

$$\|u - \mathcal{L}_N u\| = \|u - u^* + u^* - \mathcal{L}_N u\| \tag{2.39}$$

where $u^* = \mathcal{L}_N u^*$ is the optimal interpolant at point $x \in \Omega$. In fact, $\mathcal{L}_N$ is Lagrange basis constructed at the approximate Fekete points $\hat{x} = x \in \Omega$ and then used as basis at any point $x \in \Omega$ in particular $x = \hat{x}$ where $\mathcal{L}_N = 1$ and hence $u^* = \mathcal{L}_N u^*$. In the next step, by using the triangle inequality

$$\begin{aligned} E\left(I(u)\right) = \left\| u - u^* + \underbrace{u^*}_{\mathcal{L}_N u^*} - \mathcal{L}_N u \right\| &\leq \|u - u^*\| + \|\mathcal{L}_N u^* - \mathcal{L}_N u\| \leq \|u - u^*\| + \|\mathcal{L}_N\| \|u - u^*\| \\ &\leq \left(1 + \|\mathcal{L}_N\|\right) \|u - u^*\| \end{aligned} \tag{2.40}$$

Therefore the interpolation error is bounded above by the norm of the interpolation operator and hence the Lebesgue constant can be defined as

$$\Lambda_N(T) = \|\mathcal{L}_N\| \tag{2.41}$$

Therefore, we are interested in the interpolation points that result in the minimal Lebesgue constant to maximize the accuracy of the interpolation according to Eq. (2.40). The approximate Fekete points presented before have such an impressive property. Exact calculation of the Lebesgue constant has always been a hard task and mostly done using empirical relations. The closed-form formulae are only available for the case of 1D and simple interpolants like Chebyshev polynomials as mentioned before. We derive an explicit relation for the Lebesgue constant of an arbitrary shaped hull in $\mathbb{R}^d$. The key is hidden in the minimum singular value of the orthogonal basis functions $\bar{\psi}_i$ as it is shown below.

Using Eq. (2.33), Eq. (2.41) leads to

$$\Lambda_N(T) = \|\mathcal{L}_N\| = \left\| [f(x)]_{(1,N)} V \; S^{-1} \right\|, \tag{2.42}$$

or

$$\|\mathcal{L}_N\| = \sqrt{\int_\Omega \left([f(x)]_{(1,N)} V \; S^{-1} \right)^T [f(x)]_{(1,N)} V \; S^{-1} d\Omega}. \tag{2.43}$$

or

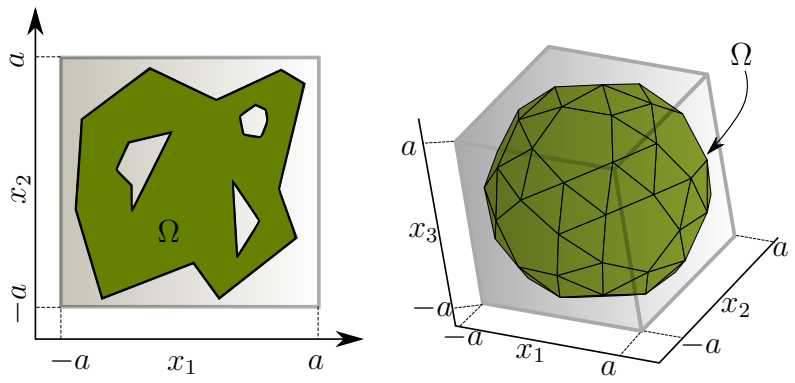
$$\|\mathcal{L}_N\| = \sqrt{\int_\Omega S^{-1} V^T \; (f^T f) \, V \; S^{-1} d\Omega} = \sqrt{\int_\Omega (f^T f) \, d\Omega \; S^{-1} V^T V S^{-1}} = \sqrt{\int_\Omega (f^T f) \, d\Omega} \; S^{-1}. \tag{2.44}$$

Therefore

$$\|\mathcal{L}_N\|_{\max} = \frac{\sqrt{\int_\Omega (f^T f) \, d\Omega}}{\sigma_{\min}} \tag{2.45}$$

Remark 3. *It is crucially important to note that the norm of the moments, i.e.* $\int_\Omega \left(f^T f \right) d\Omega$ *in the numerator of Eq. (2.45) can be significantly large (especially for the higher order polynomials) on a domain that is not centered around the origin. This will lead to significantly inaccurate interpolation results because the Lebesgue constant increases rapidly according to Eq. (2.40).*

Therefore it is important to use hulls centered around the origin (and additionally mapping them inside the unity box- see Figure 2.5), where it can be shown that the result of $\int_\Omega \left(f^T f \right) d\Omega$ is always smaller than unity and monotonically converges to unity for an infinite order polynomial (See Theo. (2.5)). In this case, the upperbound in Eq. (2.45) is very small and for the spectral hull basis evaluated on approximate Fekete points, since $\sigma_{\min}$ is close to unity, the interpolation remains very accurate for significantly higher degree polynomials. In practice, in the implementation of discontinuous finite element solution of compressible Euler equations, the authors have implemented hulls directly in the physical space without mapping. The results showed a gradual *leaking* in the x-momentum and eventually the



(a) Two Dimensional (b) Three Dimensional

Figure 2.5 The transformation of Ω to a bounded area near the origin

code blew up. There was absolutely no way for the authors to find the bug except Eq. (2.45) miraculously shed light on the source of the bug. After the authors mapped each hull to the center, the bug was completely resolved and very stable and accurate results were obtained. This side story demonstrates the significance of the presented analysis in the practical implementations.

The approximation theory of the spectral hull expansion

This section summarizes the important theorems which are necessary to prove the convergence of the spectral hull expansion. First, we start by proving the orthogonality and Parseval theorems and then the Weierstrass theorem is proven for a particular form of spectral hull expansion which utilizes orthonormal hull basis.

Theorem 2.3. *$\bar{\psi}_j$ forms an orthogonal set of basis functions for $i = 1 \dots N$.*[‖]

Remark 4. *As a result, the basis*

$$\tilde{\psi}_k = \frac{f}{\|f\|_2} V_{(k)}, \tag{2.46}$$

[‖]Proof of Theorem 2.3 is described in detail in Theorem 4.3 in [13].

is orthonormal. However, this is only used for the proof of theorems and is not used in the spectral element formulation in this paper since it requires the extra work for computation of the norm of the moment vector over the hull. Although the latter can be precomputed and tabulated.

Theorem 2.4. *Parseval theorem for the orthogonal hull expansion given in Eq. (2.34)*[**]

$$\int_{\Omega} \left(\sum_{k=1}^{m} \bar{\psi}_k w_k \right)^2 d\Omega = \|f\|_2^2 \sum_{k=1}^{m} \left(\frac{w_k}{\sigma_k} \right)^2. \tag{2.47}$$

Theorem 2.5. *For a* Ω *selected as an arbitrary subset of* $x_1 \times x_2 \times \ldots x_i \times \ldots \times x_d = \mathbb{R}^d$ *inside the d-dimensional cube* $|x_i| \leq a = \tanh(\frac{1}{2}) = 0.4621\ldots$[††] *(See Figure 2.5),* $\int_{\Omega} \|f\|_2^2 d\Omega$ *is monotonically increasing but convergent and*

$$\int_{\Omega} \|f\|_2^2 d\Omega \leq \int_{\Omega} \frac{1}{\prod_{l=1}^{d} (1 - \xi_k^2)} d\xi_1 d\xi_2 \ldots d\xi_d \leq 1, \qquad \lim_{N \to \infty} \int_{\Omega} \|f_Q\|_2^2 d\Omega = 1. \tag{2.48}$$

Note that the l^{th} GFC in the expansion $u = \sum_{k=1}^{m} \bar{\psi}_k w_k$ can be obtained by multiplying both sides with $\bar{\psi}_l$ and integrating over Ω. The result is presented below

$$\int_{\Omega} u \bar{\psi}_l d\Omega = \sum_{k=1}^{m} w_k \left(\int_{\Omega} \bar{\psi}_l \bar{\psi}_k \right). \tag{2.49}$$

Applying the orthogonality Theo. (2.3) to Eq. (2.49) yields

$$\int_{\Omega} u \bar{\psi}_l d\Omega = \sum_{k=1}^{m} w_k \frac{\|f\|_2^2 \delta_{lk}}{\sigma_l \sigma_k} = \|f\|_2^2 \frac{w_l}{\sigma_l^2}. \tag{2.50}$$

Hence

[**]Proof of Theorem 2.4 is described in detail in Theorem 4.4 in [13].
[††]Proof of Theorem 2.5 is described in detail in Theorem 4.5 in [13].

$$w_l = \frac{\sigma_l^2}{\|f\|_2^2} \int_\Omega u \bar{\psi}_l d\Omega, \tag{2.51}$$

which yields

$$\frac{w_k}{\sigma_k} = \frac{\sigma_k}{\|f\|_2^2} \int_\Omega u \bar{\psi}_k d\Omega. \tag{2.52}$$

Equation (2.52) gives us a new upper bound for $\frac{w_k}{\sigma_k}$ as follows

$$\frac{|w_k|}{\sigma_k} = \frac{\sigma_k}{\|f\|_2^2} \left\| \int_\Omega u \bar{\psi}_k d\Omega \right\| \leq \frac{\sigma_k}{\|f\|_2^2} \sqrt{\int_\Omega u^2 d\Omega} \sqrt{\int_\Omega \bar{\psi}_k^2 d\Omega}. \tag{2.53}$$

Since u is square integrable, $\|u\|_2 = \sqrt{\int_\Omega u^2 d\Omega}$ is finite. Substituting the definition $\bar{\psi}_k = f V_{(k)}/\sigma_k$ in Eq. (2.53) yields

$$\frac{|w_k|}{\sigma_k} \leq \frac{\sigma_k}{\|f\|_2^2} \|u\|_2 \sqrt{\|f \frac{V_{(k)}}{\sigma_k}\|} \leq \frac{\sigma_k}{\|f\|_2^2} \|u\|_2 \|f\|_2 \frac{1}{\sigma_k}, \tag{2.54}$$

or

$$\frac{|w_k|}{\sigma_k} \leq \frac{\|u\|_2}{\|f\|_2}. \tag{2.55}$$

It can be concluded that the k^{th} GFC is always bounded by

$$|w_k| \leq \sigma_k \max\left(\frac{\|u\|_2}{\|f\|_2}, \|\breve{u}\| \right) \tag{2.56}$$

This means that the magnitude of w_k is not necessarily monotonically decreasing, although its bound, i.e. $|w_k|_{max} = \sigma_k \times \max\left(\frac{\|u\|_2}{\|f\|_2}, \|\breve{u}\|\right)$ is always monotonically decreasing. This situation is graphically illustrated in Figure 2.6.

Theorem 2.6. *Parseval theorem for the orthonormal expansion given in Eq. (2.46)*[*]

[*]Proof of Theorem 2.6 is described in detail in Theorem 4.6 in [13].

$$\int_{\Omega}\left(\sum_{k=1}^{m}\tilde{\psi}_k\tilde{w}_k\right)^2 d\Omega=\sum_{k=1}^{m}\tilde{w}_k^2. \tag{2.57}$$

Theorem 2.7 (Weierstrass Approximation Theorem for Ω arbitrary subset of $\mathbb{R}^d$)**.** *Assume that u is a bounded real-valued function on $\Omega\subset\mathbb{R}^d$ then for every $\epsilon>0$, there exists a polynomial p such that for all $x\in\Omega$, $\|u-p\|<\epsilon$. Particularly, we also give an explicit relation for the polynomial p below.*[†]

$$p=\left(\int_{\Omega}\tilde{\psi}_{\tilde{k}}ud\Omega\right)\tilde{\psi}_{\tilde{k}},\quad \tilde{k}=\text{permutation}\left(\text{sort}_{\downarrow}\left[\int_{\Omega}f_1ud\Omega,\int_{\Omega}f_2ud\Omega,\ldots,\int_{\Omega}f_Nud\Omega\right]\left[V_{(1)},V_{(2)},\ldots,V_{(N)}\right]\right) \tag{2.58}$$

The monotonic decay of $\tilde{w}_{\tilde{k}}$ is compared to the decay of the coefficients of the orthogonal hull basis $\bar{\psi}_k$ which decay in an envelope (bound). Please see Figure 2.6 for such comparison.

Numerical results of spectral hull basis functions

Before assessing the accuracy of interpolation via approximate Fekete points on general hulls, one needs to *validate* the modal basis proposed in Eq. (2.33) on a set of Chebyshev points. Therefore, in the fist test case, a set of 20x20 Chebyshev points are generated by the Kronecker product of one-dimensional distribution. Then, the Vandermonde matrix is evaluated on these points and the modal basis Eq. (2.33) are obtained. The results are plotted in Figure 2.7. As seen, the first and the second modes have low frequency contents, while the last mode has the highest frequency. Also, symmetry is well preserved on Chebyshev points.

It is important to see the effect of eliminating the higher modes. Figure 2.8 discusses this in more details. As seen, the rank of orthogonal hull basis is 400 corresponding to a full

[†]Proof of Theorem 2.7 is described in detail in Theorem 4.7 in [13].

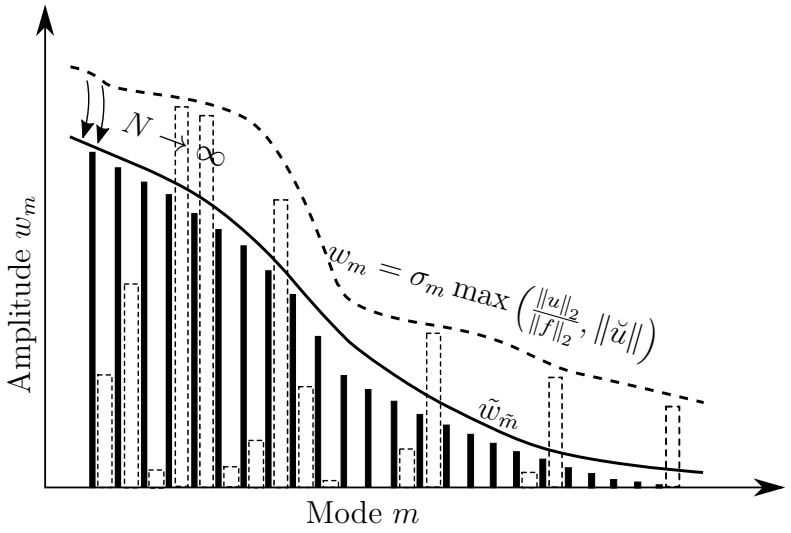


Figure 2.6 The decay of Generalized Fourier Coefficients (GFCs); dashed-bars correspond to orthogonal basis $\bar{\Psi}$. solid-bars correspond to orthonormal basis $\tilde{\Psi}$ which are monotonically decreasing according to Theo. (2.7)

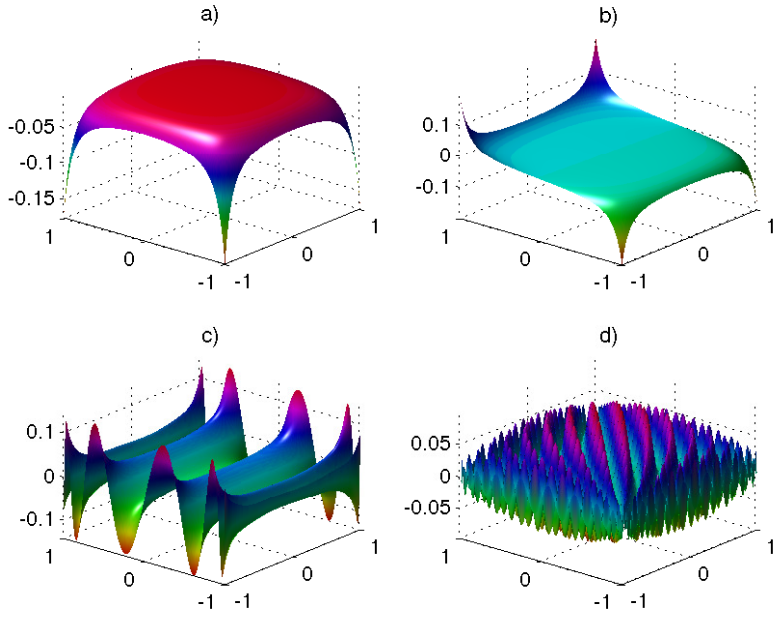


Figure 2.7 The *orthogonal* spectral hull basis Eq. (2.33) evaluated on Chebyshev points. a) $\bar{\psi}_1$, b) $\bar{\psi}_2$, c) $\bar{\psi}_{60}$, d) last mode, i.e. $\bar{\psi}_{400}$

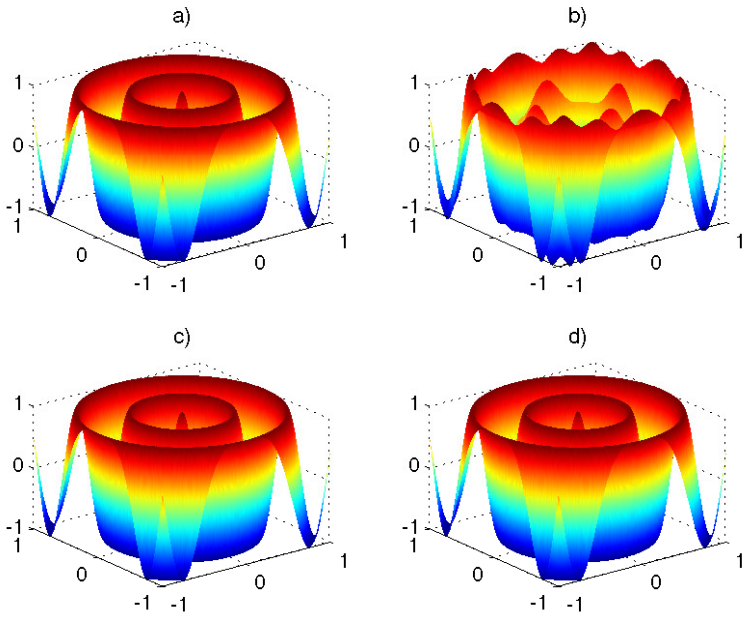


Figure 2.8 The spectral filtering of $u = \cos\left(4\pi\sqrt{x^2+y^2}\right)$ in Q space by eliminating higher frequency orthogonal hull basis Eq. (2.34). a) $k_m = 400$, i. e. *full rank*, b) $k_m = 200$, i. e. *half rank*, c) $k_m = 360$, i. e. *ninety percent rank*, d) Exact

SVD decomposition of the Vandermonde matrix evaluated at Chebyshev points as mentioned before. When all basis are included, i.e. $k_m = 400$, the reconstructed function has excellent agreement with the exact solution. Additionally, a ninety-percent reconstruction yields ideal reconstruction. It can be seen in Figure 2.8-b that the reconstruction of the function without considering higher frequncies yields an unacceptable result. These observations are in agreement with the proposed convergence theory.

After validating the orthogonal modal hull basis functions, one need to assess Alg. (1). A plot of *nodal* basis functions (See Eq. (2.25)) for a 16^{th}-order P space on a T-hull is presented in Figure 2.9-Top where the first basis ψ_1 is plotted in top-left and the 200^{th} basis is plotted in the top-right. Note that ψ_1 is one at the top corner of the T shaped hull and zero at other Fekete points. Modal basis functions (see Eq. (2.33)) for a 16^{th}-order P space

on a T-shaped hull is presented in Figure 2.9-Bottom where the first mode $\bar{\psi}_1$ is plotted in bott-left and the 200^{th} mode is plotted in the bott-right.

The orthogonal spectral hull basis functions in addition to being well-conditioned, are very accurate compared to the Radial Basis Functions (RBF) for the same degrees of freedom. Figure 2.10 compares the reconstruction of two wavelengths of sinusoidal functions using a RBF basis (left column) and orthogonal spectral hull basis functions $\bar{\Psi}$ (right column). As clearly shown, the RBF is either very inaccurate (Top-Left) or requires many DOF to yield accurate results (Bottom-Left). However, $\bar{\Psi}$ yields spectral resolution of minimum points per wavelength (see the middle of smiley-face hull). This resolution is comparable to the Fourier decomposition of smooth functions on simple rectangular geometry. These results, which demonstrate the superior accuracy/efficiency of the spectral hull basis functions, can contribute to the field of meshfree methods where RBF are extensively used.

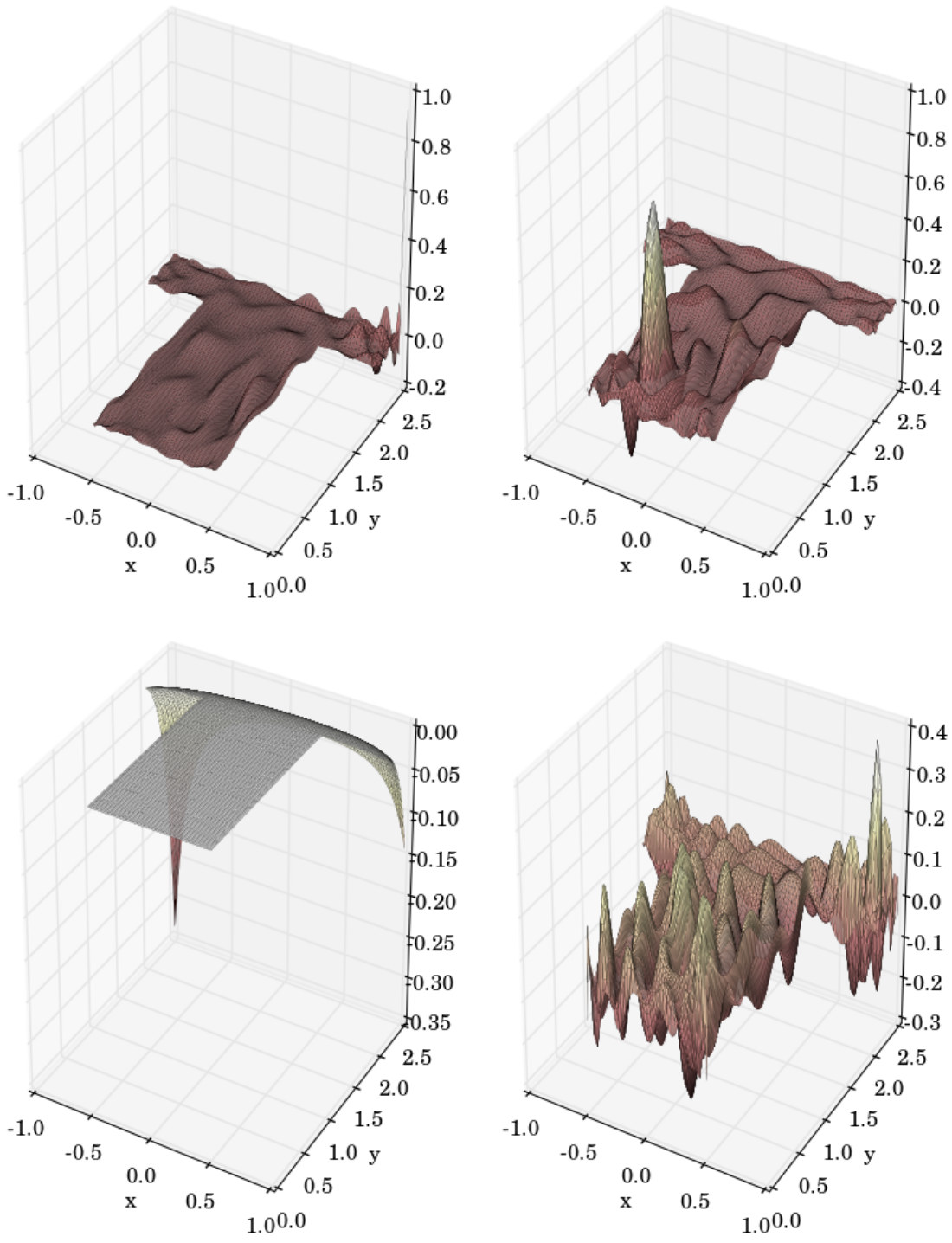


Figure 2.9 The first and the 200th shape functions of *nodal* (top) and *modal* (bottom) Approx. Fekete basis. Top Left) ψ_1, Top Right) ψ_{200}, Bott. Left) $\bar{\psi}_1$, Bott. Right) $\bar{\psi}_{200}$

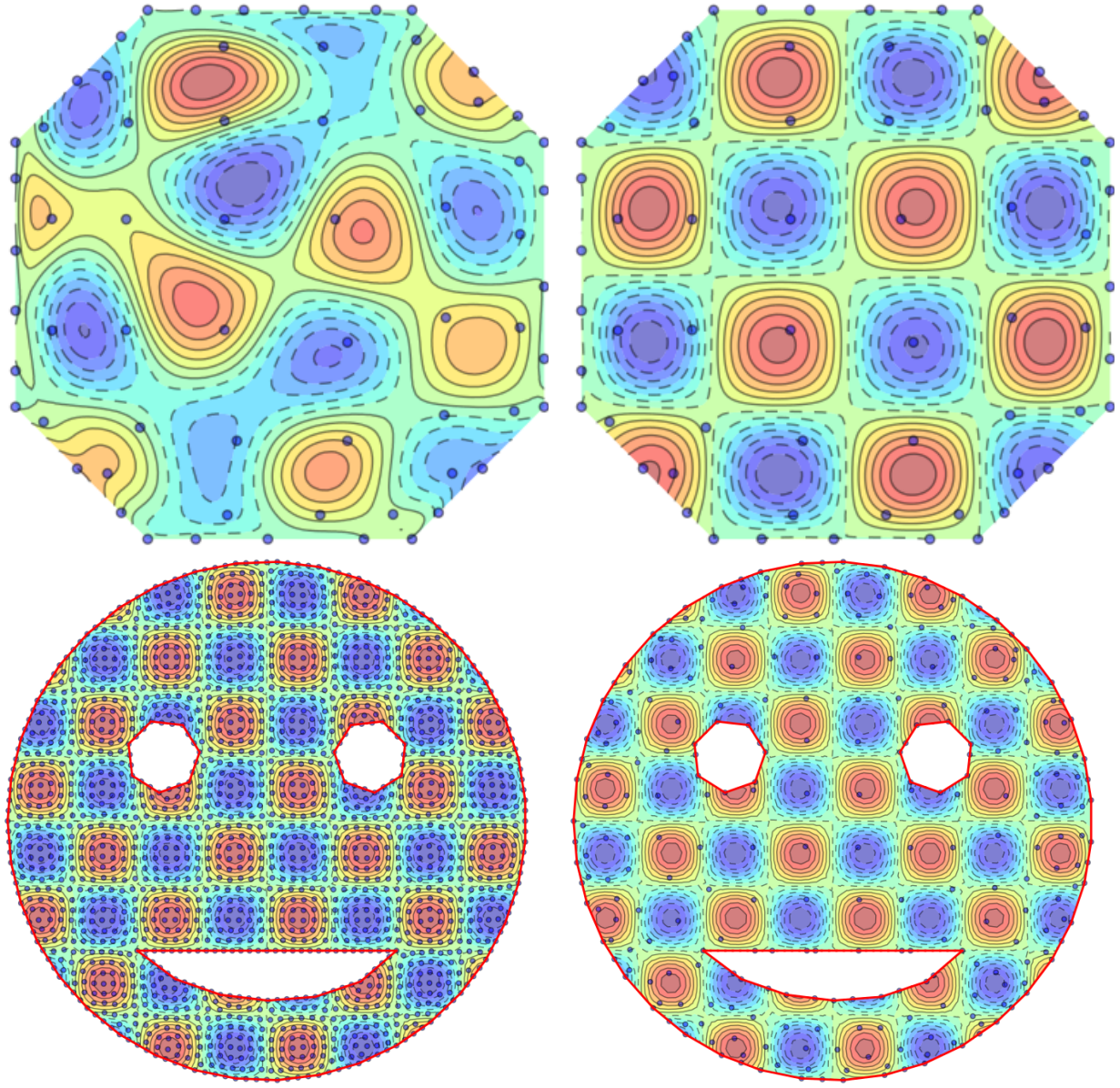


Figure 2.10 Comparison of spectral hull basis functions and RBF in the reconstruction the solution on a hull. Top row) with same DOF, Bottom row) with same L_2 error, Left column is RBF, right col. corresponds to orthogonal spectral hull basis functions $\bar{\Psi}$

To further assess the resolution of spectral hull basis functions, the polynomial order on the smiley hull is increased. The result which is presented in Figure 2.11 demonstrates

that six wavelengths can be reconstructed on this highly concave hull while spectral accuracy is preserved.

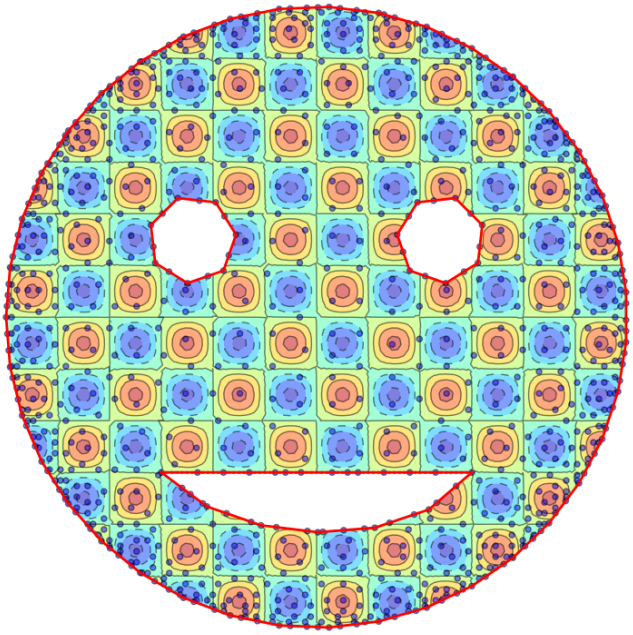


Figure 2.11 Sub-elemental resolution of six wavelengths of $u = \sin(2\pi x)\sin(2\pi y)$ on a highly concave hull. As shown in the middle, the spectral accuracy corresponding to the optimum resolution is demonstrated

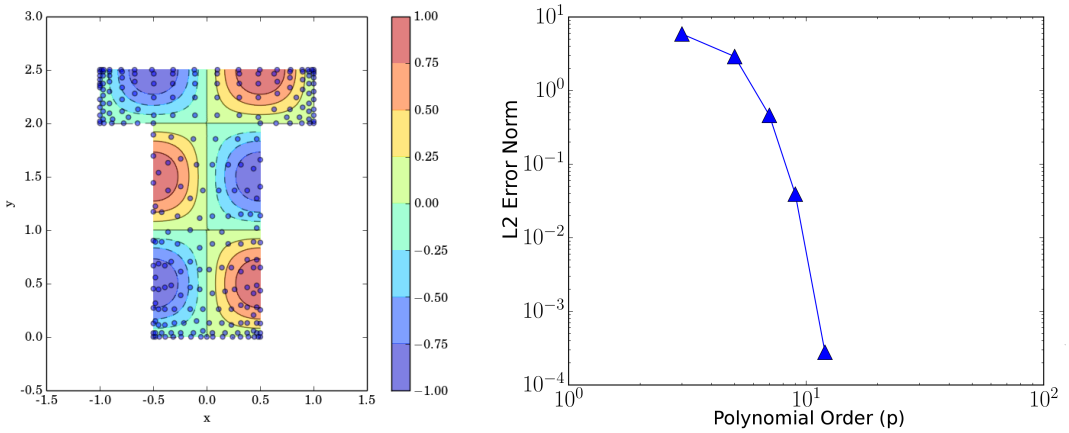


Figure 2.12 Superlinear convergence of spectral hull basis on the concave T-hull. Left) Reconstructed function using 16^{th}-order spectral hull basis constructed on approximate Fekete points. Right) L_2 error compared to analytical reconstruction

To demonstrate the super linear convergence of spectral hull basis, the concave T-shaped hull is used to reconstruct the sinusoidal function. The result is presented in Figure 2.12 where the super linear (spectral) convergence is shown.

Using the concept of GFC (see Definition (2)), it can be shown that the spectral content increases as the number of sides and/or the non-convexity of the hull increases. This can be a drawback of the generality of selecting *arbitrarily* shaped hulls. This is shown in Figure 2.13 where the GFCs of the modal expansion is plotted against the corresponding modes for different hull shapes, in which in all cases, the same function is reconstructed. As shown, the more complicated the hull gets, the more modes are required to reconstruct the function [13].

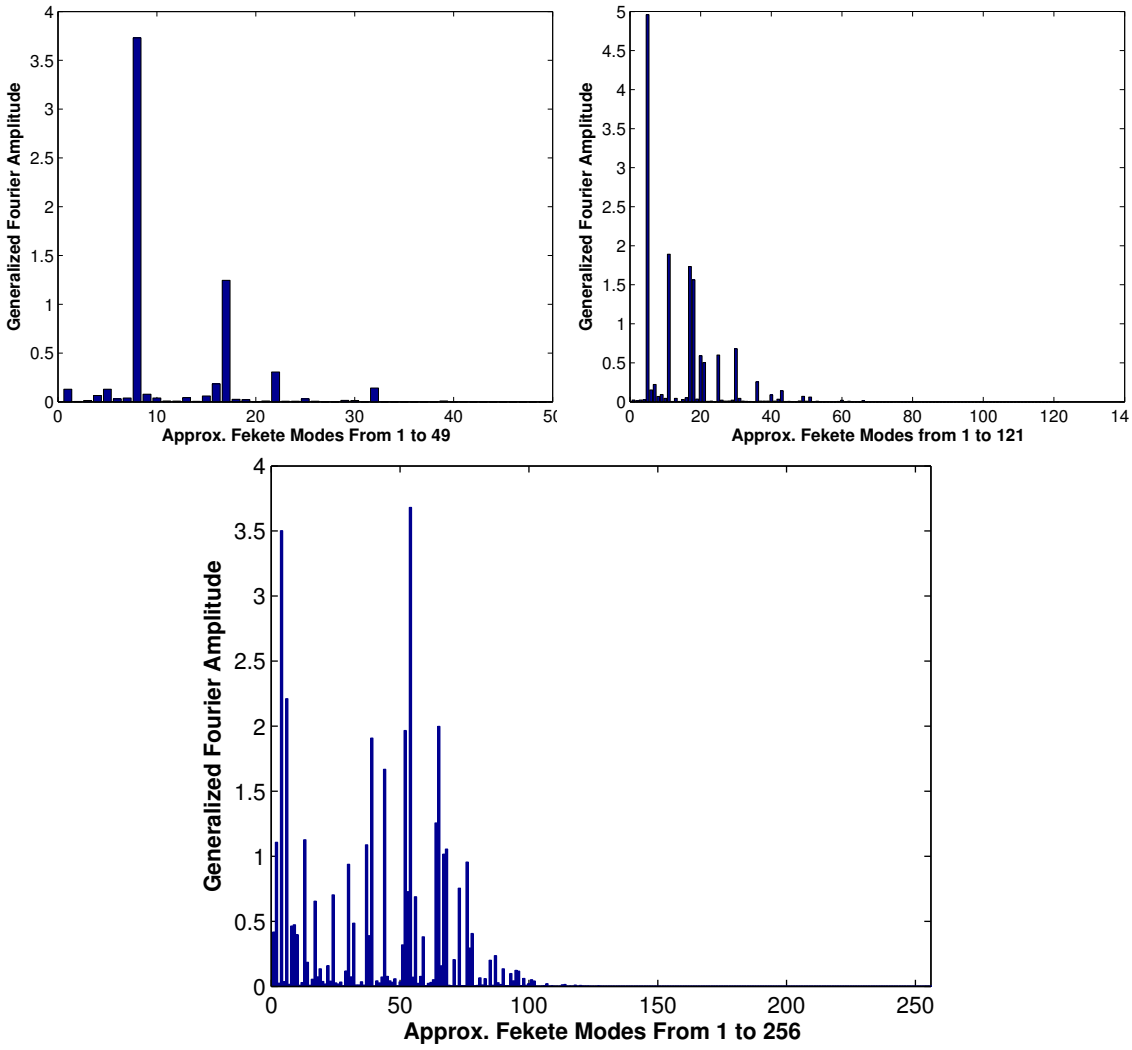


Figure 2.13 The spectra of $u = \sin(\pi x)\sin(\pi y)$ on different convex/concave elements; Left) using 7^{th} order orthogonal spectral hull basis $\bar{\Psi}$ on a quadrilateral element. Right) using 11^{th} $\bar{\Psi}$ on a hexagonal element. Bottom) using 16^{th} $\bar{\Psi}$ on a T-shape element

Application of Fekete Spectral Element methods to two-dimensional elasticity

The equations of elasticity can be obtained by considering Newton's second law of motion in the equilibrium case, which means that the summation of total linear and angular momentums in the system should be zero. For a control volume containing a solid, the momentum equation reads

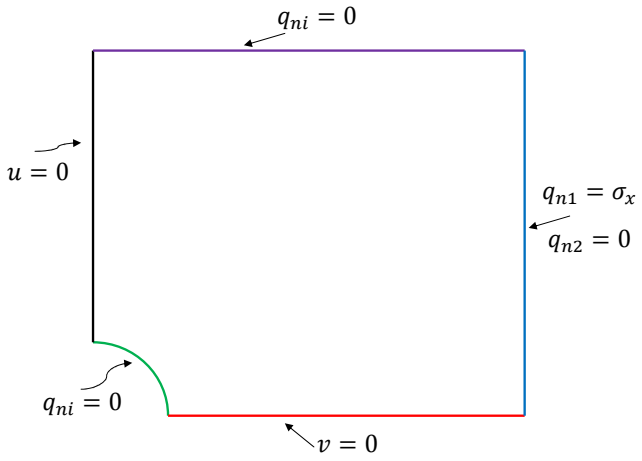


Figure 2.14 The selected part of the domain and appropriate boundary conditions

$$\int_S \sigma_{ij}.dS_j + \int_V f_i \, dV = 0. \tag{2.59}$$

where f_i is body force (like magnetic force or inertial forces like centrifugal forces etc.) and σ_{ij} is the shear stress tensor. Using Divergence theorem, we obtain

$$\int_V \partial_j \sigma_{ij} \, dV + \int_V f_i \, dV = 0. \tag{2.60}$$

which is true at each point inside an infinitesimal volume, i.e.

$$\partial_j \sigma_{ij} + f_i = 0, \tag{2.61}$$

This is simply the balance between the stress forces and body forces for a differential element. The objective of the current work is to solve eq.(2.61) using Galerkin method. Multiplying eq.(2.61) by test function w and integrating over the an spatial element yields

$$\int_\Omega -w \, \partial_j \sigma_{ij} d\Omega = \int_\Omega w \, f_i d\Omega \tag{2.62}$$

Using integration by parts, we obtain

$$-\int_{\partial\Omega} w\,\sigma_{ij}\,\hat{n}_j + \int_{\Omega} \partial_j w\,\sigma_{ij} d\Omega = \int_{\Omega} w\,f_i d\Omega \tag{2.63}$$

or

$$\int_{\Omega} \partial_j w\,\sigma_{ij} d\Omega = \int_{\partial\Omega} w\,q_{n_i} + \int_{\Omega} w\,f_i d\Omega \tag{2.64}$$

where $q_{n_i} = \sigma_{ij}\,\hat{n}_j$ is definitely a secondary variable which is used to impose natural boundary conditions as shown in Figure 2.14.

Weak Formulation

We use constitutive equations to relate the stress to strain and displacement. Assuming that the displacement field is relatively small and that the material remains in the elasticity region, it is good to use the following linearization as our constitutive(ad-hoc) relations

$$\sigma_i = C_{ij}\epsilon_j, \qquad \epsilon_j = T_{jk}u_k \tag{2.65}$$

where $\sigma_i = [\sigma_x, \sigma_y, \sigma_{xy}]^T$ is the rearranged stress vector for two dimensional case and

$$C_{ij} = \begin{bmatrix} C_{11} & C_{12} & 0 \\ C_{21} & C_{22} & 0 \\ 0 & 0 & C_{66} \end{bmatrix} \tag{2.66}$$

and

$$T_{jk} = \begin{bmatrix} \partial_x & 0 \\ 0 & \partial_y \\ \partial_y & \partial_x \end{bmatrix}. \tag{2.67}$$

Table 2.1 Contracted versus Directional Notation for Stresses and Strains

Stresses		Strains	
Condensed Notation	Directional Notation	Condensed Notation	Directional Notation
σ_1	σ_x	ϵ_1	$\epsilon_x = \frac{\partial_u}{\partial_x}$
σ_2	σ_y	ϵ_2	$\epsilon_y = \frac{\partial_v}{\partial_y}$
σ_3	σ_z	ϵ_3	$\epsilon_z = \frac{\partial_w}{\partial_z}$
σ_4	σ_{yz}	ϵ_4	$\epsilon_{yz} = \frac{\partial_v}{\partial_z} + \frac{\partial_w}{\partial_y}$
σ_5	σ_{zx}	ϵ_5	$\epsilon_{zx} = \frac{\partial_w}{\partial_x} + \frac{\partial_u}{\partial_z}$
σ_6	σ_{xy}	ϵ_6	$\epsilon_{xy} = \frac{\partial_u}{\partial_y} + \frac{\partial_v}{\partial_x}$

The definitions for the stiffness matrix constants C_{ij} can be derived from the generalized Hooke's law relating stresses to strains. Expanding eq.(2.65) considering $i, j = 1, ..., 6$ with x, y, z coordinates and displacements u, v, w as defined by Table 2.1, we get the generic form of the stiffness matrix

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{yz} \\ \sigma_{zx} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \epsilon_{yz} \\ \epsilon_{zx} \\ \epsilon_{xy} \end{bmatrix} \tag{2.68}$$

The 36 constant stiffness matrix is applicable for cases of anisotropic materials. In our analysis for this thesis, we consider only isotropic materials, though certainly, by properly defining this stiffness matrix amongst other parts of the computation (factoring in local material direction), we can consider other material types.

Isotropic materials do not see the coupling effects (extension-extension coupling, shear-extension coupling, or shear-shear coupling) seen in other material types do to infinite planes of material property symmetry, so the stiffness matrix can be modeled as follows:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{yz} \\ \sigma_{zx} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \epsilon_{yz} \\ \epsilon_{zx} \\ \epsilon_{xy} \end{bmatrix} \tag{2.69}$$

In the problem that we are going to solve, the ratio of the thickness of the plate to the its width is negligible, thus the stress field is mostly two-dimensional and the variation in the third dimension is thus negligible, plus the fact that the tensile stress is uniform and we don't have variation of tensile stress in the "z"-direction. Thus, the assumption of plane stress is valid in this case, given the following conditions

$$\sigma_z = 0, \qquad \sigma_{yz} = 0, \qquad \sigma_{zx} = 0 \tag{2.70}$$

such that

$$\epsilon_z = 0, \qquad \epsilon_{yz} = 0, \qquad \epsilon_{zx} = 0 \tag{2.71}$$

so the stiffness matrix in eq.(2.69) reduces to the form seen in eq.(2.66) [17].

Therefore, the shear stress tensor σ_{ij} in the momentum equation (2.61) can be computed using only the planar stress σ_x, σ_y and σ_{xy}. This is the reason we use the rearranged form eq.(2.65). Substituting eq.(2.66) and eq.(2.67) into eq.(2.65), the rearranged

stress vector can be written in the terms of displacement as follows.

$$\sigma_i = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & 0 \\ C_{21} & C_{22} & 0 \\ 0 & 0 & C_{66} \end{bmatrix} \times \begin{bmatrix} \partial_x & 0 \\ 0 & \partial_y \\ \partial_y & \partial_x \end{bmatrix} \times \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{u_k} \tag{2.72}$$

hence

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} C_{11}\partial_x u + C_{12}\partial_y v \\ C_{21}\partial_x u + C_{22}\partial_y v \\ C_{66}\left(\partial_y u + \partial_x v\right) \end{bmatrix} \tag{2.73}$$

In the stiffness matrix, $C_{ij} = C_{ji}$ and for isotropic materials there are no directional moduli or Poisson's ratio, so we can define the constants as follows:

$$C_{11} = C_{22} = \frac{E}{1-\nu^2} \tag{2.74}$$

$$C_{12} = C_{21} = \frac{\nu E}{1-\nu^2} \tag{2.75}$$

$$C_{66} = G = \frac{E}{2(1+\nu)} \tag{2.76}$$

Now we need to substitute (2.73) back into eq.(2.64). But before that let us expand eq.(2.64) to get the following equations:

$$\begin{aligned} \int_\Omega \left(\partial_x w\, \sigma_x + \partial_y w\, \sigma_{xy}\right) d\Omega &= \int_{\partial\Omega} w\, q_{n_1} + \int_\Omega w\, f_1 d\Omega \\ \int_\Omega \left(\partial_x w\, \sigma_{xy} + \partial_y w\, \sigma_y\right) d\Omega &= \int_{\partial\Omega} w\, q_{n_2} + \int_\Omega w\, f_2 d\Omega. \end{aligned} \tag{2.77}$$

where as described before $q_{n_1} = \sigma_{(i=1)j}\,\hat{n}_j = \sigma_x\,\hat{n}_1 + \sigma_{xy}\,\hat{n}_2$ and $q_{n_2} = \sigma_{xy}\,\hat{n}_1 + \sigma_y\,\hat{n}_2$ are secondary variables. Now substitute σ_x, σ_y and σ_{xy} from eq.(2.73) into (2.77) to obtain

$$\begin{aligned}\int_\Omega \left(\partial_x w\ \left(C_{11}\partial_x u + C_{12}\partial_y v\right) + \partial_y w\, C_{66}\left(\partial_y u + \partial_x v\right)\right) d\Omega &= \int_{\partial\Omega} w\, q_{n_1} + \int_\Omega w\, f_1 d\Omega \\ \int_\Omega \left(\partial_x w\, C_{66}\left(\partial_y u + \partial_x v\right) + \partial_y w\ \left(C_{21}\partial_x u + C_{22}\partial_y v\right)\right) d\Omega &= \int_{\partial\Omega} w\, q_{n_2} + \int_\Omega w\, f_2 d\Omega \end{aligned} \quad (2.78)$$

Now the primary variables can be expanded using basis functions $u = {}_j\, u_j$ and $v = \psi_j\, v_j$ and the test function can be selected exactly as the basis function $w = {}_i$ according to the Galerkin approach, with basis functions ${}_j$ as defined previously in eq.(2.32). Substituting in eq.(2.78) yields

$$\begin{aligned}\int_\Omega \left(\partial_x \psi_i\ \left(C_{11}\partial_x \psi_j\, u_j + C_{12}\partial_y \psi_j\, v_j\right) + \partial_y \psi_i\, C_{66}\left(\partial_y \psi_j\, u_j + \partial_x \psi_j v_j\right)\right) d\Omega &= \int_{\partial\Omega} {}_i\, q_{n_1} + \int_\Omega {}_i\, f_1 d\Omega \\ \int_\Omega \left(\partial_x \psi_i\, C_{66}\left(\partial_y \psi_j\, u_j + \partial_x \psi_j\, v_j\right) + \partial_y \psi_i\ \left(C_{21}\partial_x \psi_j\, u_j + C_{22}\partial_y \psi_j\, v_j\right)\right) d\Omega &= \int_{\partial\Omega} {}_i\, q_{n_2} + \int_\Omega {}_i\, f_2 d\Omega. \end{aligned} \quad (2.79)$$

Separating the coefficients of u_j and v_j we have

$$K_{11} = \int_\Omega \left(C_{11}\partial_x \psi_i\, \partial_x \psi_j + C_{66}\, \partial_y \psi_i\, \partial_y \psi_j\right) d\Omega \quad (2.80)$$

$$K_{12} = \int_\Omega \left(C_{12}\partial_x \psi_i\, \partial_y \psi_j + C_{66}\, \partial_y \psi_i\, \partial_x \psi_j\right) d\Omega \quad (2.81)$$

$$K_{21} = \int_\Omega \left(C_{12}\partial_y \psi_i\, \partial_x \psi_j + C_{66}\, \partial_x \psi_i\, \partial_y \psi_j\right) d\Omega \quad (2.82)$$

$$K_{22} = \int_\Omega \left(C_{22}\partial_y \psi_i\, \partial_y \psi_j + C_{66}\, \partial_x \psi_i\, \partial_x \psi_j\right) d\Omega \quad (2.83)$$

Note that $K_{12ij} = K_{21ij}^T$. Therefore the elemental equations are

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} \int_{\partial\Omega} \psi_i \, (q_{n_1} + f_1) \, d\Omega \\ \int_{\partial\Omega} \psi_i \, (q_{n_2} + f_2) \, d\Omega \end{bmatrix} \tag{2.84}$$

CHAPTER 3

IMPLEMENTATION

Development of this code has been done in Fortran90 and all code testing has been run on a mid-2012 Apple MacBook Pro with Intel(R) Core(TM) i7-3615QM CPU @ 2.30GHz and 3.8GB RAM running Ubuntu 19.04.

The technique we use begins with the establishment of the boundary which is read through a segment file format. This segment file was generated by defining the two dimensional geometric shape and then meshing with *Gmsh*. [12] The boundary elements were then extracted from the `.msh` file generated by *Gmsh* and written to the segment file format with a simple python script. For our base mesh shown in Figure 2.14, the segment file is shown in the following code snippet.

```
9
2.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
3.7272370428296950E+00  0.0000000000000000E+00  0.0000000000000000E+00
6.1339629235271911E+00  0.0000000000000000E+00  0.0000000000000000E+00
9.4874863223968866E+00  0.0000000000000000E+00  0.0000000000000000E+00
1.4160273052153840E+01  0.0000000000000000E+00  0.0000000000000000E+00
2.0671317893899960E+01  0.0000000000000000E+00  0.0000000000000000E+00
2.9743783130882210E+01  0.0000000000000000E+00  0.0000000000000000E+00
4.2385322944434051E+01  0.0000000000000000E+00  0.0000000000000000E+00
6.0000000000000000E+01  0.0000000000000000E+00  0.0000000000000000E+00
3
6.0000000000000000E+01  0.0000000000000000E+00  0.0000000000000000E+00
6.0000000000000000E+01  1.1294557270628010E+01  0.0000000000000000E+00
6.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
```

```
7
6.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
5.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
4.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
3.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
2.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
1.0000000000000000E+01  2.0000000000000000E+01  0.0000000000000000E+00
0.0000000000000000E+00  2.0000000000000000E+01  0.0000000000000000E+00
6
0.0000000000000000E+00  2.0000000000000000E+01  0.0000000000000000E+00
0.0000000000000000E+00  1.3773710213962760E+01  0.0000000000000000E+00
0.0000000000000000E+00  9.2788037877479717E+00  0.0000000000000000E+00
0.0000000000000000E+00  6.0338247573893931E+00  0.0000000000000000E+00
0.0000000000000000E+00  3.6911978286910490E+00  0.0000000000000000E+00
0.0000000000000000E+00  2.0000000000000000E+00  0.0000000000000000E+00
4
0.0000000000000000E+00  2.0000000000000000E+00  0.0000000000000000E+00
1.0000000000000000E+00  1.7320508075688770E+00  0.0000000000000000E+00
1.7320508075688781E+00  9.9999999999999933E-01  0.0000000000000000E+00
2.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
```

After reading and initializing the input boundary geometry and specifying any interior hole locations, the interior is meshed with using *Triangle*. [31] This gives us a basic $p = 1$ mesh, akin to the meshes obtained using commercial FEM solvers. At this point, we can specify and inject our high order points, by setting the parameters `pin`, to set p-order, `eltypein`, which selects either Lagrange or Fekete basis functions, `tolerance`, which specifies the duplicate node tolerance, and then calling function `add_more_points` to then generate the high order points via an a posteriori high order meshing process. The code is quite flexible allowing the selection of somewhat arbitrary p-order (Fekete is restricted to a p-order in

multiples of 3 and has been tested up to $p = 9$, Lagrange is tested up to $p = 8$), restricted mostly by computer memory capacity.

```
1  allocate(pin(grd%ntri), eltypein(grd%ntri))
2  pin = 3
3  eltypein = 1  !1 for fekete at higher order
4  tolerance = 1.0d-13
5  call add_more_points(grd, pin, eltypein, tolerance, galtype = 'PG')
```

The function `add_more_points` distributes additional points on the interior and boundary of each element in dependent on the basis selected. If Fekete bases are selected, this routine implements eq.(2.17) to generate the approximate Fekete points in each element. To achieve the curved boundaries, the boundary nodes in the mesh initialization process, are used to generate NURBS (Non-Uniform Rational B-Spline) representation of the geometry present to which the additional points can be generated upon utilizing the basis functions described previously in this text.

Next, the material properties are initialized in the function `init_material_props` by calculating the constants C_{11}, C_{12}, C_{21}, C_{22}, and C_{66} as seen in eqs.(2.74 - 2.76). Then, for each element, all elements are initialized in function `init_elem` and then the elemental stiffness matrices are computed in function `comp_elem_KQf`.

```
1  call init_material_props(nu = .33d0, E = 10600000.0d0)
2  allocate(elems(grd%ncellsg))
3  do e = 1, grd%ncellsg
4     call init_elem(elems(e), e, grd, neqs)
5     call comp_elem_KQf(elems(e), e, grd)
6     ngausstotal = ngausstotal + elems(e)%ngauss
7  end do
```

In function `init_elem`, each element is set up for computation by computing the element properties necessary for use in later steps. This includes the computation of the r and s coordinates, the weight function W, the basis function ψ, and its derivatives, and the Jacobian of transformation, its inverse and its value at each Gauss point. The function also sets up the elemental stiffness matrix K, for computation in the next step as well as the neighbor and edge tag data of the element. Further detail on the `init_elem` function can be seen in Appendix A.

Function `comp_elem_KQf` is effectively the implementation of calculating eqs.(2.80), (2.81), (2.82), and (2.83). It uses the initialized elements calculated in function `init_elem` to complete this calculation.

```
1   do i = 1, npe
2      do j = 1, npe
3         do k = 1, elem%ngauss
4            ! get grad of basis functions
5            der1(1) = elem%d_psi_d_xi(i,k); der1(2) = elem%d_psi_d_eta(i,k)
6            der2(1) = elem%d_psi_d_xi(j,k); der2(2) = elem%d_psi_d_eta(j,k)
7            ! transform computational grads to physical grads
8            der1 = matmul(elem%Jstar(:,:,k), der1)
9            der2 = matmul(elem%Jstar(:,:,k), der2)
10           psi_i_x = der1(1)
11           psi_i_y = der1(2)
12           psi_j_x = der2(1)
13           psi_j_y = der2(2)
14
15           ! filling entries of the stiffness matrix explicitly
16           elem%K(1,1,i,j) = elem%K(1,1,i,j) &
17           + (C11 * psi_i_x * psi_j_x + C66 * psi_i_y * psi_j_y) * coeff *
     elem%JJ(k) * elem%W(k)
18           elem%K(1,2,i,j) = elem%K(1,2,i,j) &
```

```
19            + (C12 * psi_i_x * psi_j_y + C66 * psi_i_y * psi_j_x) * coeff *
      elem%JJ(k) * elem%W(k)
20            elem%K(2,1,i,j) = elem%K(2,1,i,j) &
21            + (C12 * psi_i_y * psi_j_x + C66 * psi_i_x * psi_j_y) * coeff *
      elem%JJ(k) * elem%W(k)
22            elem%K(2,2,i,j) = elem%K(2,2,i,j) &
23            + (C22 * psi_i_y * psi_j_y + C66 * psi_i_x * psi_j_x) * coeff *
      elem%JJ(k) * elem%W(k)
24         end do
25      end do
26   end do
```

Next, after the elemental stiffness matrices have been calculated for all elements. The stiffness matrices are assembled to the global stiffness matrix, boundary conditions are applied, then the diagonal of the matrix is averaged in a matrix conditioning improvement routine, and the matrix is then solved for u and v in as in eq.(2.84) utilizing *LAPACK*. At this point, the solution for strain has been solved, however, some further computation is required to get the solution for stress.

The boundary conditions applied in our solution are Dirichlet and Neumann boundary conditions. The Dirichlet boundary conditions are fully high order compatible, however in the current state of our implementation, the Neumann boundary conditions implemented in a hybrid way, in that they are computed in a low order method and adapted to the high order mesh. We can still simulate the analytical solution of this domain by increasing the x-dimension of the domain such that any abnormalities from the Neumann boundary edge with applied load (as seen in Figure 2.14) are diminished where $x = 0$. Implementing the Neumann boundary conditions in a full high order method is an area of future research in this method. Here, we see the code for our implementation of the Dirichlet boundary condition.

```
! loop over 2 start and end nodes on that edge
do k = 1, 2
   pt = grd%ibedge(i,k) ! global number of that node
   x = grd%x(pt)
   y = grd%y(pt)
   KK(ieq,:,pt,:) = 0.0d0 ! freeze that row
   do j = ieq, ieq
      KK(j,j,pt,pt) = 1.0d0 ! ones on the diagonal
   end do
   call f_bc(id)%f(x,y,tmp)
   rhs(ieq,pt) = tmp(ieq)
end do ! k - loop

! start looping over nodes inside that edge
if ( allocated(grd%el2edg(ielem)%edg1) ) then
   ! associate a universal pointer to points on that edge.
   select case (iedg)
   case (1)
      inter_pts => grd%el2edg(ielem)%edg1
   case (2)
      inter_pts => grd%el2edg(ielem)%edg2
   case (3)
      inter_pts => grd%el2edg(ielem)%edg3
   case (4) ! only quad
      inter_pts => grd%el2edg(ielem)%edg4

   case default
      print *, 'something is wring in applying BCs' &
           , ' on psoints inside a boundary edge! stop'
      stop
   end select
```

```
32      ! if at least one point exists on that edge
33      if ( size(inter_pts) >= 1 ) then
34         ! loop over all interior pts on that edge
35         do k = 1, size(inter_pts)
36            pt = inter_pts(k) ! global number of that node
37            x = grd%x(pt)
38            y = grd%y(pt)
39            KK(ieq,:,pt,:) = 0.0d0 ! freeze that row
40            do j = ieq, ieq
41               KK(j,j,pt,pt) = 1.0d0 ! ones on the diagonal
42            end do
43            call f_bc(id)%f(x,y,tmp)
44            rhs(ieq,pt) = tmp(ieq)
45         end do ! k - loop
46      end if
47
48   end if ! there exists bn pts on that edge
```

Next, we see the code for the hybrid implementation of the Neumann boundary condition

```
1   pt1 = grd%ibedge(i,1) ! global number of node1
2   pt2 = grd%ibedge(i,2) ! global number of node2
3   x1 = grd%x(pt1); x2 = grd%x(pt2)
4   y1 = grd%y(pt1); y2 = grd%y(pt2)
5   ll = sqrt((x2 - x1)**2.0d0 + (y2 - y1)**2.0d0)
6   call f_bc(id)%f(x1,y1,tmp)
7   rhs(ieq,pt1) = rhs(ieq,pt1) + tmp(ieq) * ll / 2.0d0
8   rhs(ieq,pt2) = rhs(ieq,pt2) + tmp(ieq) * ll / 2.0d0
```

To increase the efficiency of the solution and therefore reducing the condition number of the matrix, a diagonal averaging technique is used. This technique works by computing the average of the diagonal (`tmp` in the code snippet below) and then replacing all values on the matrix diagonal $K_{ii} = 1$ and multiplying the corresponding values on the right-hand side of the matrix solution by the average.

```
1  diag_size = neqs * grd%nnodesg
2  tmp = 0
3  i = 0
4
5  do i = 1, diag_size
6    tmp = tmp + input(i,i)
7  end do
8
9  tmp = tmp / diag_size
10
11 i = 0
12
13 do i = 1, diag_size
14   if (input(i,i) == 1.0d0) then
15     input(i,i) = tmp
16     rhs(i) = rhs(i) * tmp
17   end if
18 end do
```

The assembly of the elemental stiffness matrix and right-hand side to the global stiffness matrix (referred to in our code as `KK`) and right-hand side is achieved with a simple function that positions the elements and nodes correctly in the global stiffness matrix utilizing the connectivity matrix called `icon`.

```
1  ! start assembling
```

```
do k = 1, grd%ncellsg ! loop over all cells
   elem => elems(k) ! load this element
   npe = grd%npe(k)
   do i = 1, npe
      pti = icon(k,i)
      rhs(:,pti) = rhs(:,pti) + elem%Q(:, i) + elem%f(:, i)

      do j = 1, npe
         ptj = icon(k,j)
         KK(:,:,pti, ptj) = KK(:,:,pti, ptj) + elem%K(:,:,i,j)
      end do

   end do

end do
```

In eq.(2.65), we see that the stress σ_i is related to the strain ϵ_j and the constant C_{ij}. This is expanded via tensor math, to eq.(2.73), which is the exact formula we use to calculate the final stress solution.

Before calculating this solution, we must first (due to our implementation) calculate the elemental gradients $\partial_x u$, $\partial_x v$, $\partial_y u$ and $\partial_y v$ for each Gauss point k in the element. This is done in our code by calling function `comp_grad_u_point`. These gradients are calculated in the r, s coordinate system, but our subsequent calculations require them to be in the x, y coordinate system. We can solve this by mapping the gradients using function `rs2xy`. This is all demonstrated in our code snippet following

```
i = 0
do e = 1, grd%ncellsg
  do k = 1, elems(e)%ngauss
```

```
4       call comp_grad_u_point(u,grd,elems(e),e,elems(e)%r(k),elems(e)%s(k),
      dudxe, dudye)
5       call rs2xy(elems(e),elems(e)%r(k),elems(e)%s(k), xtemp, ytemp)
6
7       i = i + 1
8       dudxt(:,i) = dudxe(:)
9       dudyt(:,i) = dudye(:)
10    end do
11  end do
```

With the gradients mapped, we can then calculate the stress solutions σ_x, σ_y, and σ_{xy} according to eq.(2.73).

```
1   do nnode = 1, ngausstotal
2     ustress(1, nnode) = (dudxt(1,nnode) * C11) + (dudyt(2,nnode) * C12)
3     ustress(2, nnode) = (dudxt(1,nnode) * C21) + (dudyt(2,nnode) * C22)
4     ustress(3, nnode) = (dudxt(2,nnode) * C66) + (dudyt(1,nnode) * C66)
5   end do
```

From this point, all the calculations are completed, we can then output our results. We chose to use the TECPLOT ASCII format for output due to its simplicity and versatility. The TECPLOT ASCII format can be read by *VisIt* [7], our choice for visualization.

CHAPTER 4

RESULTS

Displacement

Utilizing the code described in Implementation, we compute the solutions for the domain described in Figure 2.14 for varying orders of p up to $p = 8$ for the Lagrange case and $p = 9$ for the Fekete case. Shown first in this Results chapter are the displacement contours for the generic domain. Notice the curved elements caused by the increased elemental resolution on the high order solutions and the higher fidelity and accuracy of the solutions. Figures 4.1 through 4.6 demonstrate the displacement contours calculated using Lagrange bases.

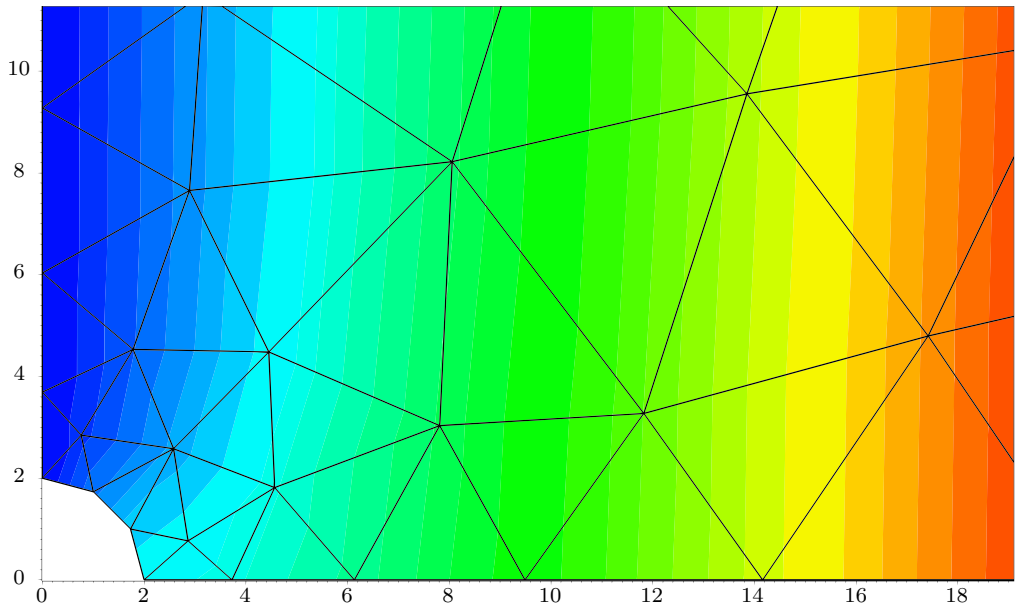


Figure 4.1 u solution on $p = 1$ grid with Lagrange Basis

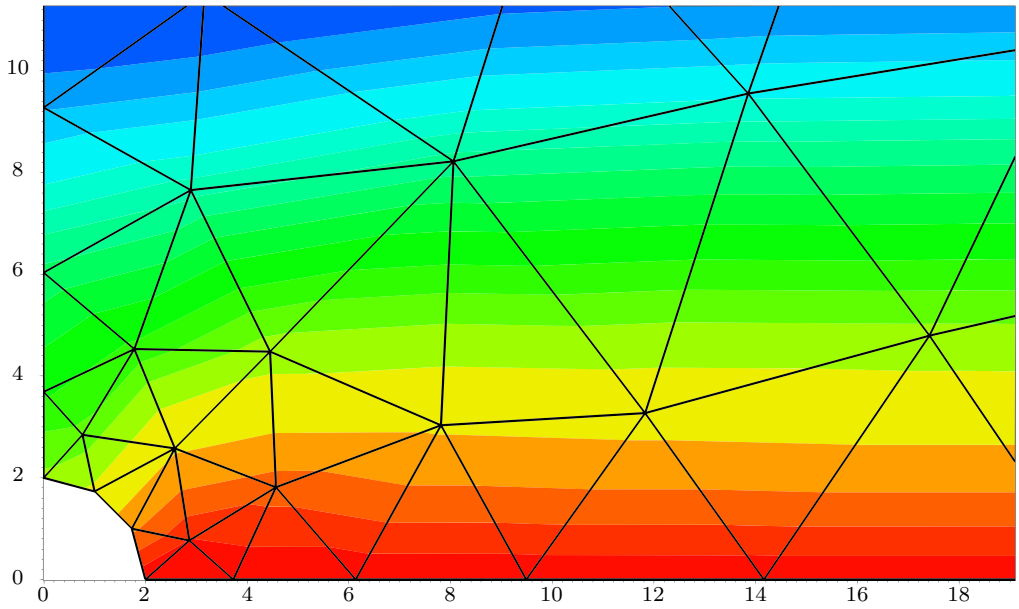


Figure 4.2 v solution on $p = 1$ grid with Lagrange Basis

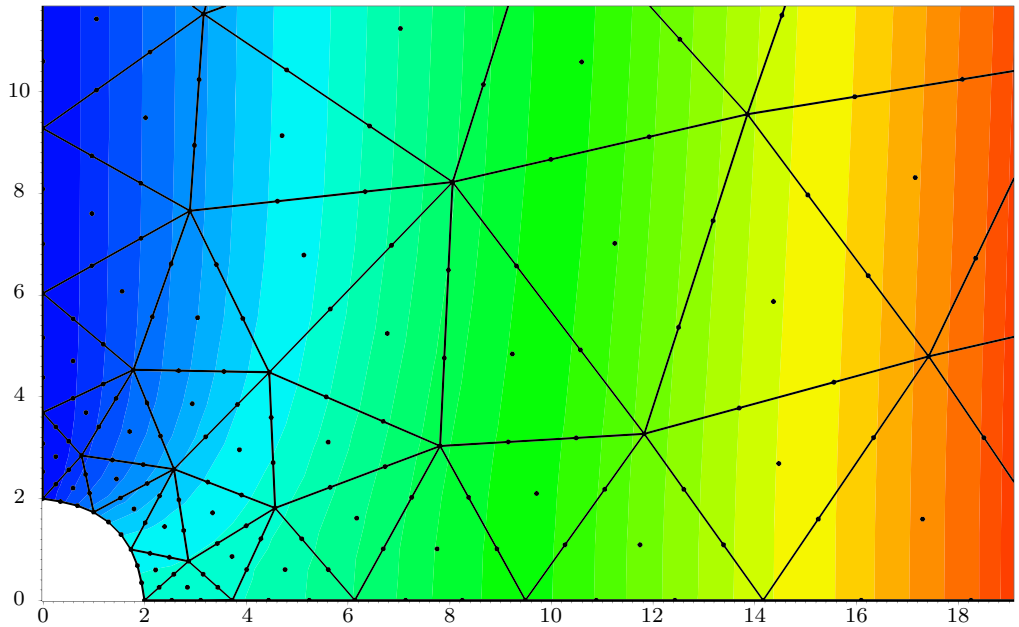


Figure 4.3 u solution on $p = 3$ grid with Lagrange Basis

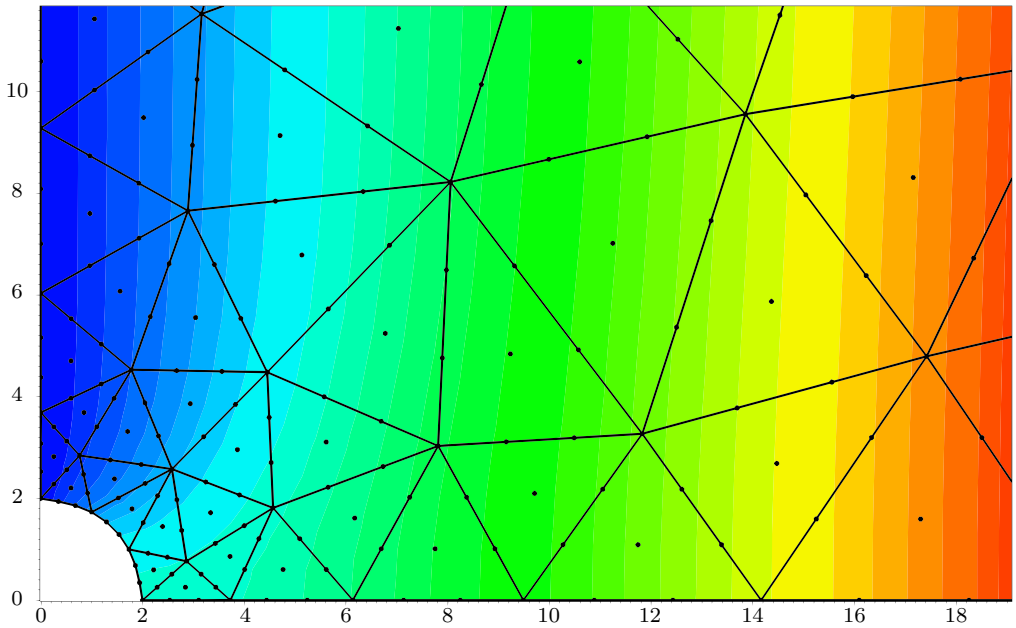


Figure 4.4 v solution on $p = 3$ grid with Lagrange Basis

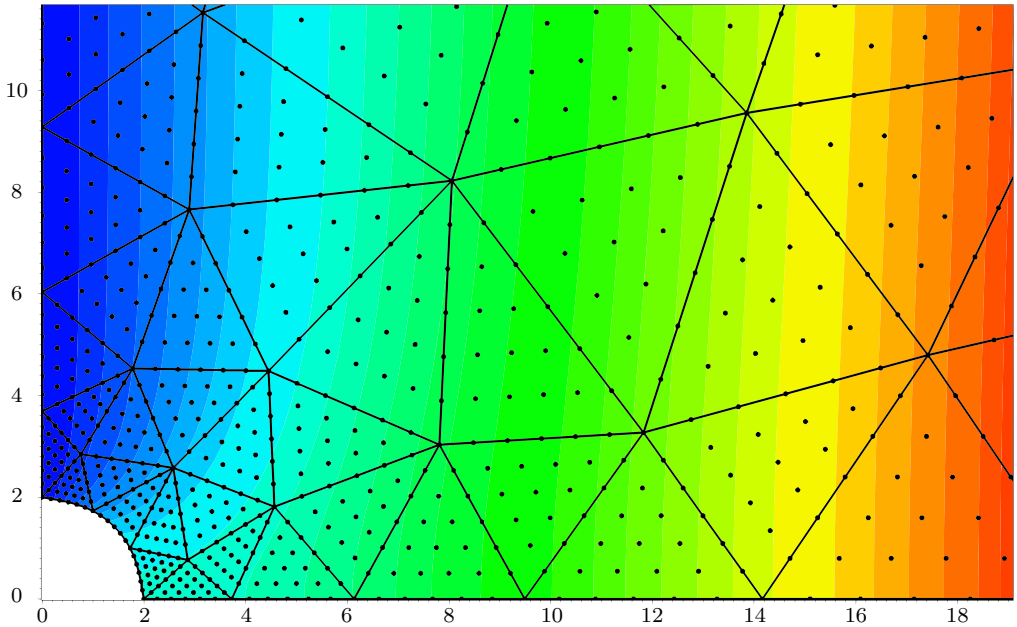


Figure 4.5 u solution on $p = 6$ grid with Lagrange Basis

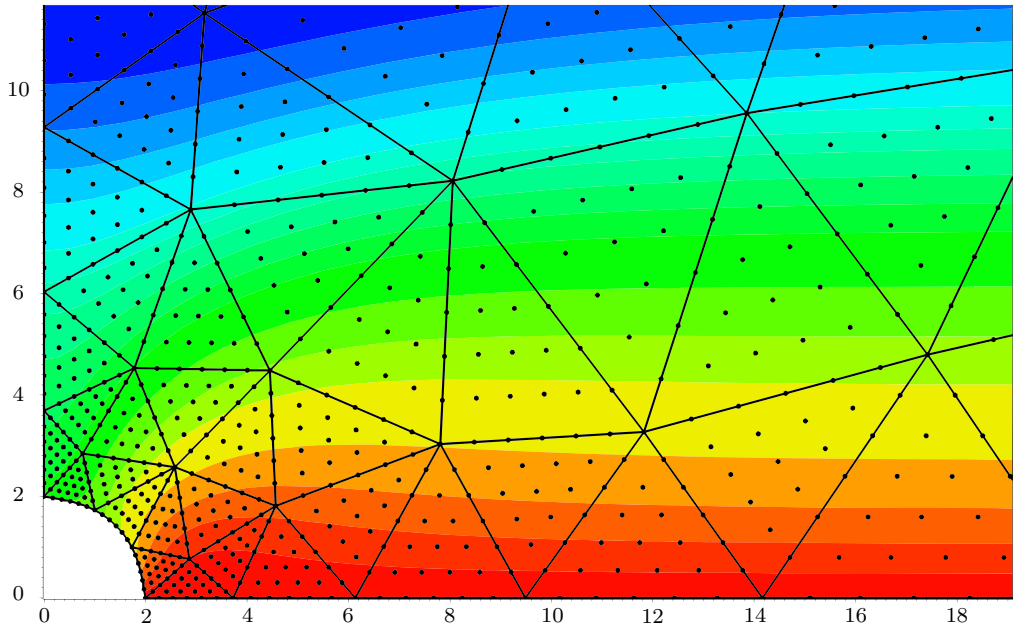


Figure 4.6 v solution on $p = 6$ grid with Lagrange Basis

We demonstrate in Figures 4.7 through 4.12, the displacement solution using Fekete bases. Note our ability to solve up to $p = 9$. With greater computer resources, this could be pushed much further. Applying the approximate Fekete points derived in eq.(2.17), we can solve the elasticity problem using the Fekete Spectral Element method Figure 4.8, which when compared to the result of traditional techniques using Lagrange Basis in Figure (4.2), show how these techniques can achieve a more accurate result as a high p-resolution low h-order mesh than with a just a low p-resolution mesh.

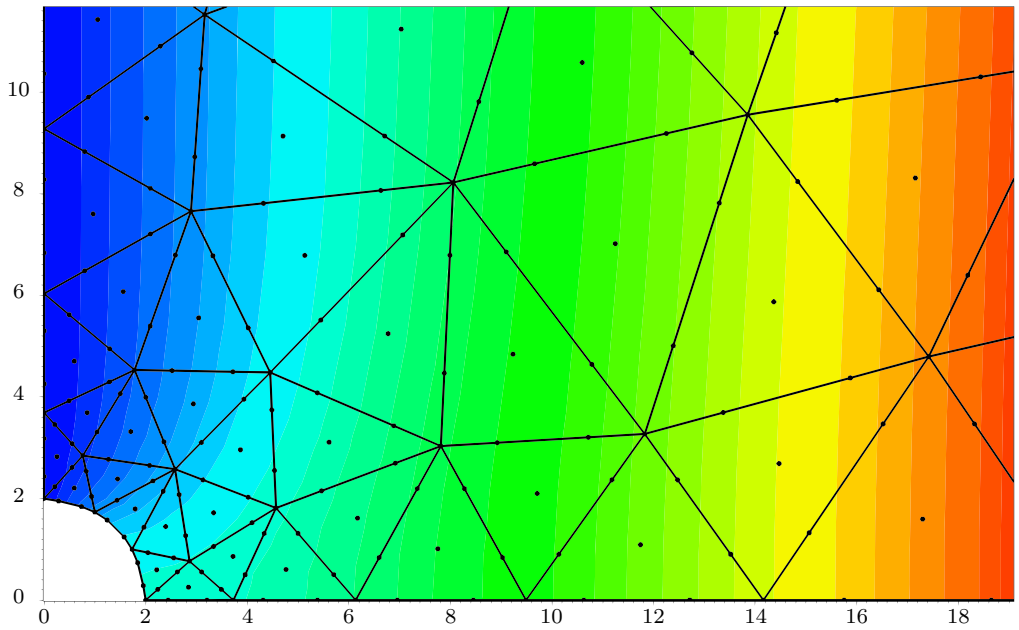


Figure 4.7 u solution on $p = 3$ grid with Fekete Basis

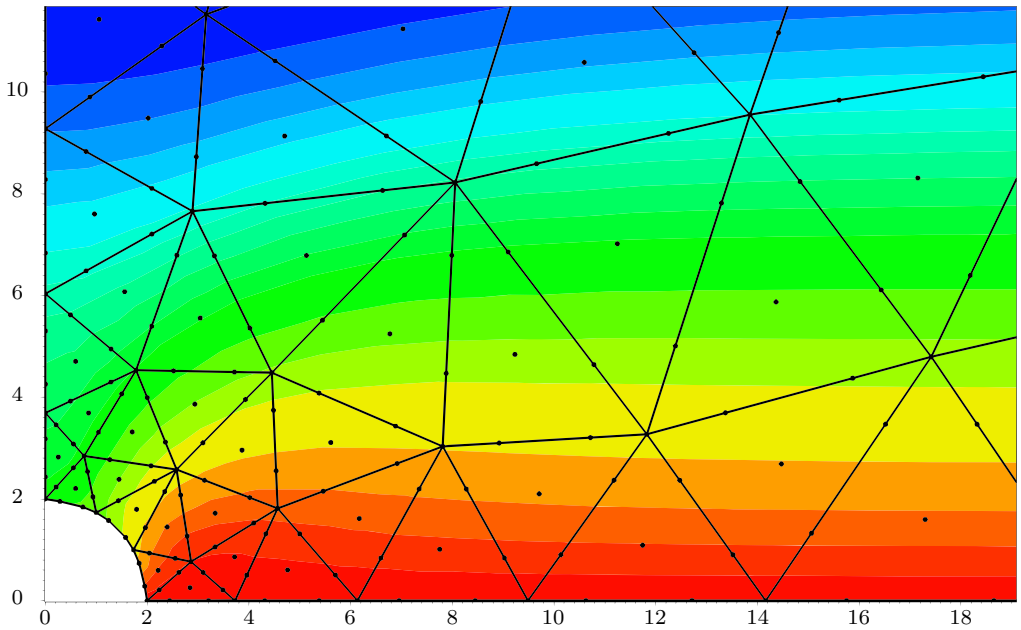


Figure 4.8 v solution on $p = 3$ grid with Fekete Basis

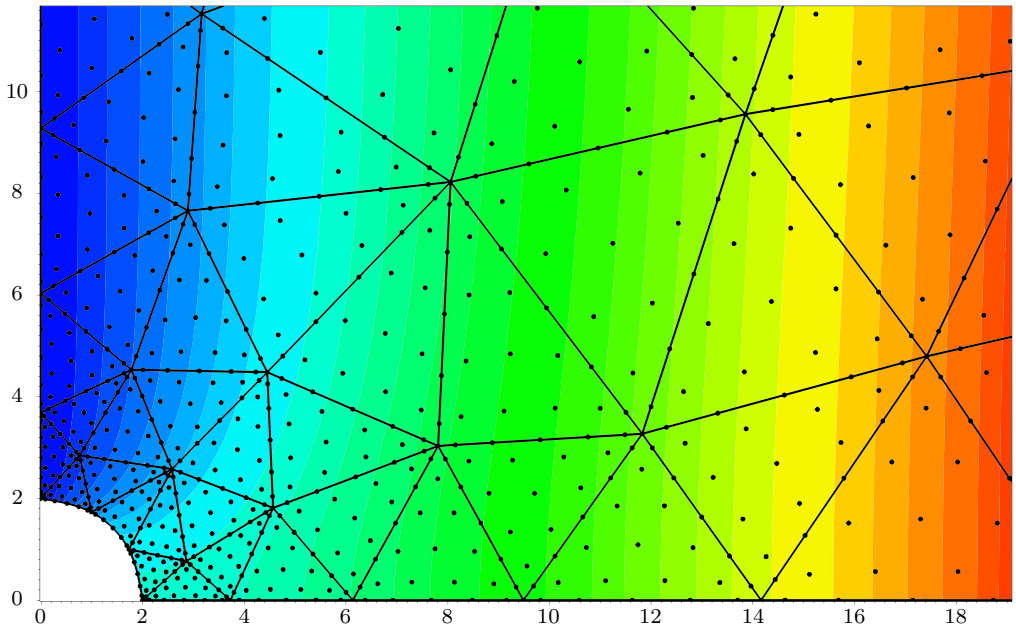


Figure 4.9 u solution on $p = 6$ grid with Fekete Basis

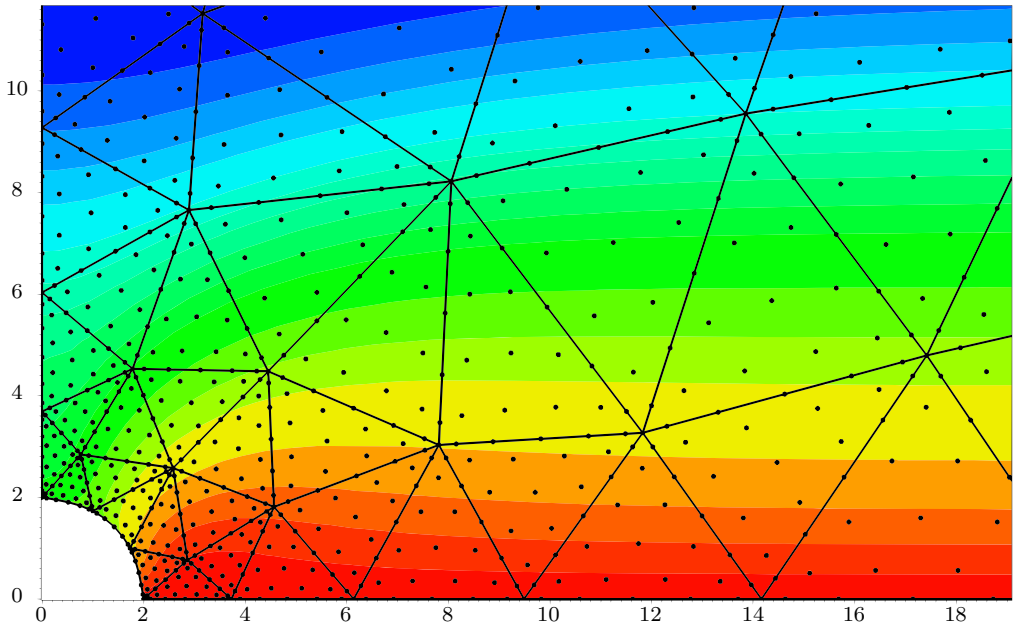


Figure 4.10 v solution on $p = 6$ grid with Fekete Basis

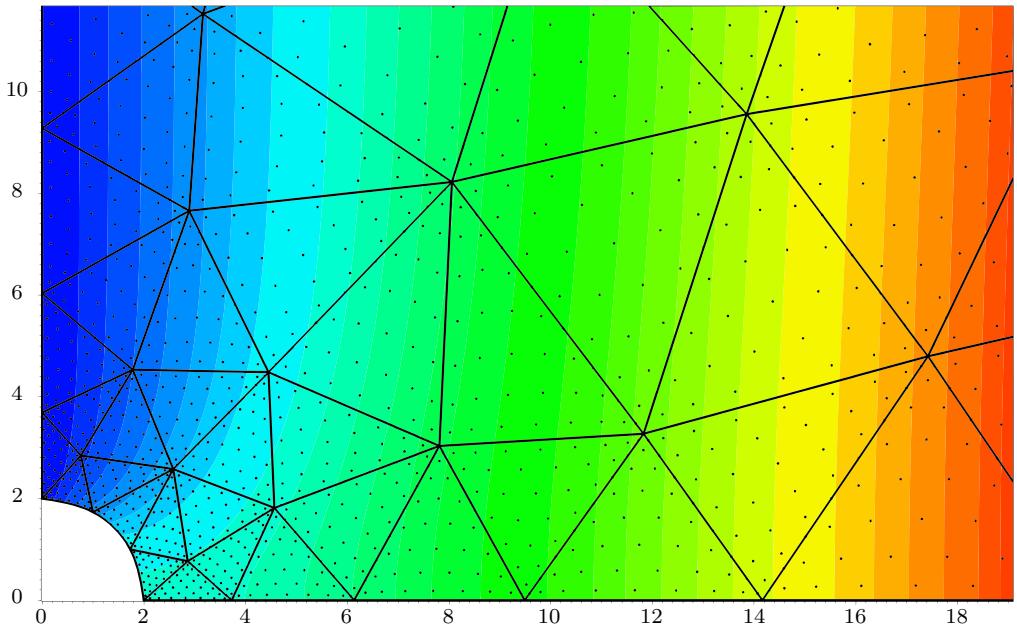


Figure 4.11 u solution on $p = 9$ grid with Fekete Basis

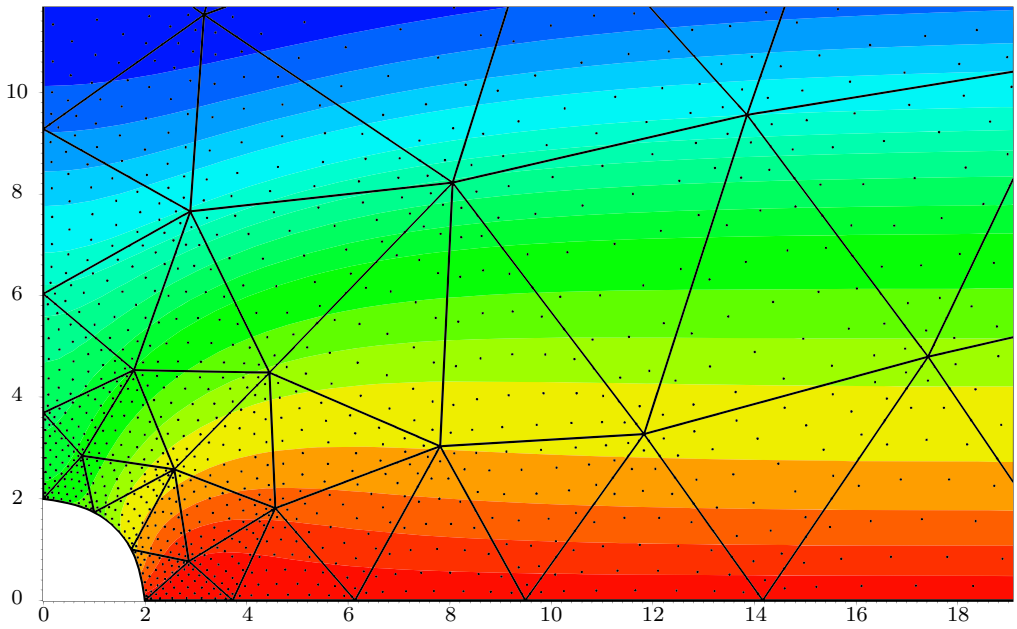


Figure 4.12 v solution on $p = 9$ grid with Fekete Basis

To compare the two techniques and check their accuracy to the analytical solution, the result was evaluated along both the $x = 0$ and $y = 0$ axes and plotted against the analytical solution. First, in Figures 4.13 and 4.15, we examine $p = 1$ Lagrange and $p = 3$ Fekete techniques. Note the ability to achieve a high accuracy result with minimal order or p.

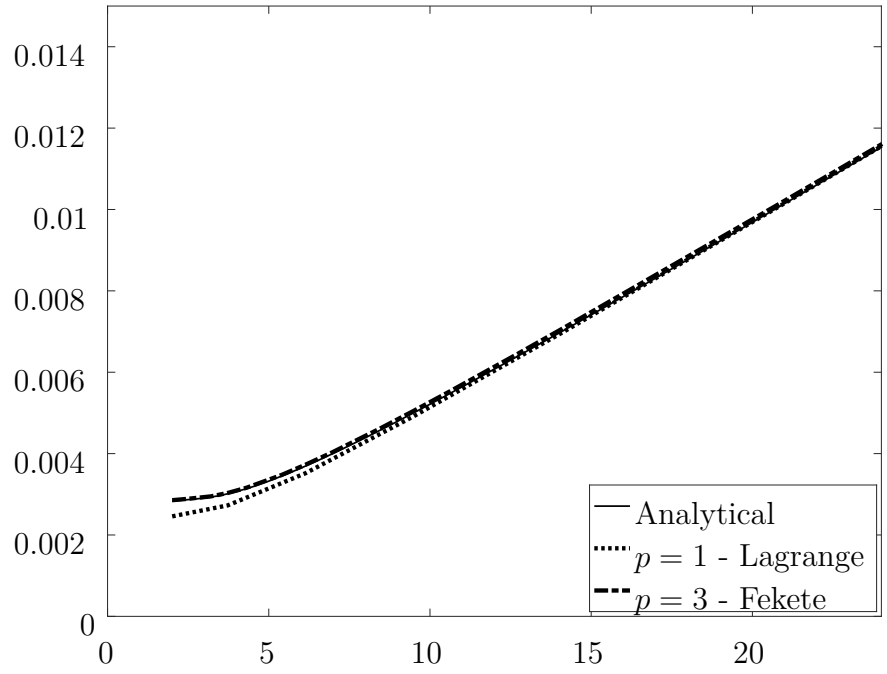


Figure 4.13 u solution comparing $p = 1$ Lagrange and $p = 3$ Fekete techniques with analytical solution

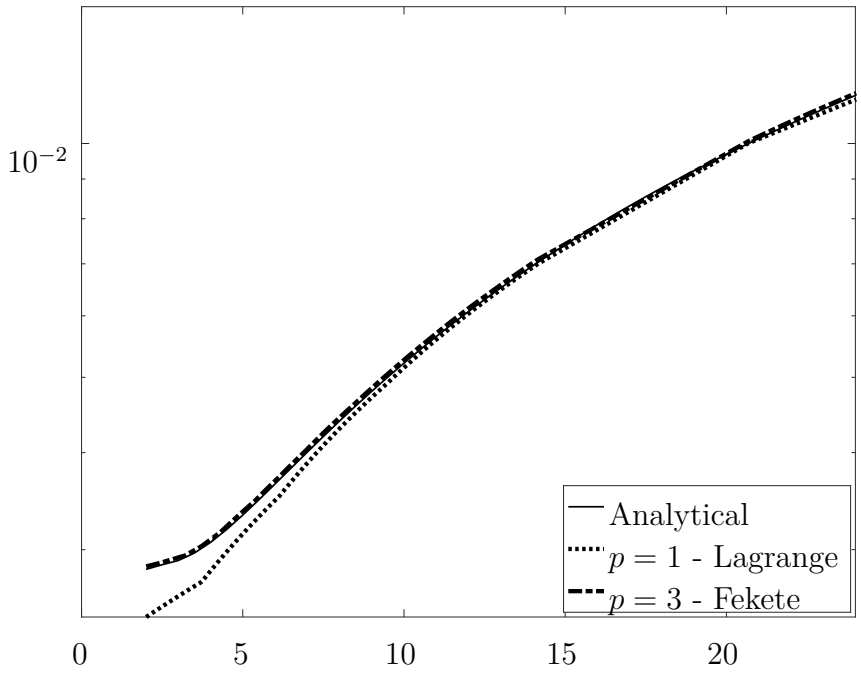


Figure 4.14 u solution comparing $p = 1$ Lagrange and $p = 3$ Fekete techniques with analytical solution (semi-log y plot)

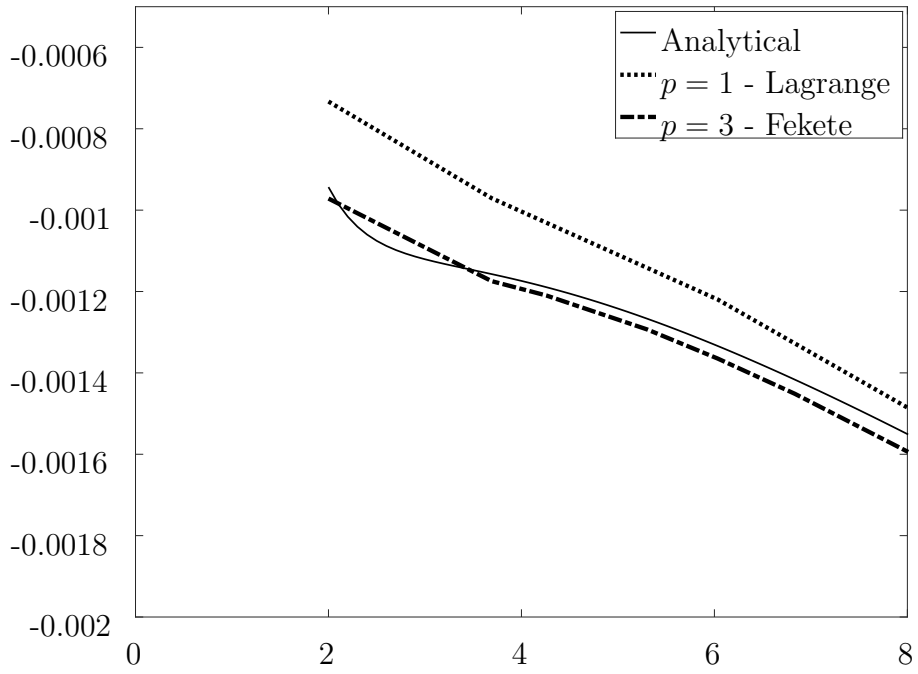


Figure 4.15 v solution comparing $p = 1$ Lagrange and $p = 3$ Fekete techniques with analytical solution

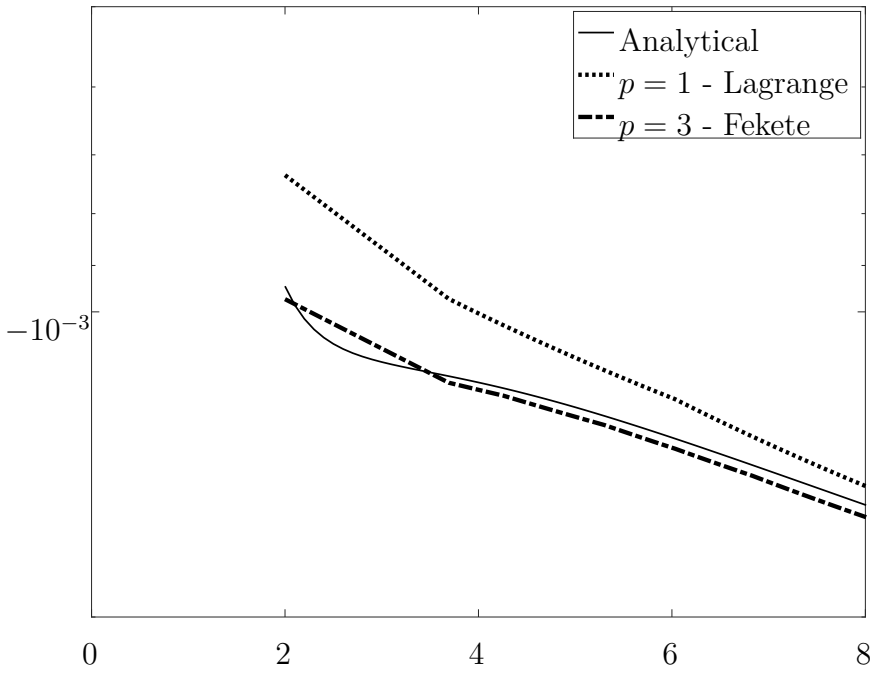


Figure 4.16 v solution comparing $p = 1$ Lagrange and $p = 3$ Fekete techniques with analytical solution (semi-log y plot)

Comparing equally higher order Fekete and Lagrange techniques in Figures 4.17 and 4.19 using the same methodology shows a similar correlation with the analytical solution, but also with each other. But when comparing the matrix condition number as in Figure 4.21 and Table 4.1, the efficiency of the Fekete techniques show, with a much more gradual curve comparatively.

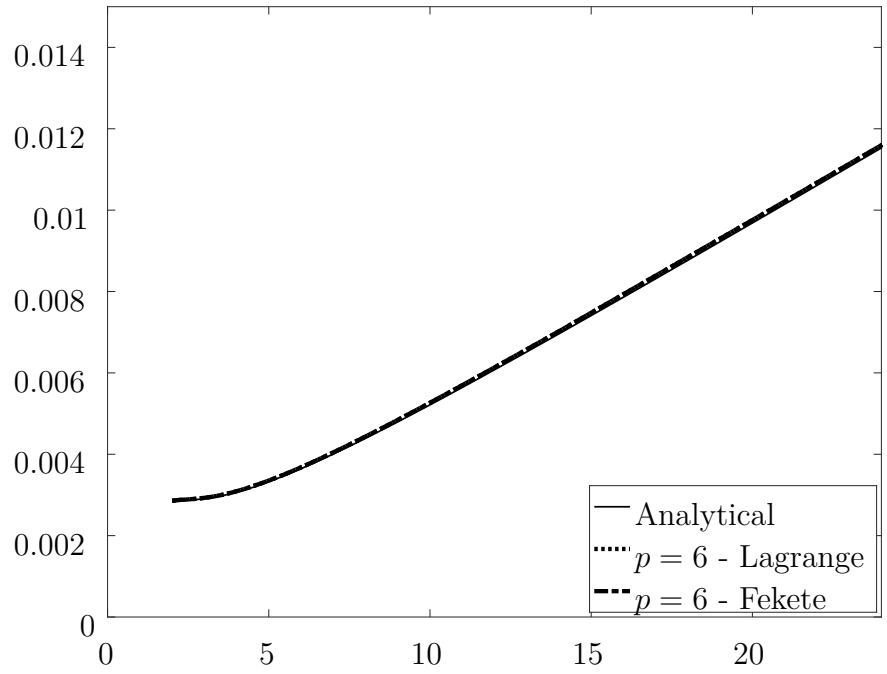


Figure 4.17 u solution comparing $p = 6$ Lagrange and $p = 6$ Fekete techniques with analytical solution

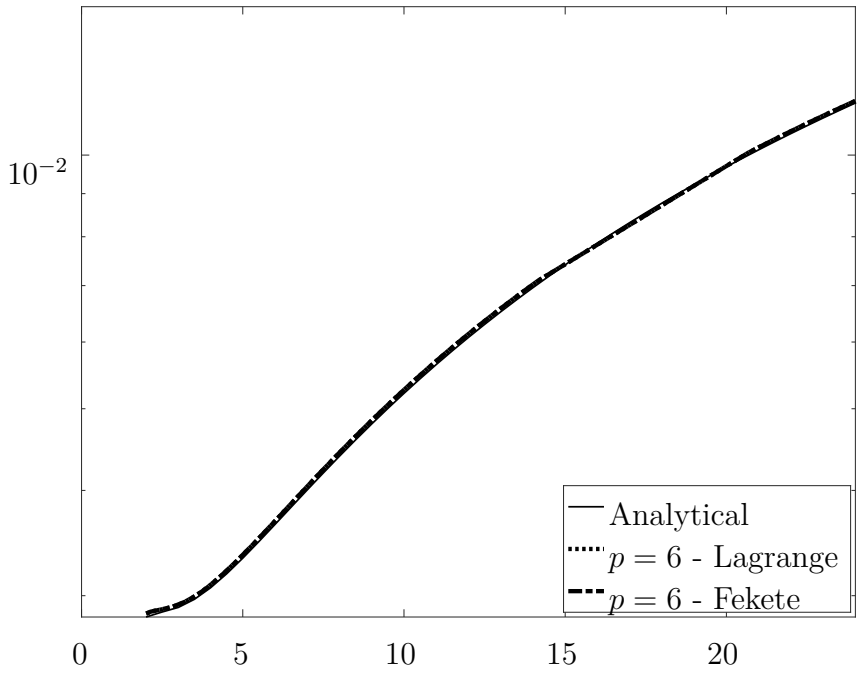


Figure 4.18 u solution comparing $p = 6$ Lagrange and $p = 6$ Fekete techniques with analytical solution (semi-log y plot)

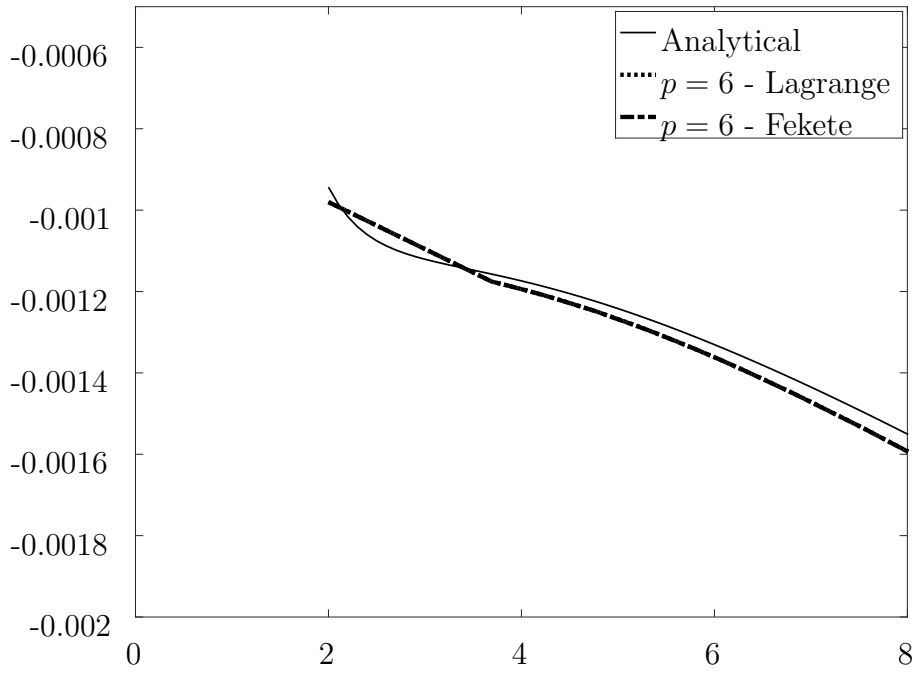


Figure 4.19 v solution comparing $p = 6$ Lagrange and $p = 6$ Fekete techniques with analytical solution

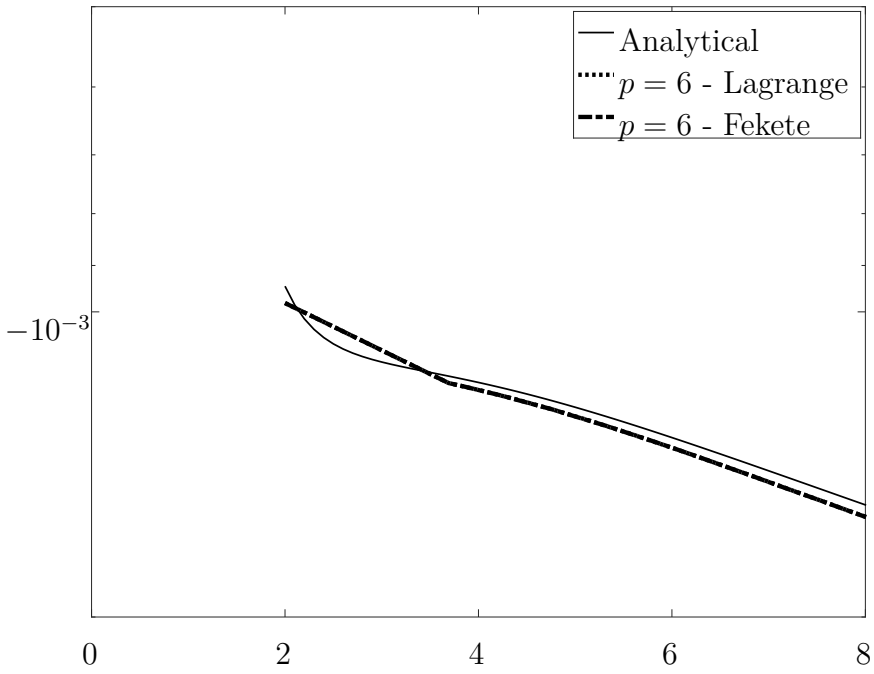


Figure 4.20 v solution comparing $p = 6$ Lagrange and $p = 6$ Fekete techniques with analytical solution (semi-log y plot)

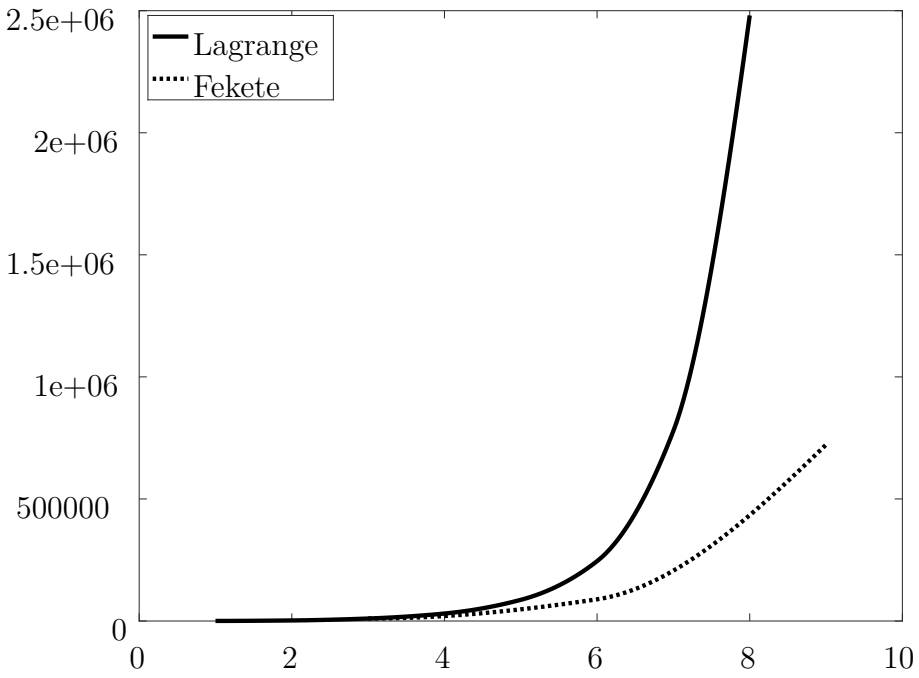


Figure 4.21 Comparison of Matrix condition number of higher order FEM methods utilizing Lagrange and Fekete techniques

Table 4.1 Comparison of Matrix condition number of higher order FEM methods utilizing Lagrange and Fekete techniques

p	Lagrange	Fekete
1	364.5510	
2	2319.899	
3	10213.14	10279.03
4	30656.84	
5	86674.31	
6	245345.7	88985.87
7	779342.1	
8	2480589	
9		722376.3

Stress

The visualization of the stress results is still in a primitive stage. At this point, we have not been able to explicitly visualize these results using our normal methods with VisIt. To visualize our stress results, we have utilized radial basis function interpolation with modifications for visualizing Fekete nodal distributions. This loses accuracy compared to pure spectral methods, but is sufficient to showcase our results.

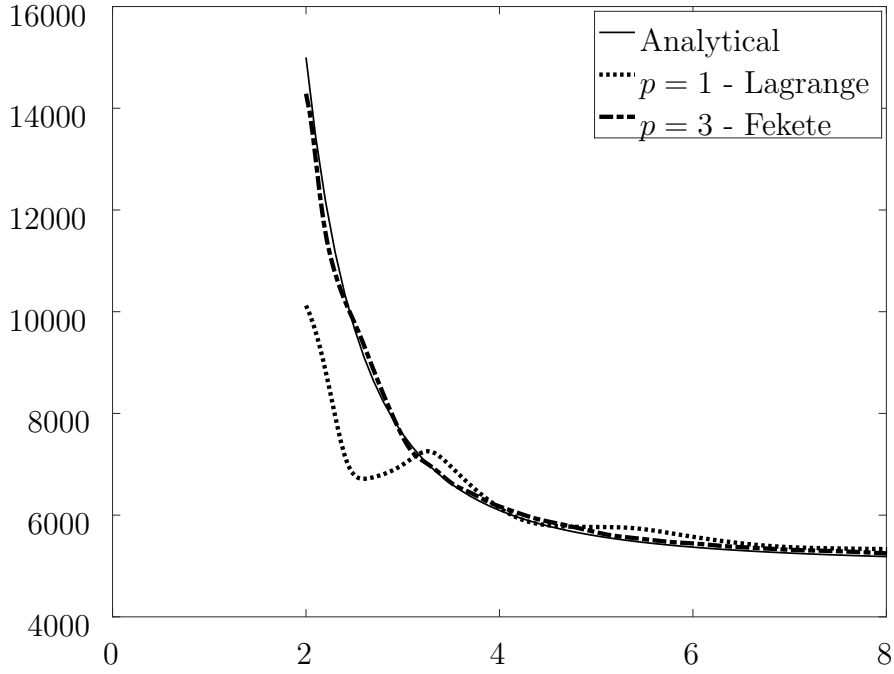


Figure 4.22 σ_x solution comparing $p = 1$ Lagrange and $p = 3$ Fekete techniques with analytical solution where $x = 0$

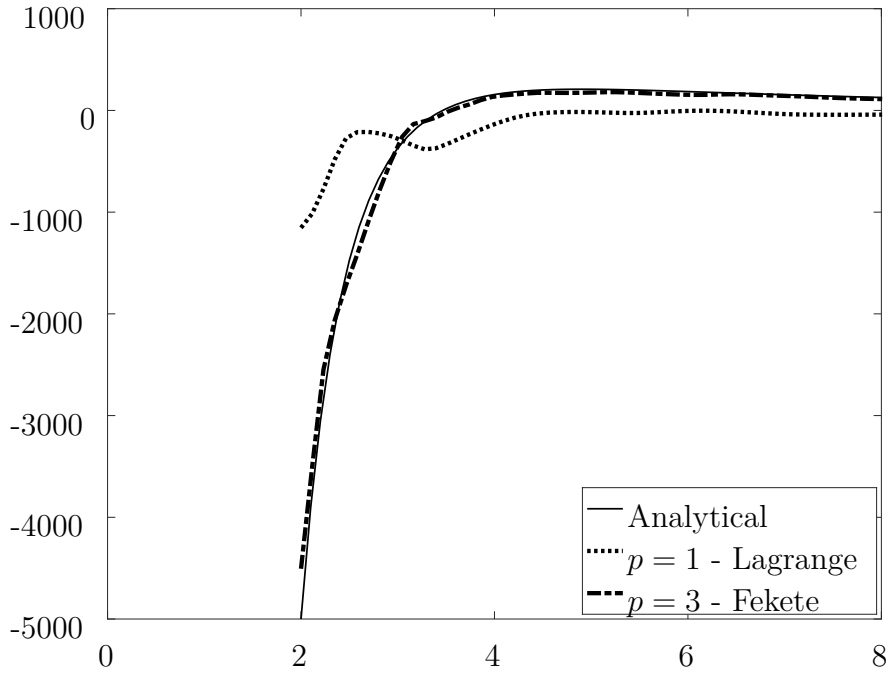


Figure 4.23 σ_y solution comparing $p = 1$ Lagrange and $p = 3$ Fekete techniques with analytical solution where $y = 0$

Figures 4.22 and 4.23 again demonstrate that the $p = 1$ solution does not provide the accuracy with a course mesh like the Fekete SEM technique does. Comparing $p = 6$ solutions in Figures 4.24 and 4.25, we again demonstrate that the Fekete method is comparable with traditional Lagrange methods, again with the greater efficiency that we demonstrated previously in Figure 4.21 and Table 4.1.

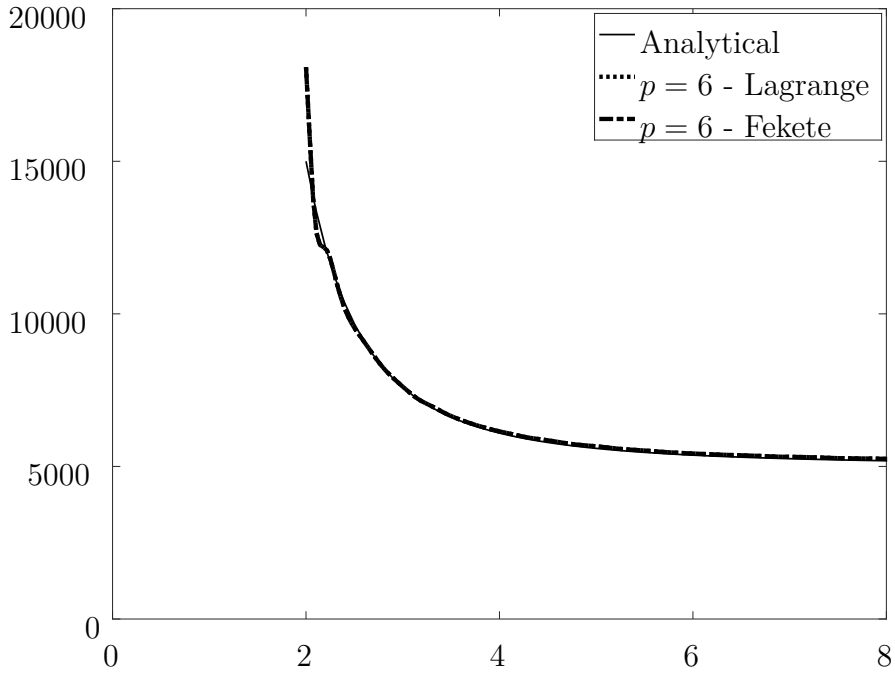


Figure 4.24 σ_x solution comparing $p = 6$ Lagrange and $p = 6$ Fekete techniques with analytical solution where $x = 0$

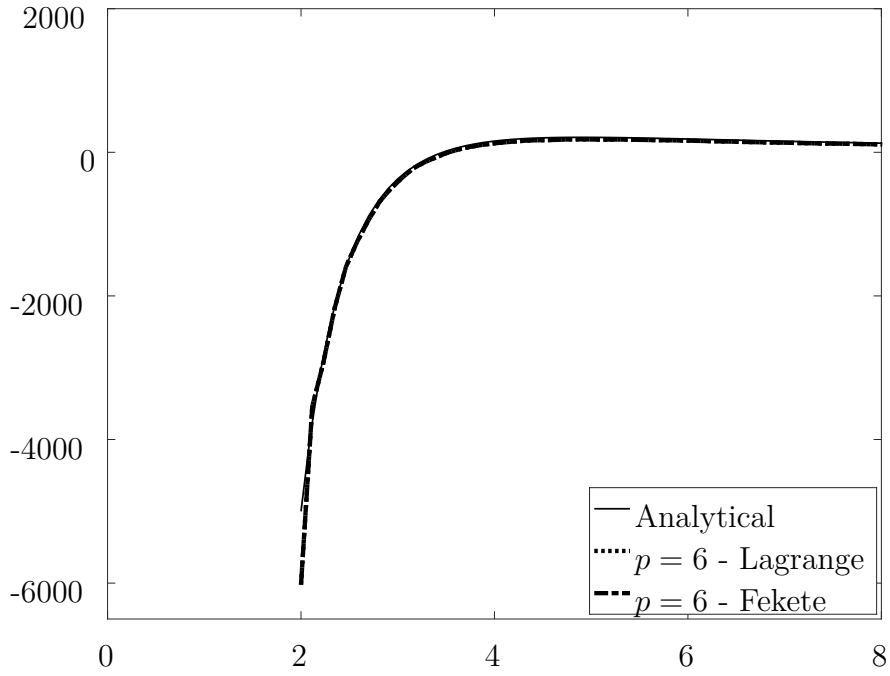


Figure 4.25 σ_y solution comparing $p = 6$ Lagrange and $p = 6$ Fekete techniques with analytical solution where $y = 0$

Complex Geometry

Complex geometry solutions can also be obtained using our solver. Adding holes to our same domain, we show the same displacement solution for the $p = 6$ Fekete case in Figures 4.26 and 4.27. Comparing these solutions to Figures 4.28 and 4.29, it can be seen that the $p = 1$ techniques at this mesh resolution do not provide adequate solutions.

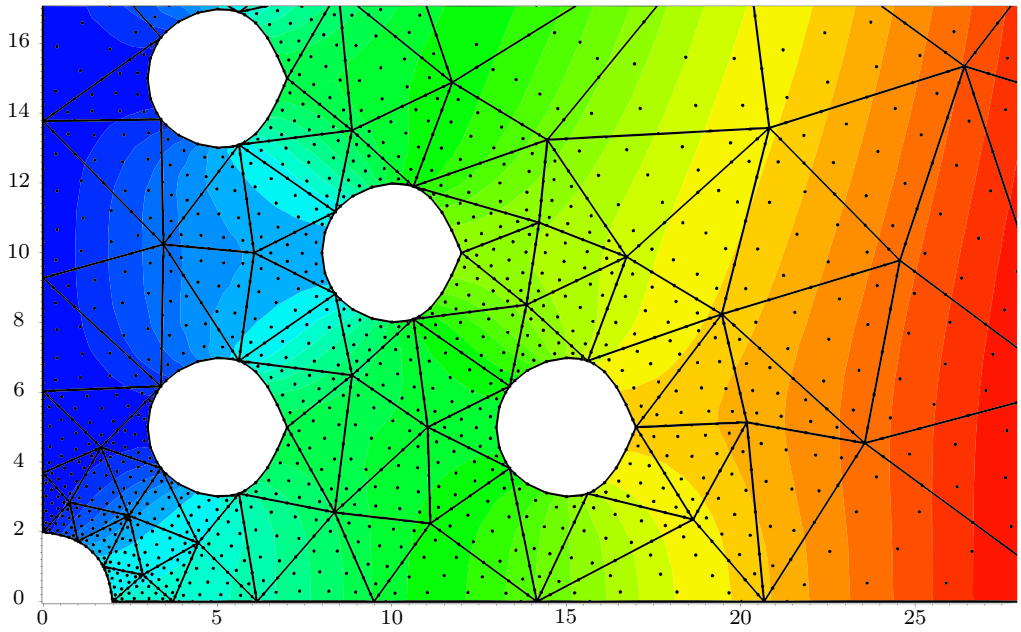


Figure 4.26 u solution of Complex Geometry data using $p = 6$ Fekete technique

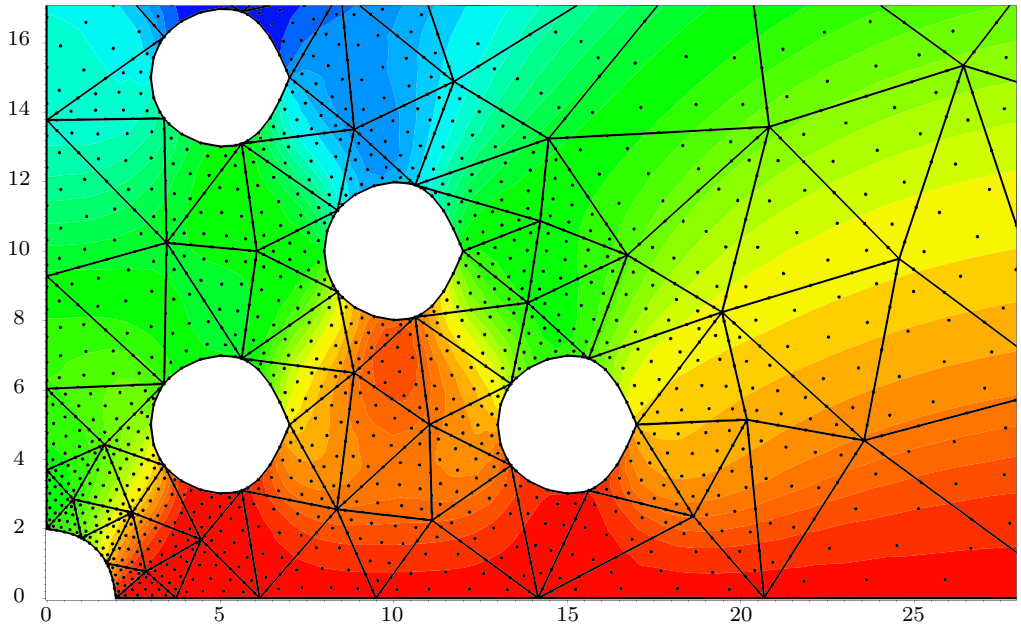


Figure 4.27 v solution of Complex Geometry data using $p = 6$ Fekete technique

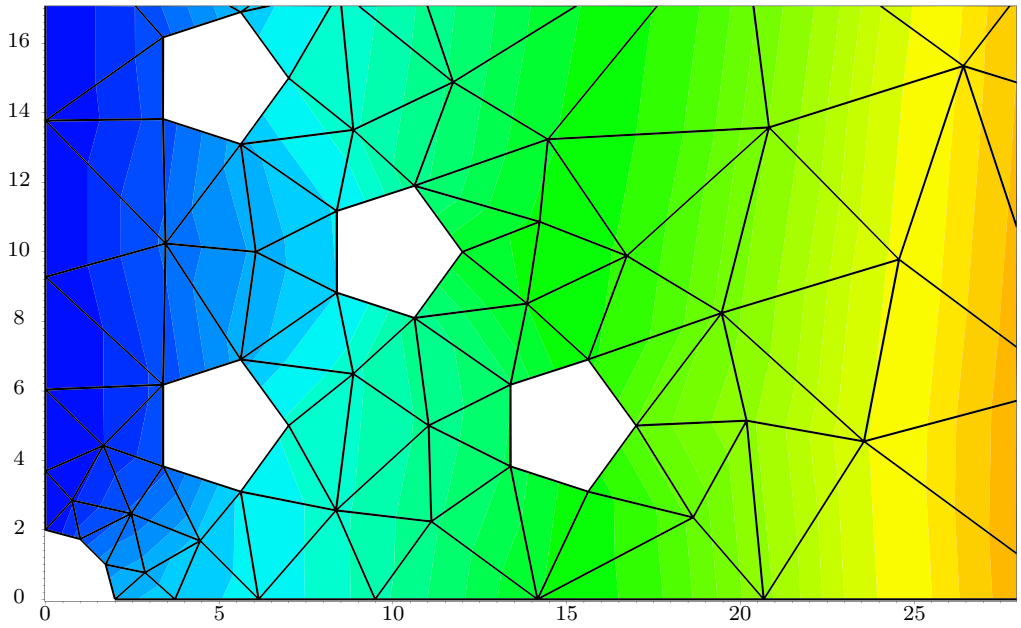


Figure 4.28 u solution of Complex Geometry data using $p = 1$ Lagrange technique

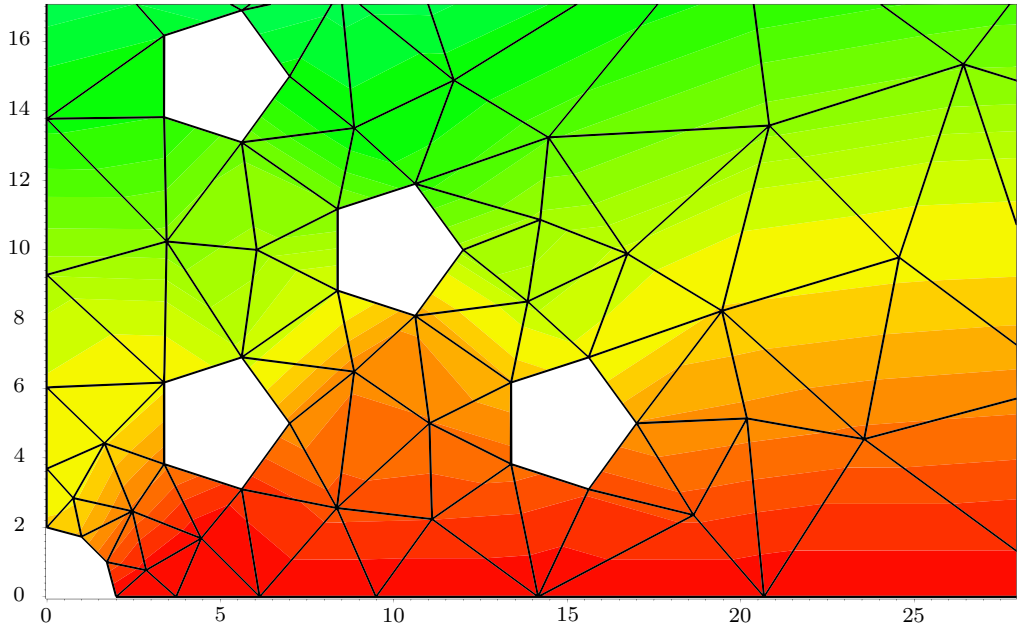


Figure 4.29 v solution of Complex Geometry data using $p = 1$ Lagrange technique

Likewise, we compare the $p = 6$ Fekete and $p = 1$ Lagrange techniques applied to the stress solution in Figures 4.30 and 4.31, with similar display of results to the displacement solution; at this mesh courseness, the solution generated by the $p = 1$ technique is not adequate.

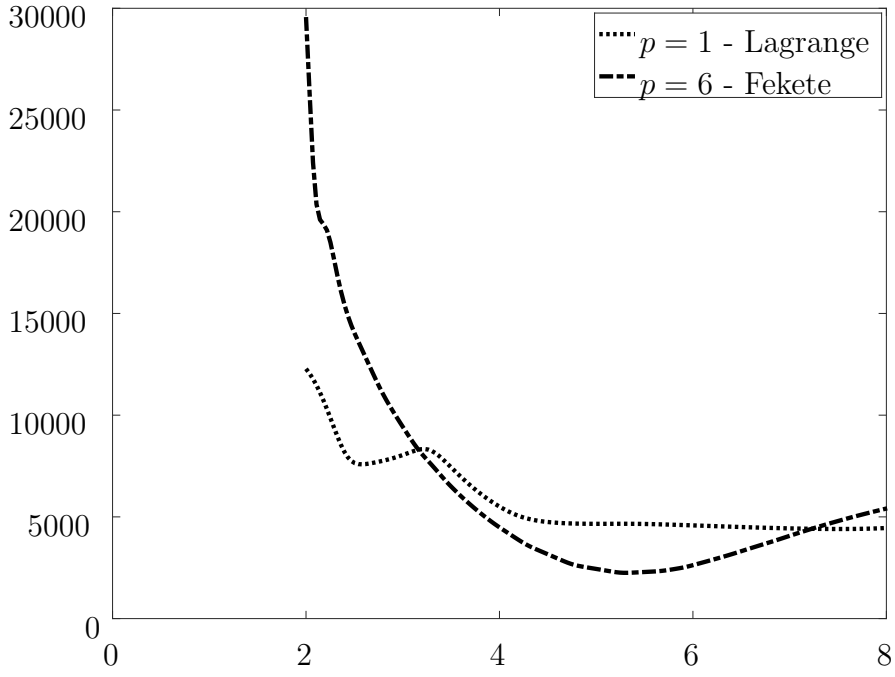


Figure 4.30 σ_x solution of Complex Geometry data comparing $p = 1$ Lagrange and $p = 6$ Fekete techniques where $x = 0$

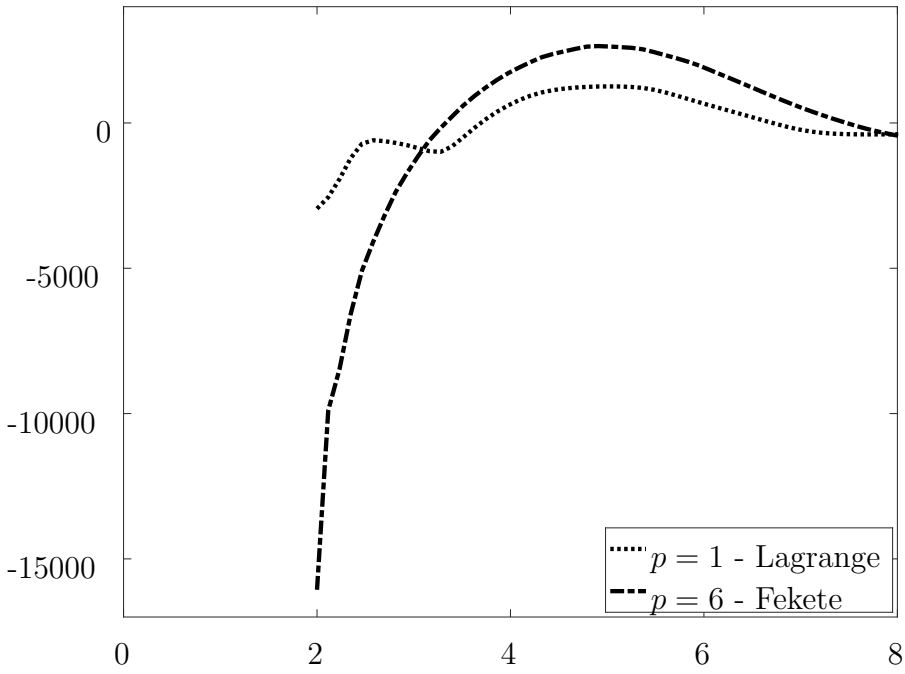


Figure 4.31 σ_y solution of Complex Geometry data comparing $p = 1$ Lagrange and $p = 6$ Fekete techniques where $y = 0$

CHAPTER 5

CONCLUSIONS

The results of this thesis give us a method that has greater accuracy than traditional low order FEM when comparing the same mesh, and higher efficiency compared to traditional Lagrangian FEM when comparing matrix condition numbers at different p-orders. This produces a flatter curve when varying p-order, which shows that higher orders than $p = 9$ as tested here should be easily achievable with this technique and the proper resources. We demonstrate the application of SEM to solid mechanics, and we also show the flexibility of our code by proving that it can also solve solutions with complex geometry and holes, which demonstrates the unlimited potential of this technique with more refinement.

Future research and improvement opportunities

As mentioned previously, the code can be easily improved by implementing proper high order Neumann boundary conditions, as well as implementing further boundary condition types and element types such as quads and hexagons. Other improvements would be to improve the parameter input method such that recompilation of the code is not necessary to specify different input files, boundary condition changes, or other parameter changes, such as p-refinement, etc. To expand upon the efficiency, implementation of sparse matrices and an iterative solver would be the logical next steps. Beyond that, implementing a non-linear solver and further solid mechanics methods such as contact or fracture mechanics would fully realize the capability of this solver.

Impact of research

As was stated in the introduction, traditionally, this method is applied to problems with smooth solutions, such as those found in fluid dynamics [2, 23, 38]. In this thesis, we show the applicability of this method to solid mechanics in the solution of elasticity with complex geometry with good accuracy. The advantage of this method is to allow for the solution of these problems with fewer elements than traditional methods allow. This decreases the computational resources needed for the solution of the problem, allowing for either the same problems to be solved faster or problems with greater complexity to be solved than traditional FEM considering the same computational resources available and achieving greater accuracy with doing so.

REFERENCES

[1] Artioli, E., Beirão da Veiga, L. & Dassi, F. 2020 Curvilinear virtual elements for 2d solid mechanics applications. *Computer Methods in Applied Mechanics and Engineering* **359**, 112667. 5

[2] Babuška, I. 1988 The p and h-p versions of the finite element method: The state of the art. In *Finite Elements* (ed. D. L. Dwoyer, M. Y. Hussaini & R. G. Voigt), pp. 199–239. New York, NY: Springer New York. 1, 9, 86

[3] Bassi, F. & Rebay, S. 1997 High-order accurate discontinuous finite element solution of the 2d euler equations. *Journal of Computational Physics* **138** (2), 251 – 285. 3

[4] Beirão da Veiga, L., Brezzi, F. & Marini, L. D. 2013 Virtual elements for linear elasticity problems. *SIAM Journal on Numerical Analysis* **51** (2), 794–812. 4

[5] Bittencourt, Marco, Vazquez, Thais & Nogueira Junior, Alberto 2007 High-order finite elements applied to structural elastic problems. In *Solid Mechanics in Brazil 2007,* (ed. Marcílio Alves & H.S. da Costa Mattos), pp. 109–123. Brazilian Society of Mechanical Sciences and Engineering. 10

[6] Bruckstein, A. M., Elad, M. & Zibulevsky, M. 2008 On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *IEEE Transactions on Information Theory* **54** (11), 4813–4820. 21

[7] Childs, Hank, Brugger, Eric, Whitlock, Brad, Meredith, Jeremy, Ahern, Sean, Pugmire, David, Biagas, Kathleen, Miller, Mark, Harrison, Cyrus, Weber, Gunther H., Krishnan, Hari, Fogal, Thomas, Sanderson, Allen, Garth, Christoph, Bethel, E. Wes, Camp, David, Rübel, Oliver, Durant, Marc, Favre, Jean M. & Navrátil, Paul 2012 VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pp. 357–372. 60

[8] Düster, A., Parvizian, J., Yang, Z. & Rank, E. 2008 The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering* **197** (45), 3768 – 3782. 6

[9] Franklin, W. R. 1970 Pnpoly - point inclusion in polygon test. *Published online* . 21

[10] Funaro, D. 1992 *Polynomial Approximation of Differential Equations. Lecture Notes in Physics Monographs* . Springer Berlin Heidelberg. 28

[11] Gervasio, Paola, Dede, Luca, Chanon, Ondine & Quarteroni, Alfio 2020 A computational comparison between isogeometric analysis and spectral element methods: Accuracy and spectral properties. *Journal of Scientific Computing* **83**, Article number: 18. 8

[12] Geuzaine, Christophe & Remacle, Jean-François 2009 Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* **79** (11), 1309–1331. 51

[13] Ghasemi, Arash 2016 Spectral hulls: a degree of freedom reducing hp-strategy in space/time. PhD thesis, University of Tennessee at Chattanooga. 16, 28, 32, 33, 34, 35, 42

[14] Ghasemi, A., Taylor, L. K. & Newman III, J. C. 2016 Spectral/hp hull: A degree of freedom reducing discontinuous spectral element method for conservation laws with application to compressible fluid flow. 16, 20

[15] Giraud, Luc, Langou, Julien, Rozložník, Miroslav & Eshof, den Jasper van 2005 Rounding error analysis of the classical gram-schmidt orthogonalization process. *Numerische Mathematik* **101** (1), 87–100. 23

[16] Hughes, T.J.R., Cottrell, J.A. & Bazilevs, Y. 2005 Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* **194** (39), 4135 – 4195. 7

[17] Jones, Robert M. 1999 *Mechanics of Composite Materials*, 2nd edn. Taylor & Francis Group. 47

[18] Konovalov, D., Vershinin, A., Zingerman, K. & Levin, V. 2017 The implementation of spectral element method in a cae system for the solution of elasticity problems on hybrid curvilinear meshes. *Modelling and Simulation in Engineering* **2017**, 7. 15

[19] Mallat, S. G. & Zhang, Zhifeng 1993 Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing* **41** (12), 3397–3415. 21

[20] Melenk, J.M., Gerdes, K. & Schwab, C. 2001 Fully discrete hp-finite elements: fast quadrature. *Computer Methods in Applied Mechanics and Engineering* **190** (32), 4339 – 4364. 12

[21] Mitchell, William F. 2015 How high a degree is high enough for high order finite elements? *Procedia Computer Science* **51**, 246 – 255, international Conference On Computational Science, ICCS 2015. 11

[22] Netz, T., Düster, Alexander & Hartmann, Stefan 2013 High-order finite elements compared to low-order mixed element formulations. *ZAMM-Zeitschrift für Angewandte Mathematik und Mechanik* **93**, 163–176. 10

[23] Pasquetti, Richard & Rapetti, Francesca 2004 Spectral element methods on triangles and quadrilaterals: Comparisons and applications. *J. Comput. Phys.* **198** (1), 349–362. 1, 13, 86

[24] Pasquetti, Richard & Rapetti, Francesca 2006 Spectral element methods on unstructured meshes: Comparisons and recent advances. *J. Sci. Comput.* **27**, 377–387. 13

[25] Pavarino, Luca, Zampieri, Elena, Pasquetti, Richard & Rapetti, Francesca 2007 Overlapping schwarz methods for fekete and gauss-lobatto spectral elements. *SIAM Journal on Scientific Computing* **29**, 1073–1092. 12

[26] Persson, Per-Olof & Strang, Gilbert 2004 A simple mesh generator in matlab. *SIAM Review* **46** (2), 329–345. 22

[27] Poya, Roman, Sevilla, Ruben & Gil, Antonio J. 2016 A unified approach for a posteriori high-order curved mesh generation using solid mechanics. *Computational Mechanics* **58** (3), 457–490. 4

[28] Rank, E., Ruess, M., Kollmannsberger, S., Schillinger, D. & Düster, A. 2012 Geometric modeling, isogeometric analysis and the finite cell method. *Computer Methods in Applied Mechanics and Engineering* **249-252**, 104 – 115, higher Order Finite Element and Isogeometric Methods. 7

[29] Schillinger, Dominik, Cai, Quanji, Mundani, Ralf-Peter & Rank, Ernst 2013 A review of the finite cell method for nonlinear structural analysis of complex cad and image-based geometric models. In *Advanced Computing* (ed. Michael Bader, Hans-Joachim Bungartz & Tobias Weinzierl), pp. 1–23. Berlin, Heidelberg: Springer Berlin Heidelberg. 6

[30] Schillinger, Dominik, Kollmannsberger, Stefan, Mundani, Ralf-Peter & Rank, Ernst 2010 The finite cell method for geometrically nonlinear problems of solid mechanics. *IOP Conference Series: Materials Science and Engineering* **10**, 012170. 6

[31] Shewchuk, Jonathan Richard 1996 Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering* (ed. Ming C. Lin & Dinesh Manocha), *Lecture Notes in Computer Science*, vol. 1148, pp. 203–222. Springer-Verlag, from the First ACM Workshop on Applied Computational Geometry. 52

[32] Smith, Simon J. 2006 Lebesgue constants in polynomial interpolation. *Annales Mathematicae et Informaticae* **33**, 109 – 123. 18

[33] Sommariva, A. & Vianello, M. 2007 Product gauss cubature over polygons based on green's integration formula. *BIT Numerical Mathematics* **47** (2), 441–453. 18

[34] Sommariva, Alvise & Vianello, Marco 2009 Computing approximate Fekete points by {QR} factorizations of vandermonde matrices. *Computers & Mathematics with Applications* **57** (8), 1324 – 1336. 18, 19, 21, 23, 24

[35] Taylor, Mark A. & Wingate, B.A. 2000 A generalized diagonal mass matrix spectral element method for non-quadrilateral elements. *Applied Numerical Mathematics* **33** (1), 259 – 265. 14

[36] Tews, R. & Rachowicz, W. 2009 Application of an automatic hp adaptive finite element method for thin-walled structures. *Computer Methods in Applied Mechanics and Engineering* **198** (21), 1967 – 1984, advances in Simulation-Based Engineering Sciences – Honoring J. Tinsley Oden. 10

[37] Trefethen, L.N. 2013 *Approximation Theory and Approximation Practice*. Siam. 17

[38] Vos, Peter E.J., Sherwin, Spencer J. & Kirby, Robert M. 2010 From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations. *Journal of Computational Physics* **229** (13), 5161 – 5181. 1, 15, 86

[39] Xie, Zhong, Sevilla, Ruben, Hassan, O. & Morgan, Kenneth 2014 The generation of arbitrary order curved meshes for 3d finite element analysis. *Computational Mechanics* **51**, 361–374. 4

APPENDIX A

ADDITIONAL CODE

Some additional examples of the source code are included here.

Initializing the element

The process for the element initialization subroutine `init_elem` is described here.

First, the subroutine variables are called.

```
1  subroutine init_elem(elem, ielem, grd, neqs)
2    implicit none
3    type(element), intent(inout) :: elem
4    integer, intent(in) :: ielem ! element number
5    type(grid), intent(inout) :: grd
6    integer, intent(in) :: neqs
7
8    ! local vars
9    integer :: i, p, rule, order_num, npe, degree
10   real*8 :: x0, y0
11   real*8, dimension(:,:), allocatable :: xy
12   type(fekete)  :: tfekete
13   integer :: ii, jj, pt1, pt2, last_pt
14   integer :: int1, int2
15   integer, dimension(:), allocatable :: pts
16
17   p = grd%p(ielem)
18   npe = grd%npe(ielem)
19   elem%number = ielem
20   elem%npe = npe
21   elem%elname = grd%elname(ielem)
22   elem%p = grd%p(ielem)
23   elem%eltype = grd%eltype(ielem)
24   elem%npedg = elem%p + 1 ! init p=0,1
```

```
25      elem%neqs = neqs
```

Next, we define the quadrature rule.

```
27      ! compute the "order_num" or number of Gauss points
28      rule = int(2.0d0 * p) ! infact it should be "p" for linear
29                   ! Laplace equation for constant shape elements
30                   ! but since we've got rational function
31                   ! for curvilinear elements, then we put
32                   ! it 2*p for safety.
```

Then, the element x, y coordinates are defined in an elemental context and quadrature is performed. Note the provisioning in the code at this point to accept quadrilateral elements.

```
34      if( allocated(elem%x) ) deallocate(elem%x)
35      allocate( elem%x(2, npe) )
36      elem%x(1, :) = grd%x( grd%icon(ielem, 1:grd%npe(ielem)) )
37      elem%x(2, :) = grd%y( grd%icon(ielem, 1:grd%npe(ielem)) )
38
39      select case (grd%elname(ielem))
40
41      case (GEN_TRIANGLE)
42         ! check to see the order of exactness is available
43         ! in the tables for the given rule
44         call dunavant_degree ( rule, degree )
45
46         ! compute the number of required Gauss points
47         call dunavant_order_num( rule, order_num )
48         elem%ngauss = order_num
49
50         ! allocate space for that
```

```
51        allocate( xy(2,order_num))
52        allocate(elem%r(order_num), elem%s(order_num), elem%W(order_num))
53
54        ! compute the absicca and weights for that rule
55        call dunavant_rule( rule, order_num, xy, elem%W )
56        elem%r = xy(1,:)
57        elem%s = xy(2,:)
58        deallocate(xy)
59
60        elem%coeff = 0.5d0
61        elem%nedgs = 3
62
63     case (GEN_QUADRI)
64
65        call grd%tfekete_table%lookup(d = rule, name = 'quadri', fekete_out
       = tfekete &
66             , spacing = 'equal_space', s = 3, echo = .true.)
67
68        ! call tfekete%init(d = rule, name = 'quadri' &
69        !      , spacing = 'equal_space', s = 3, echo = .true.)
70        elem%ngauss = size(tfekete%w)
71
72        ! allocate space for that
73        allocate(elem%r(elem%ngauss), elem%s(elem%ngauss), elem%W(elem%
      ngauss))
74
75        elem%r = tfekete%fin_coords(1, :)
76        elem%s = tfekete%fin_coords(2, :)
77        elem%W = tfekete%w
78
79        elem%coeff = 1.0d0
```

```
80        elem%nedgs = 4
81
82        ! ! deallocate stuff in tfekete object
83        ! call tfekete%clean()
84
85     case default
86        print *, 'unknown elname in obtaining quadrature rules! stop'
87        stop
88
89     end select
```

Next, the basis functions and their derivatives are initialized, then evaluated.

```
91     ! initializing the basis functions and their derivatives
92     call elem%tbasis%init(grd%maselem(ielem)%xi, grd%maselem(ielem)%eta &
93          , grd%elname(ielem))
94
95     allocate(elem%psi(npe,elem%ngauss) &
96            , elem%d_psi_d_xi(npe,elem%ngauss) &
97            , elem%d_psi_d_eta(npe,elem%ngauss) )
98
99     ! evaluating the basis function and their derivatives at
100    ! Gauss - Legendre (quadrature) points and then storing
101    do i = 1, elem%ngauss
102       x0 = elem%r(i)
103       y0 = elem%s(i)
104       call elem%tbasis%eval(x0, y0, 0,  elem%psi(:,i)        )
105       call elem%tbasis%eval(x0, y0, 1,  elem%d_psi_d_xi(:,i) )
106       call elem%tbasis%eval(x0, y0, 2,  elem%d_psi_d_eta(:,i))
107    end do
```

After that, the Jacobian of transformation and its inverse and the value of Jacobian at each Gauss-Legendre point is initialized, then evaluated. Further detail about `comp_Jstar` is shown later in this appendix.

```
109     ! allocate Jacobian of transformation
110     allocate(elem%jac(2,2, elem%ngauss), elem%Jstar(2,2, elem%ngauss) )
111     allocate(elem%JJ(elem%ngauss) )
112
113     ! compute the Jacobian of transformation matrix , its inverse and
114     ! the value of Jacobian at each Gauss-Legendre point
115     do i = 1, elem%ngauss
116        call comp_Jstar(grd, elem, ielem, i, elem%jac(:,:,i) &
117             , elem%Jstar(:,:,i), elem%JJ(i) )
118     end do
```

Then, the elemental stiffness matrices and equation right-hand-side is initialized.

```
120     ! initialize the stiffness matrix
121     allocate(elem%K(elem%neqs,elem%neqs, npe, npe))
122     allocate(elem%M(elem%neqs,elem%neqs, npe, npe))
123     elem%K = 0.0d0
124     elem%M = 0.0d0
125
126     ! allocating and initializing rhs
127     allocate(elem%Q(elem%neqs,npe))
128     allocate(elem%f(elem%neqs,npe))
129     elem%Q = 0.0d0
130     elem%f = 0.0d0
```

Next, we initialize the edges, by determining the start and end point of the segments and generate neighbor information for the connectivity matrix.

```
132     ! allocate and init edges
133     if ( allocated(elem%edgs) ) deallocate(elem%edgs)
134     allocate(elem%edgs(elem%nedgs))
135
136     ! allocate/init neighbors (initially one neighbor!)
137     !
138     ! determine the start and the end of the segments shared with
      neighbors
139     do ii = 1, elem%nedgs
140        allocate(elem%edgs(ii)%neighs(1))
141        pt1 = ii
142        pt2 = ii + 1
143        if ( ii .eq. elem%nedgs ) pt2 = 1
144        elem%edgs(ii)%neighs(1)%xs = elem%x(:, pt1)
145        elem%edgs(ii)%neighs(1)%xe = elem%x(:, pt2)
146     end do
147
148     ! adding neighbors
149     select case (grd%elname(ielem))
150     case ( GEN_QUADRI)
151
152        elem%edgs(2)%neighs(1)%elnum = grd%e2e(ielem, 1)
153        elem%edgs(3)%neighs(1)%elnum = grd%e2e(ielem, 2)
154        elem%edgs(4)%neighs(1)%elnum = grd%e2e(ielem, 3)
155        elem%edgs(1)%neighs(1)%elnum = grd%e2e(ielem, 4)
156
157     case ( GEN_TRIANGLE)
158
159        elem%edgs(2)%neighs(1)%elnum = grd%e2e(ielem, 1)
160        elem%edgs(3)%neighs(1)%elnum = grd%e2e(ielem, 2)
161        elem%edgs(1)%neighs(1)%elnum = grd%e2e(ielem, 3)
```

```
162
163     end select
164
165     ! adding points per edge
166     if ( elem%p > 0 ) then ! we have points on the edges
167        elem%npedg = size(grd%el2edg(ielem)%edg1) + 2
168        ! end points are included!
169
170        last_pt = elem%nedgs + 1
171        allocate(pts(elem%npedg))
172
173        do ii = 1, size(elem%edgs)
174           pt1 = ii
175           pt2 = ii + 1
176           if ( ii .eq. size(elem%edgs) ) pt2 = 1
177
178           if ( elem%p >= 2 ) then !higher order elements
179              int1 = last_pt
180              int2 = last_pt + elem%npedg - 3
181              pts = (/ pt1, (/ (jj, jj = int1, int2) /) , pt2 /)
182              last_pt = int2 + 1
183           else
184              pts = (/ pt1, pt2 /)
185           end if
186
187           allocate(elem%edgs(ii)%pts(elem%npedg))
188           elem%edgs(ii)%pts = pts
189        end do
190
191        deallocate(pts)
```

```
192     end if
```

Last, the boundary condition tag for each edge is defined.

```
194     ! specify the tag of the edges
195     do ii = 1, size(elem%edgs)
196        elem%edgs(ii)%tag = grd%local_edg_bc(elem%number, ii)
197     end do
198   end subroutine init_elem
```

Computing the Jacobian of transformation

The process for the Jacobian of transformation calculation subroutine `comp_Jstar` is described here. This subroutine computes the Jacobian of transformation, J or in our code `jac`, and the inverse of the Jacobian of transformation, J^* or in our code `Jstar`, at the Gauss points r_k, s_k within the element `elem` for element number `ielem` in the grid called `grd`. The determinant of Jacobian of transformation is returned in `JJ`.

First, the subroutine variables are called and zeroed.

```
1   subroutine comp_Jstar(grd, elem, ielem, k, jac, Jstar, JJ)
2     implicit none
3     type(grid), intent(in) :: grd
4     type(element), intent(in) :: elem
5     integer, intent(in) :: ielem, k
6     real*8, dimension(:,:), intent(out) :: jac, Jstar
7     real*8, intent(out) :: JJ
8
9     ! local vars
10    integer :: i
```

```
11      real*8 :: xi, yi
12      real*8, dimension(2) :: der
13      real*8 :: dx_dr, dx_ds, dy_dr, dy_ds
14      integer :: npe
15
16      ! hard reset
17      jac = 0.0d0; Jstar = 0.0d0; JJ = 0.0d0
18      xi = 0.0d0; yi = 0.0d0; der = 0.0d0
19      dx_dr = 0.0d0; dx_ds = 0.0d0; dy_dr= 0.0d0; dy_ds = 0.0d0
20      npe = grd%npe(ielem)
21      ! print *, '============================================='
22      ! print *, '============================================='
23      ! print *, 'computing jacobian for elem #', ielem
24      ! print *, '============================================='
25      ! print *, '============================================='
```

Then, components of the computation are calculated. For each point in the element, the derivatives `d_psi_d_xi` and `d_psi_d_eta` are used with the nodal coordinate to compute `dx_dr`, `dx_ds`, `dy_dr`, and `dy_ds`.

```
27      ! compute the components of jac
28      do i = 1, npe ! assuming iso-geometric expansion for x, y
29         ! print *, 'at point #', grd%icon(ielem,i)
30         der(1) =  elem%d_psi_d_xi(i,k)
31         der(2) = elem%d_psi_d_eta(i,k)
32         ! print *, 'der = [', der, ']'
33         xi = grd%x(grd%icon(ielem,i))
34         yi = grd%y(grd%icon(ielem,i))
35         ! print *, 'xi = ', xi, 'yi = ', yi
36         dx_dr = dx_dr + xi * der(1)
37         dx_ds = dx_ds + xi * der(2)
```

```
38          dy_dr = dy_dr + yi * der(1)
39          dy_ds = dy_ds + yi * der(2)
40          ! print *, ' total dx_dr = ', dx_dr
41          ! print *, ' total dx_ds = ', dx_ds
42          ! print *, ' total dy_dr = ', dy_dr
43          ! print *, ' total dy_ds = ', dy_ds
44
45      end do
```

Last, the Jacobian of transformation, its determinant and the inverse of the Jacobian J^* is computed and validated.

```
47      ! compute jac
48      jac(1,1) = dx_dr; jac(1,2) = dy_dr
49      jac(2,1) = dx_ds; jac(2,2) = dy_ds
50
51      ! comp JJ, i.e. det of jac
52      JJ = dx_dr * dy_ds - dx_ds * dy_dr
53
54      ! check it, should be valid grid!!!
55      if (JJ <= 0.0d0) then
56         print *, 'error : negative or zero Jacobian at element (',ielem,')'
57         ! print *, ' JJ = ', JJ
58         stop
59      ! else
60      !    print *, 'OKKKK : positive Jacobian at element (',ielem,')'
61      !    print *, ' JJ = ', JJ
62      end if
63
64      ! comp inverse of jac and store it in Jstar
65      Jstar(1,1) =  1.0d0 / JJ * jac(2,2)
```

```
66      Jstar(2,2) =  1.0d0 / JJ * jac(1,1)
67      Jstar(1,2) = -1.0d0 / JJ * jac(1,2)
68      Jstar(2,1) = -1.0d0 / JJ * jac(2,1)
69    end subroutine comp_Jstar
```

VITA

Kyle Shannon is a life-long Chattanooga-area resident. He attended Soddy Daisy High School in his pre-collegiate education, in which he graduated in the class of 2011. Kyle has always had an interest in mechanical engineering due to his life-long fascination of trains, and in particular steam locomotives. Due to this, Kyle enrolled in the College of Engineering and Computer Science at the University of Tennessee at Chattanooga as a Mechanical Engineering major in August 2011. He graduated in 2015 with a Bachelor of Science in Mechanical Engineering degree in May 2015. Kyle has also had a long standing goal of achieving at least a Master's degree in engineering, so in August of 2016, he enrolled again at UTC to begin his Master's studies in Automotive Systems Engineering, in which he specialized in solid mechanics and finite element analysis.

Professionally, Kyle began working at the Tennessee Valley Railroad Museum in Chattanooga, TN, in 2009 as a volunteer and joining the staff in 2011. He maintains a locomotive engineer's certification, conductor's certification, and is qualified to operate both steam and diesel locomotives. In the past, he worked in the repair shop as well as railroad operations, helping with both the maintenance and restoration of historic railway equipment, including the rebuilds of 1904-built Southern Railway 630 and 1911-built Southern Railway 4501. In January 2014, Kyle began his first internship with Volkswagen Chattanooga, interning with the Assembly department until May 2014. He returned to Volkswagen in January 2015 as an intern with the Vehicle Analysis Center, where he has been ever since, being hired as a full time employee in January 2016 with the role of Vehicle Analysis Specialist.