Overcoming Element Quality Dependence of Finite Element Methods

A Dissertation Presented

by

Rebecca Conley

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

May 2016

Stony Brook University

The Graduate School

Rebecca Conley

We, the dissertation committe for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Overcoming Element Quality Dependence of Finite Element Methods

by

Rebecca Conley

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2016

The finite element methods (FEM) are important techniques in engineering for solving partial differential equations, especially on complex geometries. One limitation of the classical FEM is its heavy dependence on element shape quality for stability and good performance. We introduce the *Adaptive Extended Stencil Finite Element Method* (AES-FEM) as a means to overcome this dependence on element shape quality. Our method replaces the traditional basis functions with a set of *generalized Lagrange polynomial (GLP) basis functions*, which we construct using local weighted least-squares approximations. The method preserves the theoretical framework of FEM, and allows imposing essential boundary conditions and integrating the stiffness matrix in the same way as the classical FEM.

In this dissertation, we describe the formulation and implementation of AES-FEM with quadratic basis functions, and analyze its consistency and stability. Next, we present an extension to high-order AES-FEM, including analysis and implementation details. High-order AES-FEM uses meshes with linear elements, thus avoiding the challenges of isoparametric elements. We present numerical experiments in both 2D and 3D for the Poisson equation and a time-independent convection-diffusion equation, including results on curved boundaries. We demonstrate high-order convergence up to sixth order of accuracy. Since AES-FEM results in a non-symmetric stiffness matrix, we compare the timing results of several combinations of linear solvers and preconditioners. The numerical results demonstrate that high-order AES-FEM is more accurate than high-order FEM, is also more efficient than FEM in terms of error versus runtime on finer meshes, and enables much better stability and faster convergence of iterative solvers than FEM over poor-quality meshes.

Acknowledgements

First, I would like to thank Dr. Jiao for all his guidance, help, and support for the last 5 years. His organization, patience, and enthusiasm make him an excellent mentor. I have learned so much from him.

Next, I would like to thank all the present and past members of my research group, including Hongxu Liu, Cao Lu, Navamita Ray, Xuebin Wang, Oliver Yang, Xinglin Zhao, and especially, Tristan Delaney and Aditi Ghai for their help preparing this thesis. Thank you for all the help and encouragement. It has been great to work with all of you.

I would like to thank my friends and family, both immediate and extended: Mom, Dad, Michelle, Kevin, Allyson, Shawn, Dan, Phyl, Uncle Bob, Jessica and too many others to name. Special thanks to my wonderful and supportive husband, Demos.

Lastly, I would like to dedicate this thesis in memory of my Nana, Ruth Charron.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Finite element methods (FEM) are one of the most important tools for solving partial differential equations on complex geometries. They account for an overwhelming majority of the commercial and research code for modeling and simulations, and there is a vast amount of theoretical work to provide a rigorous foundation; see [18, 93].

Despite their apparent success in many applications, classical finite element methods have a very fundamental limitation: *they are dependent on element shape quality.* This is especially true for elliptic and parabolic problems, for which the resulting linear system is often ill-conditioned if a mesh contains a few "bad" elements. This can lead to very slow convergence of iterative solvers and sometimes even a loss of accuracy. This element-quality dependency has impeded productivity and efficiency. A recent survey conducted by NASA found that the mesh generation remains a significant bottleneck for large scale complex simulations in the field of computational fluid dynamics [81]. Because of this, researchers and users of FEM often spend a tremendous amount of time and computing power to generate and maintain meshes, trying to fix that one last bad element. This has spurred much successful research in meshing, including Delaunay triangulation [79, 80], advancing front [54], and octree-based methods [77].

FEM originated as a second order accurate method, but for several decades, researchers have been exploring high-order variants. In this dissertation, we use the term "high-order" to refer to any method of third or greater order accuracy. Some high-order methods include isoparametric FEM [27], hp-FEM [25], discontinuous Galerkin method [21], spectral element method [16], and isogeometric analysis [42]. The element quality requirements for high-order methods are even more stringent. Despite the fact that these methods can reach exponential convergence under appropriate conditions, high-order methods have remained largely confined to academic study and has yet to make much of an impact in industry [88]. This is due to many reasons, not least of which is that high-order schemes are generally less robust [90] and generating good quality meshes for high-order methods is still not fully resolved [69].

The FEM community has long considered this dependency on element quality as a critical issue, and the community has been actively searching for alternative methods to mitigate the issue for decades. Examples of such alternative methods include the diffuse element or element-free Galerkin methods [9, 64], least-squares FEM [12], generalized or meshless finite different methods [10, 44, 52, 60, 68], generalized or extended FEM [8, 33], and partition-of-unity FEM [59]. To reduce the dependency on mesh quality, these methods avoid the use of the piecewise-polynomial Lagrange basis functions found in the classical FEM. However, they also lose some advantages of the classical FEM. In particular, for the generalized finite different methods, the strong form instead of the weak form of the PDE must be used. The partition-of-unity FEM and other similar generalizations often incur complexities in terms of imposing essential boundary conditions and/or integrating the stiffness matrix [59]. Therefore, it remains an open problem to develop a numerical method that overcomes the element-quality dependence, while preserving the theoretical framework of FEM, without complicating the imposition of boundary conditions and numerical integration.

This dissertation presents a new method, called the *Adaptive Extended Stencil Finite Element Method* (*AES-FEM*) (pronounced as ace-F-E-M), to address this open problem [22, 23]. Similar to some of the aforementioned methods, the AES-FEM replaces the piecewise-polynomial Lagrange basis functions in the classical FEM with alternative basis functions. Different from those methods, our basis functions are partition-of-unity polynomial basis functions, constructed based on local weighted least squares approximations, over an adaptively selected stencil to ensure stability. We refer to these basis functions as *generalized Lagrange polynomial* (*GLP*) basis functions. Another difference of AES-FEM from most other generalizations of FEM is that AES-FEM preserves the traditional finite element shape functions as the weight functions (a.k.a. test functions) in the weak form, to preserve the compact support of integration and the weak form after integration by parts. This combination of the basis and weight functions enables AES-FEM to overcome the element-quality dependence, while preserving the theoretical framework of FEM, without any complication in imposing essential boundary conditions or integrating the stiffness matrix. In addition, the resulting stiffness matrix of AES-FEM has virtually the same sparsity pattern as that of the classical FEM, while allowing the use of higher-degree polynomials and hence significantly improved accuracy. AES-FEM is most applicable to elliptical and parabolic problems. For solving hyperbolic problems, the weighted least squares based essentially non-oscillating (WLS-ENO) scheme has been proposed for finite volume methods in [53], which also uses local weighted least squares polynomial approximations like AES-FEM.

As a general method, AES-FEM allows polynomial basis functions of arbitrary degrees. The degree of the basis functions controls the rate of convergence. AES-FEM is based on the weighted-residual formulation of FEM instead of the Galerkin formulation, and hence the resulting system is non-symmetric, which is more expensive to solve than a symmetric system. In addition, it is more expensive to construct the basis functions of AES-FEM than to use the standard basis functions in FEM. Therefore, AES-FEM is conceivably less efficient than FEM for a given mesh. However, as we will demonstrate in our experimental results, AES-FEM is significantly more accurate than FEM on a given mesh due to its use of higher-degree basis functions, and it is also more efficient than FEM in terms of error versus runtime. Most impotently, AES-FEM enables better stability and faster convergence of iterative solvers than FEM over poor-quality meshes.

The reminder of the dissertation is organized as follows. In Chapter 2, we present some background information and recent developments of related methods. In Chapter 3, we formulate FEM from a weighted residual perspective, describe the construction of generalized Lagrange polynomial basis functions based on weighted least squares approximations, and then introduce quadratic AES-FEM. Chapter 4 contains an analysis, including consistency, accuracy and stability, of AES-FEM and a generalization to high-order. In Chapter 5, we discuss some implementation details including the utilized data structure and the applicable algorithms. In Chapter 6, we present the results of some numerical experiments with our approach.

Finally, Chapter 7 concludes the paper with a discussion.

The main contributions of this thesis are as follows. First, we extend the concept of Lagrange basis functions, which are interpolary by nature, to *generalized Lagrange polynomial basis functions*, which apply to least-squares approximation. The generalized Lagrange polynomial basis functions retain two very important properties of the Lagrange basis functions, that is, *function value as coefficient* and *partition of unity*. Second, we introduce the adaptive extended stencil finite element method (AES-FEM), which is insensitive to element-quality because it utilizes the generalized Lagrange polynomial basis functions. Third, we generalize AES-FEM to high-order accuracy, for which AES-FEM uses linear meshes even on domains with curved boundaries. We analyze the consistency, accuracy, and stability of AES-FEM and explain the use of linear elements with curved boundaries. Last, we apply AES-FEM to the Poisson equation and the convection-diffusion equation in 2D and 3D, and we present convergence of up to sixth order. We test the element-dependence of AES-FEM and FEM, and we compare the efficiency of the two methods.

In this work, we use boldface, uppercase letters for matrices, e.g., $\boldsymbol{A}$. We use boldface, lowercase letters for vectors, e.g., $\boldsymbol{u}$. Lowercase, non-bold letters are used for scalars, e.g., x. We refer to entry in the ith row and jth column of matrix $\boldsymbol{A}$ as a_{ij}. The ith row of matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A}_{(i,:)}$ and the jth column is denoted $\boldsymbol{A}_{(:,j)}$. We denote the leading principle sub-matrix of size $k \times k$ of matrix $\boldsymbol{A}$ by $\boldsymbol{A}_{1:k,1:k}$. We use ψ for weight functions and ϕ for basis functions.

Chapter 2

Background and Related Work

Researchers have long realized the demands on mesh quality made by the use of Lagrange basis functions in traditional FEM. For several decades, they have been actively searching for alternative basis functions to reduce the dependence on mesh quality. Alternatives include the diffuse element method [64] and element free Galerkin method [9]; the generalized finite element method and extended finite element method [8, 33]; the discontinuous Galerkin method [2, 21]; the spectral element method [66]; isogeometric analysis [42] and other methods using NURBS [76]; and generalized finite difference or meshless methods [44].

Many of these above mentioned methods have the ability to achieve higher order accuracy. Other high-order accurate methods include FEM with isoparametric elements [27, 43] and p- and hp-FEM [4]. High-order methods hold the promise of increased accuracy at a comparable computational cost to lower order method. Additionally, there are some problems (such as vortex-dominated flows, like helicopter blade vortex interaction) for which lower order methods are not well-suited [90].

In this chapter we discuss the aforementioned methods and other generalizations of the finite element method, include least-squares finite element method. We focus on the basis functions used in each method, its relation to higher-order accuracy, and the differences with AES-FEM.

2.1 High-Order Finite Element Order

2.1.1 Isoparametric Finite Element Method

Isoparametric finite element method uses the same, often high-order polynomial, shape functions to define both the geometry and the functional approximation [93]. This concept of elements with curved edges was popularized in the 1960s by Irons, Ergatoudis, and Zienkiewicz [27, 43] and was predicated on the earlier unpublished work by I. C. Taig [43, 93]. In 1973, Zlamal proposed a method of using curved elements only along curve boundaries and straight edge triangles on the interior [95]. In the same year, he provided some theoretical analysis of curved elements, including error bounded and an approximation theorem [94]. Isoparametric elements provided a answer to an open issue, that is, if the boundary of the domain are curved, a high-order approximation of the geometry is necessary for the accuracy of the solution to be high-order [7]. However, this solution is not without its difficulties.

In the decade following the first suggestion of the use of isoparametric elements, a series of a short articles and communications highlighted the poor performance of the serendipity-type of isoparametric elements when the elements were distorted [32, 37, 83]. In 1993, Lee and Bathe compiled these results and performed similar experiments for Lagrangian-type elements [49]. They found that while Lagrangian-type elements with angular-distortions do not suffer the same poor performance as serendipity-type elements with angular-distortions, both types of elements exhibit poor performance for curved-edge distortions. The reason for this poor performance is that the use of isoparametric elements demands a high quality mesh. If an element is too distorted then the one-to-one mapping between the local coordinates and global coordinates no longer exists, and the method will break down [93]. Another way of stating this requirement is that the Jacobian determinate must be positive everywhere within the reference element for the physical element to be geometrically valid [46].

The meshing requirements on isoparametric elements are stringent. In 1972, Ciarlet and Raviart proved that the Jacobian of the mapping and its first p derivatives must be bounded for isoparametric, curved elements [19]. Additionally, in 1986, Lenior highlighted the contradictory requirements for curved elements: the difference between the curved elements and the straight-edged elements with the same vertices must vanish fast enough, and at the same time, the approximated boundary must be close enough to the physical boundary of the domain [50].

While progress has been made to address the meshing requirements, efficient meshing of high-order, high-quality curvilinear elements is still area of active research especially in 3D. One method for refining a curved mesh is to project the new vertices onto the correct geometry, but care must be taken that the refined mesh is free of invalid elements [51]. A low-quality, but valid element, can result in an ill-conditioned Jacobian which degrades the solution and introduces approximation error [34]. Mesh optimization requires mesh-quality measures, and in 2014 Gargallo-Peiró et al. extended the the set of Jacobian-based measure for linear elements to isoparametric elements in 3D [34]. In 2015, Moxey et al. propose a technique for generating high-order boundary layer meshes by refining a valid coarse meshing [63].

Because AES-FEM uses generalized Lagrange polynomial basis functions, AES-FEM is independent of mesh quality. Additionally, AES-FEM uses linear meshes and thus does not suffer from the issues associated with high-order meshing on curved boundaries.

2.1.2 h-FEM, p-FEM, and hp-FEM

In original implementation of the finite element method, the polynomial degree of the elements is some low constant number, such as $p = 1$ or $p = 2$. To achieve convergence the diameter of the elements, denoted h, is reduced. This version of the finite element method is referred to as the h-version or h-FEM. Another option is to fix the number of elements and to increase the polynomial degree of the elements. This is referred to as the p-version or p-FEM. Finally, when a combination of the two versions is used the resulting method is referred to as the hp-version or hp-FEM [75].

Until the 1978 paper of Szabó and Mehta, it had been thought that the rate of convergence of elasticity problems with stress singularities was independent of the element polynomial p [84]. In that paper, it was demonstrated that p-FEM could converge faster than h-FEM. The theoretical foundations for p-FEM were established in the 1981 paper by Babuška, Szabó, and Katz [5], where it was proved that if there is a singularity at a vertex, the p-version converges twice as fast as the h-version. In 1985, Gui and Babuška published a detailed analysis of the 1-

dimensional p-version [39]. In the ensuing decade, Babuška and his colleagues contributed greatly to the field of p- and hp-FEM, and in 1994, Babuška and Suri collected much of the work into a survey paper covering many topics, include basic theory and a benchmark comparison between the three versions in all three dimensions [4]. By this point, it was clear that one of the greatest challenges of the methods was designing an adaptive mesh for the hp-version.

Starting in the late 1980s, researchers began to focus on developing and implementing optimal hp-adaptivity strategies. Demkowicz and his colleagues proposed a refinement strategy that calculates a reference solution and is based on edge refinements [25]. Šolín's proposed refinement strategy also calculates on a reference solution but is based on element refinements [82]. Ainsworth and Senior proposed a strategy that uses three error estimates to estimate the smoothness of the solution [1]. A recent survey paper by Mitchell compares thirteen strategies on twenty-one benchmark problems and found that both Demkowicz's and Šolín's strategies perform the best in terms of convergence but that they are every expensive [62]. Picking the best strategy is problem dependent, and there is a trade-off concerning accuracy and expense.

Because of the use of high-order basis functions, p- and hp-FEM have the same challenges with meshing as discussed in the previous subsection. Additionally, since in p-FEM the elements are typically larger than in h-FEM, it is absolutely necessary to represent the domain properly and special care must be taken with integration [56]. In 2010, Luo et al. presented a method for constructing nearly optimal meshes on 3-dimensional, curved domains with singularities and thin sections for the p-version. Their method starts with a CAD model and uses medial surface points to identify thin sections [57].

In addition to the area of meshing, another major challenge of *hp*-FEM is guaranteeing the inter-element continuity of the shape functions. This can be addressed by using constrained approximation [24, 82], although this is algorithmic complex especially in 3D. Another strategy is multi-level hp-adaptivity, which uses coarse mesh everywhere and then overlays finer meshes in areas of interest [92]. Thus the continuity of the shape functions is guaranteed by construction.

The use of linear meshes and generalized Lagrange polynomial basis functions in AES-FEM prevent the challenges associated with meshing from affecting AES-FEM.

2.2 Diffuse Element Method and Element Free Galerkin Method

Various alternatives of finite element methods have been proposed in the literature to mitigate the mesh-quality dependence. One of the examples is the *diffuse element method* (DEM) [64], proposed by Nayroles, Touzot, and Villon in 1992. Similar to AES-FEM, DEM constructs local approximations to an unknown function based on a local weighted least squares fitting at each node. However, unlike AES-FEM, the DEM is based on the Galerkin formulation, which requires the shape functions to have a compact support for efficiency. To this end, DEM relies on a weight function that vanishes at a certain distance from a node, in a manner similar to the moving least squares fittings [48]. The accuracy and efficiency of numerical integration in DEM depends on the particular choice of the weight function. In contrast, based on a weighted-residual formulation, AES-FEM enforces the compact support of the weak form with the weight functions, so the shape function does not need to have a compact support.

Another approach, which is closely related to DEM, is the *element-free Galerkin method* (EFGM) [9], proposed by Belytschko, Lu, and Gu in 1994. As a Galerkin method, EFGM also requires a compact support of its shape functions, which serve as both the trial functions and weight functions. Similar to DEM, EFGM constructs the shape functions based on moving least squares, for which the weight function plays a critical role in terms of accuracy and efficiency. However, depending on the weight functions, the shape functions in EFGM may not be polynomials. It requires special quadrature rules with more quadrature points than those of standard FEM [9], and it also requires evaluating a moving least squares fitting at each quadrature point. In addition, EFGM requires the use of Lagrange multipliers for essential boundary conditions. In contrast, AES-FEM can utilize the same treatments of boundary conditions and numerical integration as the standard FEM.

2.3 Generalized Finite Element Methods

Generalized Finite Element Methods (GFEM), Extended Finite Element Methods (XFEM), and Partition of Unity Methods (PUM) are all methods which seek to extend standard FEM through the addition of enrichment functions, in order to better model discontinuous or singular phenomena such as the cracks, dislocations, grain boundaries, phase boundaries, material interfaces, shocks, and boundary layers. The enrichment functions are special functions which are designed to capture the behavior of the solution locally (such as jumps or singularities). They may be obtained from asymptotic solutions which are not exact solutions. The main advantage of these methods is that the mesh may be independent of the physical entities causing the discontinuity. The mesh does not need to conform to a discontinuity or be refined near a singularity, and there is no longer a need for continuous remeshing as the problem evolves. There has been much research about these methods and there are many excellence survey papers, including [8] and [33]. It should be noted that while GFEM, XFEM, and PUM developed from different roots, their current implementations are virtually indistinguishable, and we will refer to them collectively as GFEM.

While GFEM may offer an efficient method to model discontinuities, the addition of the enrichment functions often leads to some complications. First, the resulting basis functions may be linearly dependent (or nearly so) and this leads to ill-conditioning. This becomes even more of a problem when multiple enrichments are used. Second, the enforcement of essential boundary conditions is no longer a trivial matter. One must use Lagrange multipliers, penalty method, or another method to impose them. Third, the enrichment functions often contain jumps, kinks or high gradients and thus the standard Gauss quadrature rules cannot be used. Depending on the properties of the enrichment functions, strategies for accurate quadrature include element decomposition for discontinuous enrichment functions, special quadrature rules for singular enrichment functions, and the use of a large number of Gauss points for enrichment functions with high gradients. Finally, there are difficulties with blending elements, which are elements where some but not all of the nodes are enriched. Partition of unity no longer holds in these elements, which can negatively effect convergence. There are many strategies for dealing with blending elements, with differing applicability and efficacy.

2.4 Other Variants and Generalizations of FEM

Starting in the 1970s there have been many new methods which have build off of ideas central to the finite element method and the weighted residual formulation. The aim of all of these methods is to extend the positive aspects of FEM; some to problems whose behavior is not well captured by FEM like shocks or convection dominated problems, and others to decrease the dependency on mesh quality.

2.4.1 Discontinuous Galerkin Method & Internal Penalty Method

In the 1970s, two Galerkin methods based on piecewise continuous polynomial function spaces were developed simultaneously and independently [2, 21]. Interior Penalty Method was developed for elliptic and parabolic problems; it grew from the observation that if Dirichlet boundary conditions could be imposed weakly then so could inter-element continuity. Discontinuous Galerkin (DG) Method was originally conceived for hyperbolic problems, specifically for neutron transport [71]. DG was then extended to elliptic and parabolic problems; it is especially useful for problems with a dominate convective part and a non-negligible diffusive part. Later it was realized that these two methods were the same and now they have been united under the name Discontinuous Galerkin Methods [2]. The trial and test functions are piecewise continuous polynomials and continuity between elements is not enforced. Most Discontinuous Galerkin methods use a numerical flux at the element boundaries; the way this flux is defined will change the method being used.

DG methods are able to achieve high accuracy and can be used on complex geometries [21]. Bassi and Rebay applied high order accurate DG to the Navier-

Stokes and compressible Euler equations [6, 7]. More recent developments include compact DG [67], which have a more compact stencil than local DG methods, and hybridized DG [20], which have a relatively low number of degrees of freedom. Unlike DG, AES-FEM uses basis functions which have inter-element continuity.

2.4.2 Spectral Element Method

In 1984, Patera combined the ideas of the finite element method with those of the spectral method to invent the Spectral Element Method (SEM) [66]. The objective was to exploit the ease with which the former handled complex domains and capture the high accuracy of the latter. High-order, orthogonal functions, such as Chebyshev or Legendre polynomials, are used for the trial and test functions. Quadrature is performed using Legendre-Gauss–Lobatto points, which allows for the convenient enforcement of boundary conditions. Assuming the solution is sufficiently smooth, exponential (or spectral) convergence is achieved. While increasing the degree of the polynomial basis used can increase the accuracy exponentially, if too high of a degree is used the method becomes prohibitively expensive. Most implementations are based on quadrilaterals and hexahedra, which may not allow complex geometries to be captured well, but research is being performed into triangle and tetrahedral meshing for SEMs [15].

2.4.3 Least-Squares Finite Element Method

Since the 1990s, Bochev and Gunzburger have popularized a method known as Least-Squares Finite Element Method (LSFEM) [12]. This method solves a given PDE by minimizing a global error residual in the least squares sense. LSFEM was developed to extend the many positive aspects of the Rayleigh-Ritz formulation to problems that the Rayleigh-Ritz formulation does not apply to (such as non-self adjoint problems) and to problems for which the standard variational formulation results in a saddle-point problem (such as the Stokes problem in primitive variables). LSFEM has several positive properties, not the least of which is that the resulting system is always sparse, symmetric, and positive definite. There are still several challenges and open problems. In order to make LSFEM practical to implement, any higher order PDE must be decomposed into a system of first order PDEs. This not only increases the size of the system and the number of unknowns, but can also introduce ambiguity into the problem because there is often more than one way to perform the decomposition. Additionally, because of the global nature of the least squares problem, LSFEM does not have local conservation of mass. One may use Lagrange multipliers to recover mass conservation but the resulting linear system is indefinite.

2.4.4 Isogeometric Analysis

Isogeometric analysis (IGA) uses NURBS (Non-Uniform Rational B-Splines) or T-splines as basis functions instead of the standard FEM basis functions [42]. These methods can deliver high accuracy over very coarse meshes and can be advantageous for problems that can benefit from high-degree continuity, such as thin-shell modeling. Another related method is the NURBS-enhanced finite element method (NEFEM), which uses NURBS to represent the boundary of the computational domain and uses standard piecewise polynomials for solutions [76]. These methods bypass the process of generating meshes in the traditional sense, and they can deliver accurate solutions with relatively fewer degrees of freedom. However, they do not alleviate the dependency on mesh quality, because NURBS and T-splines tend to impose a stronger requirement on mesh quality than the traditional FEM.

2.5 Generalized Finite Difference & Weighted Least Squares

The study of generalized finite difference (GFD) methods or meshless finite difference methods (MFDM) has been an active research area for at least five decades. In 1960, Forsythe and Wasow [30] first suggested applying the finite different method to irregular grids. This idea was not actualized until 1972 when Jensen [44] used the Taylor series expansion to get 2nd order finite difference formulations on 6 point stars. The resulting matrices were often singular or ill-conditioned. In 1975 Perrone and Kao [68] attempted to tackle the problem of ill-conditioning. They proposed the inclusion of more nodes in the stars and the selection of nodes based on the 'eight-segment' criterion instead of only using distance as the criterion. In 1980, Liszka and Orkisz [52] further improved the ill-conditioning problem by selecting nodes based on the 'four-quadrant' criterion and then minimizing the norm of the matrix equation to solve for the coefficients.

In more recent years, Benito, Ureña, and Gavete have analyzed the influence of several key parameters (including the number of and selection criteria for the nodes in a star), presented an h-adaptivity procedure, and applied GFD to parabolic and hyperbolic PDEs and the advection-diffusion equation [10, 11, 35, 70]. Milewski demonstrated how to apply GFD to both the weak form and the strong form of a Poisson equation and released his Matlab code [60, 61]. Jiao et al. computed differential quantities, such as normals and curvatures, on surfaces using a generalized finite difference method based on weighted least squares [45, 89]. They explored methods to improve accuracy and stability, including node selection, a column scaling matrix, and the solution of the weighted least square problem using QR factorization with column pivoting.

Chapter 3

Formulation of AES-FEM

The main idea of AES-FEM is the use of an alternative set of basis functions. For the basis functions, we propose the use of a set of generalized Lagrange polynomial basis functions (GLPBF) computed using a weighted least squares formulation. We use the standard FEM (hat) basis functions for the weight functions (a.k.a. test functions). In this chapter, we describe the weighted residual formulation of FEM, define generalized Lagrange polynomial basis functions based on weighted least squares, explain the stable computation of the GLP basis functions, and then describe AES-FEM.

3.1 Weighted Residual Formulation of FEM

We will approach the formulation of the finite element method through the lens of the weighted residual method for solving PDEs. The main idea of the formulation is to consider the unknown solution as a linear combination of basis (trial) functions and then select the coefficients such that the residual is orthogonal to a set of weight (test) functions. Depending on the choice of the weight functions, one will derive different numerical methods, such as the collocation method, the least squares method, and the Galerkin method. Details about weighted residual and FEM can be found in [14, 18, 28]. In the following, we give a brief overview of weighted residuals for completeness.

Consider a linear differential operator $\mathcal{L}$ defined on a bounded, simply-connected domain Ω, with outward unit normal vector $\boldsymbol{n}$. Denote the boundary of Ω as $\Gamma = \Gamma_D \cup \Gamma_N$, where Γ_D and Γ_N are disjoint sets on which Dirichlet and Neumann boundary conditions are specified, respectively. We want to find a function U such that

$$\mathcal{L}U = \rho \tag{3.1.1}$$

subject to the boundary conditions

$$U = g \text{ on } \Gamma_D \quad \text{and} \quad \frac{\partial U}{\partial \boldsymbol{n}} = h \text{ on } \Gamma_N. \tag{3.1.2}$$

Eq. (3.1.1) is the strong form of the PDE. In the weighted residual formulation, we introduce a set of weight functions $\boldsymbol{\Psi} = \{\psi_1, \dots, \psi_n\}$, by requiring the residual $\mathcal{L}U - \rho$ to be orthogonal to ψ_i, i.e.,

$$\int_\Omega \psi_i \left(\mathcal{L}U - f\right)\, dV = 0. \tag{3.1.3}$$

Typically, $\boldsymbol{\Psi}$ forms a partition of unity, i.e., $\sum_{i=1}^{n} \psi_i = 1$, and in this case the weighted-residual formulation satisfies global conservation in the sense of

$$\int_\Omega \left(\mathcal{L}U - \rho\right)\, dV = 0. \tag{3.1.4}$$

If $\mathcal{L}$ involves higher than first order derivatives, then integration by parts is often used to decrease the order of the derivatives to obtain the weak form. To approximate u, let $\mathbf{\Phi} = \{\phi_1, \ldots, \phi_n\}$ be a set of basis functions, and define an approximation

$$U \approx \sum_{j=1}^{n} x_j \phi_j. \tag{3.1.5}$$

Substituting (3.1.5) into (3.1.3) and rearranging the equations, we then obtain

$$\sum_{j=1}^{n} x_j \int_\Omega \psi_i \left(\mathcal{L}\phi_j\right)\, dV = \int_\Omega \psi_i \rho \, dV, \tag{3.1.6}$$

leading to a system of equations in x_j. At this point for simplicity, let us consider the Poisson equation with Dirichlet boundary conditions, for which the weighted-residual formulation is given by

$$\int_\Omega \psi_i \nabla^2 U \, dV = \int_\Omega \psi_i \rho \, dV. \tag{3.1.7}$$

Substituting (3.1.5) into (3.1.7), we obtain

$$\sum_{j=1}^{n} x_j \int_\Omega \psi_i \nabla^2 \phi_j \, dV = \int_\Omega \psi_i \rho \, dV. \tag{3.1.8}$$

The finite element method uses integration by parts to reduce the order of derivatives required by (3.1.8). If ψ_i has weak derivatives and satisfies the condition $\psi_i|_{\Gamma_D} = 0$, then after integrating by parts and imposing the boundary conditions, we arrive at

$$-\sum_{j=1}^{n} x_j \int_\Omega \nabla\psi_i \cdot \nabla\phi_j \, dV = \int_\Omega \psi_i \rho \, dV. \tag{3.1.9}$$

Note that, (3.1.9) is often referred to as the weak form of the Poisson equation. Taking (3.1.9) over the n weight functions, we obtain the linear system

$$\boldsymbol{K}\boldsymbol{x} = \boldsymbol{g}, \tag{3.1.10}$$

where $\boldsymbol{K}$ is the stiffness matrix and $\boldsymbol{g}$ is the load vector, with

$$k_{ij} = -\int_\Omega \nabla\psi_i \cdot \nabla\phi_j \, dV \qquad \text{and} \qquad g_i = \int_\Omega \psi_i f \, dV. \tag{3.1.11}$$

If the weight functions are chosen to be the same as the basis functions, then we will arrive at the Galerkin method. In this paper, we introduce a new set of basis functions based on weighted least squares and we use the standard linear FEM "hat functions" as the weight functions.

3.2 Weighted Least Squares Approximations

In this subsection, we review numerical differentiation based on weighted least squares approximations, as described in [45, 89]. Similar to interpolation-based approximations, this method is based on Taylor series expansion. Let us take 2D as an example, and suppose $f(\boldsymbol{u})$ is a bivariate function with at least $d+1$ continuous derivatives in some neighborhood of $\boldsymbol{u}_0 = (0,0)$. Denote $c_{jk} = \frac{\partial^{j+k}}{\partial u^j \partial v^k} f(\boldsymbol{u}_0)$. Then for any $\boldsymbol{u}$ in the neighborhood, f may be approximated to the $(d+1)$st order accuracy about the origin $\boldsymbol{u}_0$ as

$$f(\boldsymbol{u}) = \underbrace{\sum_{p=0}^{d} \sum_{j,k\geq 0}^{j+k=p} c_{jk} \frac{u^j v^k}{j!k!}}_{\text{Taylor Polynomial}} + \underbrace{\mathcal{O}(\|\boldsymbol{u}\|^{d+1})}_{\text{remainder}}. \tag{3.2.1}$$

Analogous formulae exist in 1D and 3D. The derivatives of the Taylor polynomial are the same as those of f at $\boldsymbol{u}_0$ up to degree d. Therefore, once we have calculated the coefficients for the Taylor polynomial, finding the derivatives of f at $\boldsymbol{u}_0$ is trivial. We proceed with calculating the coefficients as follows.

For any point $\boldsymbol{u}_0$, we select a stencil of m nodes from the neighborhood around $\boldsymbol{u}_0$. Stencil selection is described further in Section 5.2. We do a local parameterization of the neighborhood so that $\boldsymbol{u}_0$ is located at the origin $(0, 0)$ and the coordinates of the other points are given relative to $\boldsymbol{u}_0$. Then substituting these points into (3.2.1), we obtain a set of approximate equations

$$\sum_{p=0}^{d} \sum_{j,k\geq 0}^{j+k=p} c_{jk} \frac{u_i^j v_i^k}{j!k!} \approx f_i, \tag{3.2.2}$$

where $f_i = f(\boldsymbol{u}_i)$ and the c_{jk} denote the unknowns, resulting in an $m \times n$ system. There are $n = (d+1)(d+2)/2$ unknowns in 2D and $n = (d+1)(d+2)(d+3)/6$ unknowns in 3D. Let $\boldsymbol{V}$ denote the generalized Vandermonde matrix, $\boldsymbol{c}$ denote the vector of unknowns (i.e., the c_{jk}) and $\boldsymbol{f}$ denote the vector of function values. Then we arrive at the rectangular system

$$\boldsymbol{V}\boldsymbol{c} \approx \boldsymbol{f}. \tag{3.2.3}$$

Let us now introduce some notation to allow us to write the Taylor series in matrix notation before we proceed with our discussion of solving (3.2.3). Let $\boldsymbol{\mathcal{P}}_k^{(d)}(\boldsymbol{u})$ denote the set of all k-dimensional monomials of degree d and lower, stored in ascending order as a column vector. If no ambiguities will arise, we will use $\boldsymbol{\mathcal{P}}$ in place of $\boldsymbol{\mathcal{P}}_k^{(d)}(\boldsymbol{u})$. For example, for second degree in 2D we have

$$\boldsymbol{\mathcal{P}}_2^{(2)}(\boldsymbol{u}) = \begin{bmatrix} 1 & u & v & u^2 & uv & v^2 \end{bmatrix}^T. \tag{3.2.4}$$

Let $\boldsymbol{D}$ be a diagonal matrix consisting of the fractional factorial part of the coefficients, i.e. $\frac{1}{j!k!}$ in (3.2.1). For example, for second degree in 2D we have

$$\boldsymbol{D} = \text{diag}\left(1,\ 1,\ 1,\ \frac{1}{2},\ 1,\ \frac{1}{2}\right). \tag{3.2.5}$$

Then we may write the Taylor series as

$$f(\boldsymbol{u}) = \boldsymbol{c}^T \boldsymbol{D}\boldsymbol{\mathcal{P}}(\boldsymbol{u}). \tag{3.2.6}$$

To solve (3.2.3), we use a weighted linear least squares formulation [38], that is, we will minimize a weighted norm (or semi-norm)

$$\min_{\boldsymbol{c}} \|\boldsymbol{V}\boldsymbol{c} - \boldsymbol{f}\|_{\boldsymbol{W}} \equiv \min_{\boldsymbol{c}} \|\boldsymbol{W}(\boldsymbol{V}\boldsymbol{c} - \boldsymbol{f})\|_2, \tag{3.2.7}$$

where $\boldsymbol{W}$ is an $m \times m$ diagonal weighting matrix. The entries of $\boldsymbol{W}$ assign weights to the rows of matrix $\boldsymbol{V}$. Specifically, if we denote the diagonal entries of $\boldsymbol{W}$ as w_i, then row i is assigned weight w_i. These weights can be used to prioritize the points in the system: we assign heavier weights to the nodes that are closer to the center point. By setting a weight to zero (or very close to zero), we may also filter out outliers or other undesirable points. Note that for a given node, the weighting matrix $\boldsymbol{W}$ is constant.

If $\boldsymbol{f}$ is in the column space of $\boldsymbol{V}$, then the solution of the linear system is not affected by a nonsingular weighting matrix. However, if $\boldsymbol{f}$ is not in the column space, which is often the case, then different weighting schemes can lead to different solutions. Choosing a good weighting matrix is application specific. For quadratic approximations, we compute the weights as follows. Let h denote the maximum radius of the neighborhood, that is

$$h = \max_{1 \le i \le m} \{\|\boldsymbol{u}_i\|_2\}. \tag{3.2.8}$$

Then

$$w_i = \left(\frac{\|\boldsymbol{u}_i\|_2}{h} + \epsilon\right)^{-1}, \tag{3.2.9}$$

where ϵ is a small number, such as $\epsilon = 0.01$, for avoiding division by zero.

After the weighting matrix has been applied, we can denote the new system as

$$\boldsymbol{M}\boldsymbol{c} \approx \tilde{\boldsymbol{f}}, \quad \text{where} \quad \boldsymbol{M} = \boldsymbol{W}\boldsymbol{V} \text{ and } \tilde{\boldsymbol{f}} = \boldsymbol{W}\boldsymbol{f}. \tag{3.2.10}$$

This resulting system may be rank-deficient or ill-conditioned. This is a challenge that GFD researchers have been dealing with since the 1970s [44]. The ill-conditioning may arise from a number of issues including poor scaling, an insufficient number of nodes in the neighborhood, or a degenerate arrangement of points. We resolve these issues with neighborhood selection, discussed in Section 5.2. We can address the scaling issue with the use of a diagonal scaling matrix $\boldsymbol{S}$. Let $\boldsymbol{a}_j$ denote the jth column of an arbitrary matrix $\boldsymbol{A}$. A typical choice for the jth entry of

$\boldsymbol{S}$ is either $1/\left\|\boldsymbol{a}_j\right\|_2$, which approximately minimizes the 2-norm condition number of $\boldsymbol{AS}$ [38], or $1/\left\|\boldsymbol{a}_j\right\|_\infty$ [17]. Using exact arithmetic, the matrix $\boldsymbol{S}$ does not affect the solution, but it can significantly improve the conditioning and thus the accuracy in the presence of rounding errors. After applying the scaling matrix to $\boldsymbol{WV}$, the problem becomes

$$\min_{\boldsymbol{d}} \left\|\tilde{\boldsymbol{V}}\boldsymbol{d} - \tilde{\boldsymbol{f}}\right\|_2, \quad \text{where } \tilde{\boldsymbol{V}} \equiv \boldsymbol{WVS} = \boldsymbol{MS} \text{ and } \boldsymbol{d} \equiv \boldsymbol{S}^{-1}\boldsymbol{c}. \tag{3.2.11}$$

Conceptually, the solution to the above problem may be reached through the use of a pseudoinverse. We will have

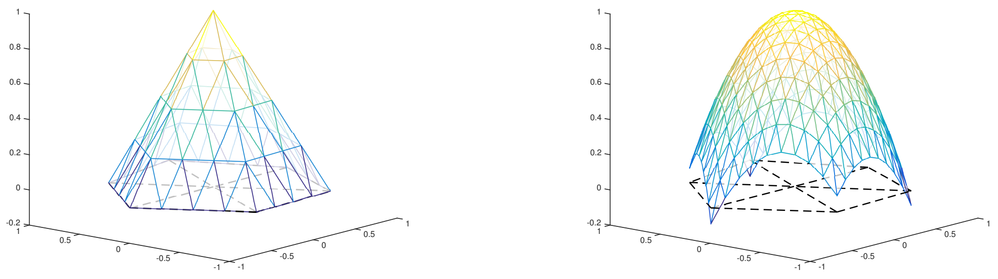
$$\boldsymbol{d} = \tilde{\boldsymbol{V}}^+ \tilde{\boldsymbol{f}} \quad \text{where } \tilde{\boldsymbol{V}}^+ \equiv \left(\tilde{\boldsymbol{V}}^T \tilde{\boldsymbol{V}}\right)^{-1} \tilde{\boldsymbol{V}}^T. \tag{3.2.12}$$

However, since the resulting system may still be rank-deficient or ill-conditioned, we solve it using QR factorization with column pivoting, as discussed in Subsection 5.3. Finally, we get the vector of partial derivatives for the Taylor polynomial

$$\boldsymbol{c} = \boldsymbol{Sd}. \tag{3.2.13}$$

3.3 Description of Generalized Lagrange Polynomial Basis Functions

We now define basis functions based on weighted least squares. Note that the standard finite element methods use piecewise Lagrange polynomial shape functions, which have two especially important properties: the coefficients of the basis func-



(a) FEM ‘hat’ basis function (a) AES-FEM GLP basis function

Figure 3.1: The ‘hat’ basis function exhibits the Kronecker Delta property, whereas a GLP basis function does not.

tions have the physical meaning of the function values or their approximations at the nodes, and the basis functions form a partition of unity. We refer to the two properties as *function value as coefficient* and *partition of unity*, respectively. These properties are desirable in ensuring the consistency of the method based on these basis functions and also for the ease of imposing Dirichlet boundary conditions. However, the traditional concept of the Lagrange basis functions is interpolatory, so they are not applicable to least squares. We now generalize this concept, so that it can be applicable to least-squares-based basis functions. Figure 3.1 contains a ‘hat’ function, which is an example of a linear Lagrange function, and an example of a quadratic GLP basis function.

Definition 1. Given a set of degree-d polynomial basis functions $\{\phi_i\}$, we say it is a set of degree-d *generalized Lagrange polynomial (GLP) basis functions* if:

1. $\sum_i f\left(x_i\right)\phi_i$ approximates a function f to $\mathcal{O}\left(h^{d+1}\right)$ in a neighborhood of the stencil, where h is some characteristic length measure, and

2. $\sum_i \phi_i = 1$.

We now define a set of GLP basis functions based on weighted least squares.

Given a stencil $\{x_i\}$, we follow the procedure in Subsection 3.2. When computing the jth basis function ϕ_j, let $\boldsymbol{f} = \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the jth column of the identity matrix. Following (3.2.12) and (3.2.13), we have

$$\boldsymbol{c} = \boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}\boldsymbol{e}_j. \tag{3.3.1}$$

Thus for the jth basis function, the vector $\boldsymbol{c}$ is exactly the jth column of $\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}$. We define a set of basis functions as

$$\boldsymbol{\Phi} = \left(\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}\right)^{T}\boldsymbol{D}\boldsymbol{\mathcal{P}}. \tag{3.3.2}$$

For a more concrete example, if we denote the elements of $\tilde{\boldsymbol{V}}^{+}$ as a_{ij} we can see that the jth basis function for degree 2 in 2D is

$$\phi_j = w_j\left(a_{1j}s_1 + a_{2j}s_2x + a_{3j}s_3y + a_{4j}s_4\frac{1}{2}x^2 + a_{5j}s_5xy + a_{6j}s_6\frac{1}{2}y^2\right). \tag{3.3.3}$$

Note that w_j is a constant scalar, so ϕ_j is a polynomial. The basis functions in (3.3.2) are an example of GLP basis functions. We summarize this key feature in the following theorem.

Theorem 2. *The basis functions based on weighted least squares as defined in (3.3.2) are generalized Lagrange polynomial basis functions.*

We shall postpone the proof of this theorem to Section 4.2, where we will also analyze the accuracy and stability of finite element discretization based on these basis functions. In the following, we will finish the description of AES-FEM.

3.4 Stable Computation of GLP Basis Functions

Even with proper neighborhood selection, it cannot be guaranteed that the least-squares problem in (3.2.11) will be well-conditioned. Hence, it is critical to use a robust method that ensures the accuracy and stability of the approximate solutions.

Note that one standard technique in linear algebra for solving rank-deficient least-squares problems is truncated singular value decomposition (TSVD) [38]. The TSVD is not recommended here, because it can result in the loss of partition of unity of the basis functions. The reason is as follows. When truncating SVD, one truncates any singular value σ_j that is smaller than $\epsilon\sigma_1$, where σ_1 is the largest singular value and ϵ is some small positive value, such as 10^{-4}. These singular values may be necessary for computing the constant terms in the GLP basis functions, and hence their loss can result in a set of basis functions that lack the partition of unity and in turn may compromise convergence.

We avoid the above issue by using truncated QR factorization with column pivoting (QRCP). When performing QRCP, one can find a numerical rank r of matrix $\boldsymbol{R}$ so that the condition number $\kappa(\boldsymbol{R}_{1:r,1:r}) < 1/\epsilon$, where ϵ is some small positive value, such as 10^{-4}. This is elaborated in the below discussion of Algorithm 1. If r is less than the size of $\boldsymbol{R}$, then any diagonal entry of $\boldsymbol{R}$ in position $r + 1$ or greater is truncated, thus truncating the $(r + 1)$th and subsequent columns of $\boldsymbol{Q}$. In QRCP, we require the first column not to be permuted, and this ensures the resulting basis functions satisfy the property of partition of unity.

In Algorithm 1, we present the procedure for initializing the generalized Vandermonde matrix $\tilde{\boldsymbol{V}}$ and factoring it using QRCP. The generalized Vandermonde matrix is formed from the local coordinates of the stencils and is scaled by the col-

Algorithm 1 Initialization of a Generalized Vandermonde Matrix

```
function: initiate_GVM
input: 1. $\boldsymbol{x}_k$: local coordinates of stencil
       2. $\boldsymbol{w}$: vector of row weights
       3. $p$: desired degree for $\boldsymbol{V}$
       4. $\epsilon$: tolerance for rank deficiency
output: struct gvm: with fields $\boldsymbol{W}$, $\boldsymbol{S}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{P}$, $r$ (estimated rank)
1: create generalized Vandermonde matrix $\boldsymbol{V}$ from local coordinates $\boldsymbol{x}_k$
2: determine column scaling matrix $\boldsymbol{S}$
3: $\boldsymbol{W} \leftarrow \text{diag}(\boldsymbol{w})$
4: $\tilde{\boldsymbol{V}} \leftarrow \boldsymbol{WVS}$
5: solve $\tilde{\boldsymbol{V}}\boldsymbol{P} = \boldsymbol{QR}$
6: estimate rank $r$ from $\boldsymbol{R}$ so that $r = \max\{i | \text{cond}(\boldsymbol{R}_{1:i,1:i}) \leq 1/\epsilon\}$
```

umn scaling matrix $\boldsymbol{S}$ and the row scaling matrix $\boldsymbol{W}$. The resulting matrix is then factored using QRCP. We use a variant of Householder triangularization [38] since this procedure is more efficient and stable than alternatives (such as Gram-Schmidt orthogonalization). When implementing this procedure, the QR factorization of $\tilde{\boldsymbol{V}}$ can overwrite $\boldsymbol{V}$. The jth Householder reflection vector is of size $n - j + 1$. By requiring the first element of the vector to be positive, the first element may be reconstructed from the other elements, and thus only $n - j$ entries are required to store the jth Householder reflection vector. The Householder vectors are stored in the lower triangular part of $\boldsymbol{V}$ and the $\boldsymbol{R}$ entries are stored in the upper part. The permutation matrix $\boldsymbol{P}$ is stored in a permutation vector.

In addition to computing the QR factorization of the generalized Vandermonde matrix, an estimation of the numerical rank of $\boldsymbol{R}$ is also computed in Algorithm 1. The rank is important for ensuring the overall stability of other algorithms that use this initialization step. In order to estimate the numerical rank, we estimate the condition numbers of the leading principal sub-matrices of $\boldsymbol{R}$, $\boldsymbol{R}_{1:r,1:r}$, and find the largest r such that $\tilde{\kappa}(\boldsymbol{R}_{1:r,1:r}) \leq 1/\epsilon$ where $\tilde{\kappa}$ is the estimated condition number and

Algorithm 2 Approximating $\mathcal{D}f$ at given point $\boldsymbol{x}$ from WLS

```
function diff_WLS
input: 1. struct gvm: with fields $\boldsymbol{W}$, $\boldsymbol{S}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{P}$, $r$ (estimated rank)
       2. coefficients $\boldsymbol{a} = \mathcal{DP}(\boldsymbol{x})$
output: weights $\boldsymbol{d}$, so that $\boldsymbol{d}^T\boldsymbol{g} = \mathcal{D}f(\boldsymbol{x})$ for $\boldsymbol{g}$ containing $f(\boldsymbol{x}_k)$ at stencil points
1: $\boldsymbol{a} \leftarrow (\boldsymbol{P}_{:,1:r})^T \boldsymbol{S}^{-1}\boldsymbol{a}$;
2: $\boldsymbol{a} \leftarrow \boldsymbol{R}_{1:r,1:r}^{-T}\boldsymbol{a}$;
3: $\boldsymbol{d} \leftarrow \boldsymbol{W}\boldsymbol{Q}_{:,1:r}\boldsymbol{a}$;
```

ϵ is some given drop-off tolerance depending on the degree of polynomials. Note that since the matrix $\boldsymbol{R}$ is small, the condition numbers in different norms differ by only a small factor. Therefore for efficiency, we estimate the condition number of $\boldsymbol{R}$ in the 1-norm using the algorithm described in [41].

Once the generalized Vandermonde matrix has been initialized, it may be used to construct generalized finite differentiation operators from the weighted least squares approximations, as described in Algorithm 2. The input for this algorithm is the output from Algorithm 1 and a vector $\boldsymbol{a}$. The vector $\boldsymbol{a} = \mathcal{D}\boldsymbol{\mathcal{P}}(\boldsymbol{x})$ contains the values for some specified derivative $\mathcal{D}$ of the monomial basis functions $\boldsymbol{\mathcal{P}}$ at point $\boldsymbol{x}$. For example, let $\mathcal{D}$ be $\frac{\partial}{\partial y}$. Then in 2D, we have $\boldsymbol{a} = \mathcal{D}\boldsymbol{\mathcal{P}}(x, y) = [0\;0\;1\;0\;x\;2y]^T$ and in 3D, we have $\boldsymbol{a} = \mathcal{D}\boldsymbol{\mathcal{P}}(x, y, z) = [0\;0\;1\;0\;0\;x\;0\;2y\;0\;0]^T$. The algorithm returns a vector of weights $\boldsymbol{d}$ so that $\boldsymbol{d}^T\boldsymbol{g} = \mathcal{D}f(\boldsymbol{x})$ for a vector $\boldsymbol{g} = [f_1\; f_2\; \dots\; f_m]^T$ containing the values of the function at the stencil points. Note that for a GLP basis function, the returned weights are the values of the specified derivative at the points in the stencil.

In terms of the computational cost, the step that dominates Algorithm 1 is the QRCP factorization, which takes $\mathcal{O}\left(2mn^2 - \frac{2}{3}n^3\right)$ flops where $\tilde{\boldsymbol{V}}$ is $m \times n$ [38]. Here, m is the number of points in the stencil and n is the number of terms in the Taylor series expansion (for second order expansion $n = 6$ in 2D and $n = 10$

in 3D). As long as the valance is bounded, that is m is bounded, this algorithm is executed in a constant time. Compared to calculations based on the standard finite-element basis functions, which are tabulated, the computation based on the GLP basis functions is more expensive. This leads to higher cost of AES-FEM in assembling the stiffness matrix and load vector, as we discuss next. However, this cost is a small constant per element, and AES-FEM can be more efficient overall by delivering higher accuracy, as we will demonstrate in Section 6.

3.5 Description of Quadratic AES-FEM

Starting with the weighted residual formulation for FEM from (3.1.9), we propose using GLP basis functions for the basis functions and using the traditional hat functions for the weight functions. More specifically, for a given node i and its associated weight function ψ_i, a specific set of GLP basis functions $\{\phi_j\}$ is constructed from a weighted stencil $(\boldsymbol{X}_i, \boldsymbol{w}_i)$ of n neighboring vertices centered at node i. This weight function ψ_i and its associated set of GLP basis functions $\{\phi_j\}$ are used to compute the ith row of the stiffness matrix, as given by (3.1.11). Note that, because a different set of basis functions is associated with each weight function, the basis functions on a given element differ row-to-row in the stiffness matrix.

For AES-FEM, when using the 1-ring neighborhood of a vertex as the stencil, we can use quadratic GLP basis functions, which have the advantage of decreased dependence on element quality and improved accuracy over standard finite element with linear basis functions, while virtually preserving the sparsity pattern of the stiffness matrix, as we will demonstrate in Section (6.1).

In terms of the computation of the load vector, we can use either the FEM or

AES-FEM basis functions. We refer to the AES-FEM with these two options as "AES-FEM 1" and "AES-FEM 2," respectively. Additional implementation details will be given in Subsection 5.3.

Chapter 4

Analysis and High-Order Generalization of AES-FEM

In this chapter, we analyze the consistency and stability of AES-FEM, and we present a generalization of AES-FEM to high-order. We start by explaining the applicability of Green's identity to the GLP basis functions. We will then prove that basis functions in (3.3.2) are GLP basis functions and discuss the consistency, accuracy, and stability of AES-FEM. Finally, explain the high-order extension and discuss high-order AES-FEM over curved boundaries using meshes with linear elements.

4.1 Green's Identity and Integration by Parts

For a given weight function ψ, the GLP basis functions are continuously differentiable everywhere in within the domain of integration. Hence, in a *variational sense*, the GLP basis functions still allow one to use Green's identities to formulate a weak form. Let ϕ_j be any GLP basis function computed from a weighted stencil, and ψ_i be a classical FEM shape function with compact support contained within the set Ω. Therefore, for any partial derivative operation ∂, it follows that

$$\int_\Omega (\partial\phi_j)\, \psi_i \, \mathrm{d}x = -\int_\Omega \phi_j \, (\partial\psi_i)\; \mathrm{d}x. \tag{4.1.1}$$

Because of this property, the weak-form formulation with the GLP basis functions in AES-FEM is mathematically sound.

In addition, since the finite-element basis functions ψ_i vanish along the boundary, after integration by parts, we have the identities

$$\int_\Omega \psi_i \nabla^2 \phi_j \, dV = -\int_\Omega \nabla\psi_i \cdot \nabla\phi_j \, dV. \tag{4.1.2}$$

Therefore, we can reduce the order of derivatives similar to the classical FEM, without introducing additional boundary integrals to the computation.

4.2 Properties of Generalized Lagrange Polynomial Basis Functions

We now show that the basis functions in (3.3.2) are indeed generalized Lagrange polynomial basis functions, which follows from the two lemmas below.

Lemma 3. *Let $\{x_i\}$ be a stencil with m nodes and stencil diameter h. Let $\{\phi_j\}$ be the complete set of basis functions of degree up to d as defined by (3.3.2) on this* *stencil. Given an arbitrary function f, define the following approximation of f*

$$f^h(x) \equiv \sum_{j=1}^{m} f_j \phi_j(x) = \boldsymbol{f}^T \boldsymbol{\Phi}, \tag{4.2.1}$$

where $f_j = f(x_j)$ and $\boldsymbol{f} = [f_1\ f_2\ \dots\ f_m]^T$. If the rescaled matrix $\tilde{\boldsymbol{V}}$ has a bounded condition number, then f^h approximates f to $\mathcal{O}\left(h^{d+1}\right)$. In addition, given any degree-k differential operator $\mathcal{D}$, if f is continuously differentiable up to degree k, then $\mathcal{D}f^h$ approximates $\mathcal{D}f$ to $\mathcal{O}\left(h^{d-k+1}\right)$.

Proof. First, we show that the approximation f^h is equivalent to directly solving a weighted least squares problem for a local polynomial fitting of f. Using the method described in Subsection 3.2, we get the coefficients $\boldsymbol{c} = \boldsymbol{S}\tilde{\boldsymbol{V}}^{\,+}\boldsymbol{W}\boldsymbol{g}$ where $\boldsymbol{g} = [f_1\ f_2\ \dots\ f_m]^T$. Thus, the local polynomial fitting of f is

$$\begin{aligned} f(\boldsymbol{x}) &\approx \boldsymbol{c}^T \boldsymbol{D}\boldsymbol{\mathcal{P}} \\ &= \left(\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}\boldsymbol{g}\right)^T \boldsymbol{D}\boldsymbol{\mathcal{P}} \\ &= \boldsymbol{g}^T \left(\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}\right)^T \boldsymbol{D}\boldsymbol{\mathcal{P}} \\ &= \boldsymbol{g}^T \boldsymbol{\Phi} \\ &= f^h(\boldsymbol{x}). \end{aligned}$$

It follows from Theorem 4 in [45] that f^h approximates f to $\mathcal{O}\left(h^{d+1}\right)$, and $\mathcal{D}f^h$ approximates $\mathcal{D}f$ to $\mathcal{O}\left(h^{d-k+1}\right)$ for any degree-k differential operator $\mathcal{D}$. □

Lemma 4. *The basis functions in (3.3.2) form a partition of unity, i.e., $\sum_{j=1}^{m} \phi_j(x) = 1$.*

Proof. We will show that on a given stencil $\{x_i\}$, the GLP basis functions $\{\phi_j\}$

form a partition of unity. Let $\boldsymbol{V}$ be the generalized Vandermonde matrix for the given stencil. For example, for second order expansion in 2D, we have

$$\boldsymbol{V} = \begin{bmatrix} 1 & x_1 & y_1 & \dots & \frac{1}{2}y_1^2 \\ 1 & x_2 & y_2 & \dots & \frac{1}{2}y_2^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & y_m & \dots & \frac{1}{2}y_m^2 \end{bmatrix}. \tag{4.2.2}$$

For a given function f, using the truncated Taylor series, we have

$$\boldsymbol{V}\boldsymbol{c} = \boldsymbol{g}, \tag{4.2.3}$$

where $\boldsymbol{c}$ is the vector of partial derivative values. Applying the diagonal row weighting matrix $\boldsymbol{W}$ and the diagonal column scaling matrix $\boldsymbol{S}$, we have

$$\tilde{\boldsymbol{V}}\left(\boldsymbol{S}^{-1}\boldsymbol{c}\right) = \boldsymbol{W}\boldsymbol{g} \quad \text{where} \quad \tilde{\boldsymbol{V}} \equiv \boldsymbol{W}\boldsymbol{V}\boldsymbol{S}. \tag{4.2.4}$$

This is a least squares problem, and the solution for $\boldsymbol{c}$ may be reached through the use of a pseudoinverse,

$$\boldsymbol{c} = \boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}\boldsymbol{g}. \tag{4.2.5}$$

For the jth GLP basis function, we have $\boldsymbol{g}_j = [0 \dots 1 \dots 0]^T$, where the 1 is in the jth position, and hence the columns of $\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}$ multiplied by the Taylor constants $\boldsymbol{D}$ give the coefficients for the GLP basis functions. This implies that the entries of the ith row of $\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}$ correspond to the coefficients of the ith terms in the set of basis functions.

To finish the proof, it suffices to show that the sum of the entries in the first row of $\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}$ is 1, and the sum of the entries in any other row is 0. Let vector $\boldsymbol{w}$ be the diagonal entries of $\boldsymbol{W}$. Every entry of the first column of $\boldsymbol{V}$ is equal to 1, thus the first column of $\tilde{\boldsymbol{V}}$ is then $s_1\boldsymbol{w}$. Denote the ith row of $\tilde{\boldsymbol{V}}^{+}$ as $\tilde{\boldsymbol{v}}^{+}_{(i,:)}$. The sum of the entries of the ith row of $\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}$ is $s_1\tilde{\boldsymbol{v}}^{+T}_{(i,:)}\boldsymbol{w}$. Since $\tilde{\boldsymbol{V}}^{+}$ is a left inverse of $\tilde{\boldsymbol{V}}$, we have $\tilde{\boldsymbol{V}}^{+}\tilde{\boldsymbol{V}} = \boldsymbol{I}$, and hence

$$s_1\tilde{\boldsymbol{v}}^{+T}_{(i,:)}\boldsymbol{w} = \begin{cases} 1 & i = 1 \\ \\ 0 & 2 \leq i \leq n \end{cases}. \tag{4.2.6}$$

Therefore, the GLP basis functions form a partition of unity. □

From the above lemmas, the basis functions in (3.3.2) satisfy both the properties of function value as coefficient and partition of unity, and hence they are GLP basis functions, as claimed in Theorem 2.

4.3 Consistency and Accuracy of AES-FEM

The accuracy of the AES-FEM depends on its consistency and stability. We first consider the consistency of AES-FEM, in terms of its truncation errors in approximating the weighted-residual formulation (3.1.3) by (3.1.6) using the GLP basis functions. In a nutshell, the consistency of the AES-FEM follows directly from Lemma 3. For completeness, we consider a specific example of solving the Poisson equation. The analysis for other PDEs can be derived in a similar fashion.

Theorem 5. *Consider a bounded domain Ω with a piecewise smooth boundary Γ. Suppose U is smooth and thus $\|\nabla U\|$ is bounded. Then, when solving the Poisson equation on Ω using AES-FEM with degree-d GLP basis functions in (3.1.9), for* *each ψ_i the weak form (3.1.9) is approximated to $\mathcal{O}(ch^d)$, where h is some characteristic length measure of the mesh and $c = \int_\Omega ||\nabla\psi_i||\, dx$.*

Proof. Let u be the exact solution on a mesh with mesh size h, and let

$$\tilde{U} = \sum_{j=1}^{n} x_i \phi_j \tag{4.3.1}$$

denote the approximation to u using degree-d GLP basis functions. When using degree-d GLP basis functions, it follows from Lemma 3 in [23] that

$$\|\nabla U - \nabla \tilde{U}\| = \mathcal{O}\left(h^d\right) \tag{4.3.2}$$

within each element. Under the assumption that U is twice differentiable, ∇U is bounded, and hence

$$\left|\int_\Omega \left(\nabla U - \nabla\tilde{U}\right)\cdot\nabla\psi_i\, dV\right| = \mathcal{O}\left(h^d\right)\left|\int_\Omega \nabla U\cdot\nabla\psi_i\, dV\right|. \tag{4.3.3}$$

Assume $\|\nabla U\|$ is bounded, then $\left|\int_\Omega \nabla U\cdot\nabla\psi_i\, dV\right| = \mathcal{O}\left(\int_\Omega \|\nabla\psi_i\|\, dV\right)$.

Furthermore, if ρ is approximated by degree-d GLP basis functions as $\tilde{\rho} = \sum_{j=1}^{n}\rho_i\phi_j$, then we have $\left|\int_\Omega (\rho-\tilde{\rho})\,\psi_i\, dV\right| = \mathcal{O}\left(h^{d+1}\right)\left|\int_\Omega \rho\psi_i\, dV\right|$. Assume ρ is bounded, then $\left|\int_\Omega \rho\psi_i\, dV\right| = \mathcal{O}\left(\int_\Omega \psi_i\, dV\right)$, which is higher order than $\int_\Omega \|\nabla\psi_i\|\, dV$ for hat functions. Therefore, the residual in the weak form is $\mathcal{O}(ch^d)$, where $c = \mathcal{O}\left(\int_\Omega \|\nabla\psi_i\|\, dV\right)$. □

More specifically, when using quadratic GLP basis functions, the truncation errors are second order in the stiffness matrix. The truncation errors in the load vector is third order for quadratic AES-FEM 2 when ρ is also approximated using quadratic GLP basis functions, but it is second order for quadratic AES-FEM 1 when ρ is approximated using linear FEM hat functions.

4.4 High-Order Basis Functions

In AES-FEM, we use the GLP basis functions for ϕ_j, where the degree of the basis functions may vary. For the test functions ψ_i, we use the piecewise-linear traditional FEM shape functions, a.k.a. the "hat" functions, independent of the degree of the basis functions. Note that for different ψ_i, we may use different basis functions, constructed using WLS with different stencils at each node. Therefore, it is possible to adapt the degree of the polynomials for different ψ_i. In this work, we shall assume that the degree of the basis functions is uniform.

The order of convergence of AES-FEM is determined by the degree of the basis functions, as summarized by Theorem 5. Like most other PDE methods, as long as the method is stable, the rounding errors do not dominate the truncation errors, and there is no systematic cancelation of truncation errors, we expect the solution to converge at the same rate as the local truncation errors. This hold for even-degree basis functions for even-degree PDEs, as we will demonstrate numerically in Chapter 6. In Section 4.4, we discuss systematic cancelation of odd-degree basis functions for an even-degree PDE, which results in a convergence rate one degree lower than the degree of the basis function.

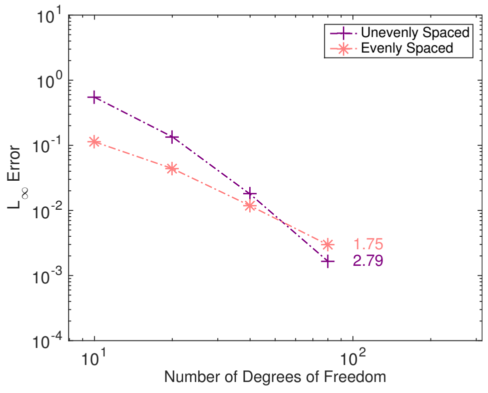
However, when using AES-FEM with odd-degree basis functions, the situation is more complicated. In practice, we observe that if the highest-order spatial derivative of the PDE is even, which is typically the case for elliptic and parabolic PDEs, the order of convergence may be one less than the degree of the basis functions when using odd-degree polynomials. This is because of the cancelation of the odd functions in the integrals over equally spaced grids. To illustrate this effect, let us consider an odd-degree polynomial $f(x) = x^{2m+1}$. Suppose the two elements incident on the node at $x = 0$ are $[-l_1, 0]$ and $[0, l_2]$, and the test function associated the node is ψ. On an equally spaced grid, $l_1 = l_2$, $\psi'(x) = -\psi'(-x)$ and $f'(x) = f'(-x)$. Therefore,

$$\int_{-l}^{0} \psi'(x)\, f'(x)\, dx = -\int_{0}^{l} \psi'(x)\, f'(x)\, dx. \tag{4.4.1}$$

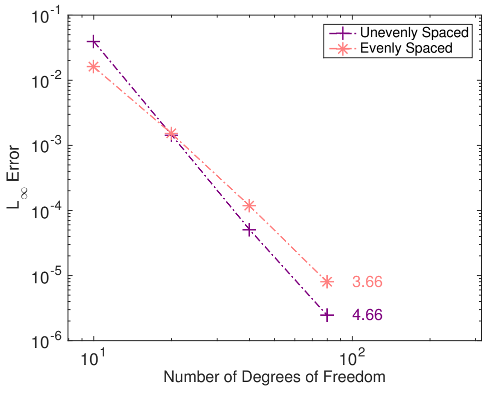
Integrating the weak form corresponding to the test function ψ, we obtain

$$\int_{\Omega} \psi'\, f'\, dx = \int_{-l}^{l} \psi'\, f' = \int_{0}^{l} \psi'\, f'\, dx - \int_{0}^{l} \psi'\, f'\, dx = 0. \tag{4.4.2}$$

For nearly even-spaced grids, which is typically the case, the integral corresponding to odd-degree polynomials would be close to zero, so using degree-$(2m + 1)$ polynomials would lead to similar errors as using degree-$2m$ polynomials. Figure 4.1 demonstrates this behavior numerically with the Poisson equation in 1D using degree-3 and 5 basis functions. It can be seen that on unequally spaced grids, the convergence rate is approximately equal to the degree of polynomials, but for equally spaced grids, the convergence was lower, although the error may be smaller. This behavior can also be observed in 2D and 3D. Therefore, we will use only even-degree basis functions in the week forms for even-order PDEs. For odd-order PDEs, odd-degree basis functions would be recommended for AES-FEM.



(a) Degree-3 AES-FEM.



(b) Degree-5 AES-FEM.

Figure 4.1: The errors from AES-FEM with odd-degree basis functions for the Poisson equation in 1D. The number to the right of each curve indicates the average convergence rate.

4.5 Treatment of Curved Geometries

Another critical issue of AES-FEM is the resolution of curved geometries. For most high-order FEM methods, elements near the boundary must be curved and approximate the boundary to a high enough order [7, 50]. This is because the mid-edge or mid-face nodes of those high-order elements must approximate the geometry accurately. In AES-FEM, it is also important for all the nodes to approximate the boundary to high-order accuracy. However, because the basis functions in AES-FEM are constructed from the nodes, independently of the elements, and its test functions are hat functions, AES-FEM uses linear elements without mid-edge or mid-face nodes. Although the linear elements gives only a piecewise linear approximation to the boundary, the effect of this linear approximation is confined in the approximation to the weighted residual formulation (3.1.3) without compromising the order of accuracy. Specifically, instead of solving (3.1.3), we would be solving a perturbed integral equation

$$\int_{\Omega^h} \psi_i \left(\mathcal{L}U - \rho\right)\, dV = 0, \tag{4.5.1}$$

where Ω^h denotes the geometric realization of a linear mesh of Ω. Note that (4.5.1) remains an exact equality for the exact U and ρ, and hence Theorem (5) remains valid for the equation

$$\int_{\Omega^h} \psi_i \nabla^2 U \, dV = \int_{\Omega^h} \psi_i \rho \, dV \tag{4.5.2}$$

in place of (3.1.7). Therefore, after substituting the numerical approximations of U into 3.1.9, the order of the local truncation errors remains the same, and hence the order of convergence is preserved. The only compromise of using Ω^h instead of Ω is that if the test functions $\boldsymbol{\Psi}$ forms a partition of unity, the global conservation is satisfied in the sense of $\int_{\Omega^h} (\mathcal{L}U - \rho)\, dV = 0$ instead of $\int_{\Omega} (\mathcal{L}U - \rho)\, dV = 0$. This slight deviation of global conservation does not constitute a problem, because global conservation is in general never satisfied exactly even in FEM after boundary conditions are imposed.

To discretize the PDE fully, it is important that boundary conditions are imposed in a fashion that preserves the order of accuracy. For Dirichlet boundary conditions, we enforce them strongly by simply substituting the function values of Dirichlet nodes into the equations, thanks to the use of GLP basis functions. For Neumann boundary conditions, we also propose to impose them strongly by using the Dirac delta function as test functions at Neumann nodes, multiplied by $\mathcal{O}(h^{d-1})$ in d-dimensions to ensure the stiffness matrix is properly scaled. This is analogous to the generalized finite difference method except for the scaling part. This approach ensures the consistency of the discretization, and it avoids the need of high-order accurate boundary integrals, which would have been required if Neumann boundary conditions were to be imposed weakly as in FEM. Note that accurate normals at Neumann nodes are still required, which can be computed to high-order accuracy as described in [45]. In addition, since the stencils for Neumann nodes are one-sided, it requires special care to ensure stability in the computation of the GLP basis functions. We defer the robust treatment of Neumann boundary conditions to future work, and focus on Dirichlet boundary conditions in our numerical experimentations.

4.6 Stability

For elliptic PDEs, the stability of a method depends on the condition number of its coefficient matrix, which can affect the performance of iterative solvers and the accuracy of the solution. It is well known that the traditional finite element method may be unstable for poorly shaped meshes [3], and some meshless methods may also suffer from instability when two points nearly coincide. AES-FEM avoids these potential instability issues.

As a concrete example, let us consider the Poisson equations with Dirichlet boundary conditions, whose coefficient matrix is the stiffness matrix. It is well known that the condition number of the stiffness matrix is proportional to h^{-2}, where h is some characteristic length of the mesh [55]. However, if the condition number is significantly larger, then the method is said to be *unstable*, which can happen due to various reasons.

The ill-conditioning of any local stiffness matrix may lead to poor scaling and in turn ill-conditioning of the global stiffness matrix. This is owing to the following fact, which is given as Theorem 2.2.26 in [91].

Proposition 6. *For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $m \geq n$, its condition number in any p-norm, denoted by $\kappa_p(\boldsymbol{A})$, is bounded by the ratio of the largest and smallest column vectors in p-norm, i.e.,*

$$\kappa_p(\boldsymbol{A}) \geq \frac{\max_{1\leq i\leq n} \left\|\boldsymbol{a}_i\right\|_p}{\min_{1\leq j\leq n} \left\|\boldsymbol{a}_j\right\|_p}, \tag{4.6.1}$$

where $\boldsymbol{a}_k$ denotes the kth column vector of $\boldsymbol{A}$.

The above fact offers an intuitive explanation of a source of ill-conditioning in traditional finite element methods due to poorly shaped elements: poorly shaped elements may lead to unbounded large entries in local stiffness matrices, so the column norms of the global stiffness matrix would vary substantially, and in turn the global stiffness matrix is necessarily ill-conditioned. In the context of AES-FEM, there can be two potential sources of local instability due to poor scaling. First, the unnormalized local Vandermonde system given in (3.2.3) is in general very poorly scaled. We resolved this by normalizing the Vandermonde system to avoid poor scaling. Second, the normalized Vandermonde system may still be ill-conditioned occasionally, when a stencil is degenerate or nearly degenerate, which could lead to unbounded large values in the local stiffness matrix. We resolved this issue by using QR with column pivoting and condition-number estimation.

Even if the local stiffness matrices are bounded, the global stiffness matrix may still be ill-conditioned due to linearly dependent rows or columns. This is the potential source of instability for some meshless methods when two points nearly coincide; the two points may share the same stencil and basis functions, so that the rows or columns corresponding to the two points would be nearly identical. Therefore, these meshless methods also require good point distributions. In AES-FEM, we utilize the mesh topology to construct the stencil, as we will describe in Section 5.2. This ensures that no two vertices share the same stencil unless there are coincident points, and hence it gives a strong guarantee that the rows in the global stiffness matrix are linearly independent.

The aforementioned reasons are the most common causes of instability for solving elliptic PDEs. Another source of instability is a cluster of coincident points or inverted elements, which rarely happen in practice, and hence we defer their treatments to future work. As we will demonstrate numerically in Chapter 6, by resolving these instabilities, AES-FEM produces well-conditioned stiffness matrices for meshes, even with very bad quality elements or point distributions.

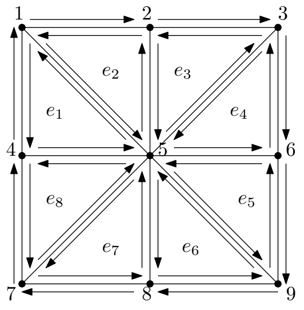
Chapter 5

Implementation

We discuss the practical aspects of the implementation of AES-FEM in this chapter. We start with a discussion of the utilized mesh data structure and then explain how this enables quick and efficient neighborhood selection. Finally, the algorithms are presented and runtime is analyzed.

5.1 Data Structure

We use an Array-based Half-Facet (AHF) data structure [26] to store the mesh information. In a d-dimensional mesh, the term *facet* refers to the $(d-1)$-dimensional mesh entities; that is, in 2D the facets are the edges, and in 3D the facets are the faces. The basis for the half-facet data structure is the idea that every facet in a manifold mesh is made of two half-facets oriented in opposite directions. We refer to these two half-facets as *sibling half-facets*. Half-facets on the boundary of the domain have no siblings. In 2D and 3D, the half-facets are *half-edges* and *half-faces*, respectively. We identify each half-facet by a two tuple: the element ID and a local



element	vertices		
1	1	4	5
2	1	5	2
3	2	5	3
4	3	5	6
5	6	5	9
6	8	9	5
7	7	8	5
8	5	4	7

element	sibhes		
1	nil	$\langle 8,1\rangle$	$\langle 2,1\rangle$
2	$\langle 1,3\rangle$	$\langle 3,1\rangle$	nil
3	$\langle 2,2\rangle$	$\langle 4,1\rangle$	nil
4	$\langle 3,2\rangle$	$\langle 5,1\rangle$	nil
5	$\langle 4,2\rangle$	$\langle 6,2\rangle$	nil
6	nil	$\langle 5,2\rangle$	$\langle 7,2\rangle$
7	nil	$\langle 6,3\rangle$	$\langle 8,3\rangle$
8	$\langle 1,2\rangle$	nil	$\langle 7,3\rangle$

vertex	v2he
1	$\langle 1,1\rangle$
2	$\langle 2,3\rangle$
3	$\langle 3,3\rangle$
4	$\langle 8,2\rangle$
5	$\langle 1,3\rangle$
6	$\langle 4,3\rangle$
7	$\langle 7,1\rangle$
8	$\langle 6,1\rangle$
9	$\langle 5,3\rangle$

Figure 5.1: An example of half edges and associated data structure.

facet ID within the element. In 2D, we store the element connectivity, sibling half-edges, and a mapping from each node to an incident half-edge. In 3D, we store the element connectivity, sibling half-faces, and a mapping from each node to an incident half-face. For an example of a 2D mesh and the associated data structure, see Figure 5.1. This data structure allows us to do neighborhood queries for a node in constant time (provided the valance is bounded). For additional information about the AHF data structure, see [26].

5.2 Neighborhood Selection

5.2.1 Neighborhood Selection for Quadratic AES-FEM

The use of the AHF data structure allows us to quickly find the neighborhood of a node. We use the concept of rings to control the size of the neighborhood. The *1-ring neighbor elements* of a node are defined to be the elements incident on the node. The *1-ring neighborhood* of a node contains the nodes of its 1-ring neighbor elements [45]. Most of the time, when using GFD with second order basis functions or when constructing second order GLP basis functions, the 1-ring neighborhood of

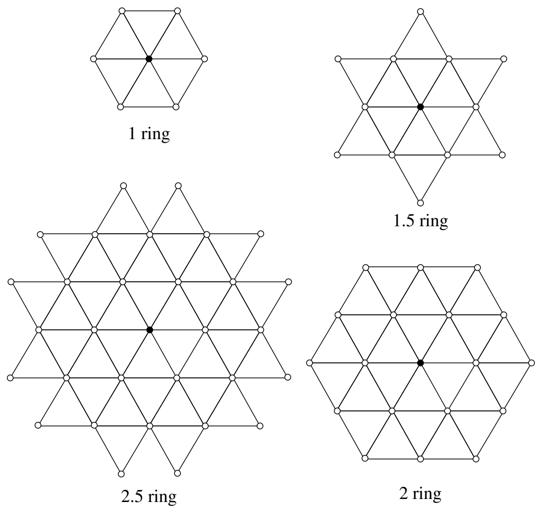


Figure 5.2: Examples of 2D stencils with 1-ring, 1.5-ring, 2-ring, and 2.5-ring neighborhoods of center node (in solid black).

a node supplies the appropriate number of nodes. If the valance is low, it might be necessary to further expand and collect more nodes for the neighborhood. Therefore, for any integer $k \geq 1$, we define the $(k+1)$-*ring neighborhood* as the nodes in the k-ring neighborhood plus their 1-ring neighborhoods.

As k increases, the average size of the k-ring neighborhood grows very quickly. The granularity can be fine-tuned by using fractional rings. In 2D we use half rings, which are defined in [45]; for any integer $k \geq 1$ the $(k + {}^1\!/_2)$-*ring neighborhood* is the k-ring neighborhood plus the nodes of all the faces that share an edge with the k-ring neighborhood. See Figure 5.2 for a visualization of rings and half-rings in 2D. In 3D, we use one-third and two-third rings, as defined in [23]; for any integer $k \geq 1$, the $(k + {}^1\!/_3)$-*ring neighborhood* contains the k-ring neighborhood plus the nodes of all elements that share a face with the k-ring neighborhood. The $(k + {}^2\!/_3)$-*ring neighborhood* contains the k-ring neighborhood plus the nodes of all faces that share an edge with the k-ring neighborhood.

Table 5.1: Comparison of the average number of nodes per ring versus the number of coefficients for 2D (left) and 3D (right) Taylor polynomials.

Degree	#Coeffs.	Ring	#Nodes
2	6	$1^1/_2$	12.85
3	10	2	19.55
4	15	$2^1/_2$	30.53
5	21	3	39.07
6	28	$3^1/_2$	54.80

Degree	#Coeffs.	Ring	#Nodes
2	10	1	15.46
3	20	$1^1/_3$	31.89
4	35	$1^2/_3$	50.17
5	56	2	72.64
6	84	$2^1/_3$	127.83

Note that for 2D triangular and 3D tetrahedral meshes, the 1-ring neighborhood typically has enough points for constructing quadratic GLP basis functions. Therefore, the stiffness matrix from AES-FEM has a similar sparsity pattern to that from standard FEM with linear shape functions. However, when the 1-ring neighborhood is too small, the extended stencil with a larger ring allows AES-FEM to overcome mesh-quality dependence and improve its local stability.

5.2.2 Neighborhood Selection for High-Order AES-FEM

To achieve high-order accuracy, a critical question is the selection of the stencils at each node for the construction of the GLP basis functions. We utilize meshes for speedy construction of the stencils. Given a simplicial mesh (i.e. a triangle mesh in 2D or a tetrahedral mesh in 3D), we use the concept of ring, as defined above, to select the neighborhood.

In Table (5.1), we compare the average number of nodes in a given ring, denoted by m, to the number of unknowns for a given degree in (3.2.3), denoted by n. We have found that having approximately $m \approx 1.5n$ works well. For 2D triangular meshes, the $1^1/_2$-ring has an appropriate number of nodes for quadratic basis functions. The 2-ring, $2^1/_2$-ring, 3-ring, and $3^1/_2$-ring typically provide an appropriate number of nodes for degree 3 to 6 basis functions, respectively. For 3D tetrahedral meshes, the 1-ring, $1^1/_3$-ring, $1^2/_3$-ring, 2-ring, and $2^1/_3$-ring have an appropriate number of nodes for degrees 2 to 6 basis functions, respectively. If a particular neighborhood does not provide enough points, we further expand the stencil to a larger ring. This allows AES-FEM to overcome element-quality dependence and also to improve the stability of its local computations.

5.3 Assembly of Stiffness Matrix and Load Vector

Algorithm 3 presents a summary of the AES-FEM procedure for assembling the stiffness matrix and load vector for a PDE with Dirichlet boundary conditions. Unlike the standard FEM procedure, we build the stiffness matrix row by row, rather than element by element. This is because the most computationally expensive part of the procedure is to compute the derivatives for the set of basis functions on each stencil. A weight function is nonzero only on the neighborhood around its corresponding node. Since the weight functions correspond to the rows, we assemble the stiffness matrix row by row, ensuring that we will only need to compute the derivatives for each neighborhood once.

When computing a row of the stiffness matrix, the first step is to obtain the stencil of node k. This step is performed by utilizing the data structure presented in Subsection 5.1 and the proper size of the stencil is ensured by choosing the ring sizes adaptively. Next, the local coordinates are calculated for the points in the stencils and the row weights are computed. Using Algorithm 1, the QR factorization of the generalized Vandermonde matrix is computed for the neighborhood. Then for each element that contains node k, we perform the integration of the weak form in a manner that is similar to standard FEM. The element Jacobian is computed and used to find the local coordinates of the quadrature points. The derivatives of the weight function are computed, that is $\nabla\psi_i$, at the quadrature points of the current element. Recall that the weight functions are the standard hat functions. Next the derivatives of the basis functions, that is $\nabla\phi_j$, are computed at the quadrature points of the current element. The basis functions are the GLP basis functions and thus Algorithm 2 is used. The value of the integral on the current element is computed and either added to the stiffness matrix or subtracted from the load vector, depending on whether the basis function corresponds to a node with Dirichlet boundary conditions.

When computing the load vector, typically the entries $b_i = \int \psi_i \rho \, dV$ are computed using a quadrature rule. One may evaluate ρ at the quadrature points in two ways. The first way is to use the standard procedure in FEM, i.e., to use the FEM basis functions and the values of ρ at the nodes of the element. Let $\boldsymbol{m}$ be the vector of FEM shape functions evaluated at quadrature point $\boldsymbol{x}_k$ and $\boldsymbol{g}_{\text{elem}}$ be the vector of the function values at the nodes of the element. Then, we have

$$\rho\left(\boldsymbol{x}_k\right) = \boldsymbol{m}^T \boldsymbol{g}_{\text{elem}}. \tag{5.3.1}$$

Alternatively, we may use GLP basis functions to interpolate the values of ρ at the quadrature points. We can approximate an arbitrary function using the set of GLP basis functions. In matrix notation, we have

$$\rho\left(\boldsymbol{x}_k\right) = \boldsymbol{g}_{\text{sten}}^T \left(\boldsymbol{S}\tilde{\boldsymbol{V}}^{+}\boldsymbol{W}\right)^T \boldsymbol{D}\boldsymbol{\mathcal{P}}\left(\boldsymbol{x}_k\right), \tag{5.3.2}$$

where $\boldsymbol{g}_{\text{sten}}$ is the vector of function values at the nodes in the stencil and the vector $\boldsymbol{\mathcal{P}}\left(\boldsymbol{x}_k\right)$ has been evaluated at the quadrature point $\boldsymbol{x}_k$. As mentioned earlier, we refer to the variant of AES-FEM using the former method of calculating the load vector as AES-FEM 1 and refer to the latter variant as AES-FEM 2.

We use Gaussian quadrature to perform the integration within each element. For quadratic GLP basis functions in 2D, we use a 1-point rule for stiffness matrix, and a 3-point rule for the load vector. In 3D, we use a 1-point rule for the stiffness matrix and a 4-point rule for the load vector. These rules are exact because the basis functions and their derivatives are quadratic and linear, respectively.

It is worth noting that because of the properties of generalized Lagrange polynomial basis functions, Dirichlet boundary conditions may be imposed in AES-FEM in the same manner as in standard FEM. One does not need to use Lagrange multipliers or a penalty method. Additionally, the standard method for imposing Neumann boundary conditions may be used in AES-FEM.

All the steps inside of the primary for-loop are executed in constant time, assuming that the size of each neighborhood is bounded. Therefore, the assembly of the stiffness matrix in AES-FEM has an asymptotic runtime of $\mathcal{O}(n)$, where n is the number of nodes in the mesh. When using quadratic AES-FEM 2, that is when WLS approximation is used to compute the approximation of ρ at the quadrature points, Algorithm 2 is called again. While this function is constant in runtime, it has a large coefficient and thus takes longer than approximating the values of f using FEM (hat) basis functions. Therefore, the assembly time for quadratic AES-FEM 2 is longer than that for quadratic AES-FEM 1, as can be seen in Chapter 6.

Finally, for completeness, we include Algorithm 4 to summarize the GFD procedure for solving PDEs with Dirichlet boundary conditions. We use this algorithm

Algorithm 3 Building a Stiffness Matrix and Load Vector using AES-FEM

```
function: aes_fem
input: 1. $\boldsymbol{x}$, elem, opphfs, vh2f: mesh information
       2. $p$: desired degree for GLP functions
       3. $\epsilon$: tolerance for rank deficiency
       4. AESFEM1: boolean for AES-FEM 1 or AES-FEM 2
       5. isDBC: flags for Dirichlet boundary conditions
output: stiffness matrix $\boldsymbol{K}$ and load vector $\boldsymbol{b}$
 1: for each node without Dirichlet boundary conditions do
 2:     obtain neighborhood of node
 3:     calculate local parameterization $\boldsymbol{x}_k$ and row weights $\boldsymbol{w}$ for neighborhood
 4:     aes_gvm ← initiate_GVM($\boldsymbol{x}_k$, $\boldsymbol{w}$, $p$, $\epsilon$)
 5:     obtain local element neighborhood
 6:     for each element in local neighborhood do
 7:         calculate element Jacobian and local coordinates of quad-points
 8:         calculate derivatives of FEM shape functions at quad-points
 9:         $\boldsymbol{a} \leftarrow \mathcal{DP}(\boldsymbol{x})$ where $\mathcal{DP}(\boldsymbol{x})$ is defined by the PDE we are solving
10:         GLPderivs ← diff_WLS(aes_gvm, $\boldsymbol{a}$)
11:         for each node in neighborhood do
12:             if not Dirichlet BC node then
13:                 add integral to appropriate stiffness matrix entry
14:             else
15:                 subtract integral from load vector
16:             end if
17:         end for
18:         if AESFEM1 then
19:             calculate load vector over current element using FEM approximations
                for quad-points
20:         else
21:             calculate load vector over current element using GLP approximations
                for quad-points
22:         end if
23:     end for
24: end for
```

Algorithm 4 Constructing a GFD coefficient matrix

```
function: gfd
input: 1. $\boldsymbol{x}$, elem, opphfs, vh2f: mesh information
       2. $p$: desired degree for GFD functions
       3. isDBC: flags for Dirichlet boundary conditions
output: GFD matrix $\boldsymbol{K}$ and vector $\boldsymbol{b}$
 1: for each node without Dirichlet boundary conditions do
 2:     obtain neighborhood of node
 3:     calculate local parameterization and row weights for neighborhood
 4:     gfd_cvm ← initiate_CVM($\boldsymbol{x}_k$, $\boldsymbol{w}$, $p$, $\epsilon$)
 5:     $\boldsymbol{a} \leftarrow \mathcal{D}\boldsymbol{\mathcal{P}}(\boldsymbol{x})$ where $\mathcal{D}\boldsymbol{\mathcal{P}}(\boldsymbol{x})$ is defined by the PDE we are solving
 6:     GFDderivs ← diff_WLS(gfd_cvm, $\boldsymbol{a}$)
 7:     for each node in local neighborhood do
 8:         if not BC node then
 9:             enter value in matrix
10:         else
11:             subtract from RHS vector
12:         end if
13:     end for
14: end for
```

primarily for comparison with the AES-FEM algorithm. We can see that for GFD, we need to use Algorithm 2 to compute the weights once for each non-Dirichlet node in the mesh; that is, if there are n non-Dirichlet nodes, Algorithm 2 is called n times. For AES-FEM, for a given node, we use this algorithm once for every element containing that node. Thus if every node has a neighborhood of k elements and there are n non-Dirichlet nodes, we call the algorithm kn times. Therefore, the assembly time is longer for AES-FEM than for GFD, as we will see in Chapter 6.

Chapter 6

Numerical Results and Applications

In this chapter, we assess the accuracy, efficiency, and element-quality dependence of AES-FEM and compare with FEM and GFD, and we compare the runtimes for a variety of combinations of linear solvers and preconditioners. In Section 6.1, we focus on quadratic AES-FEM 1, quadratic AES-FEM 2, linear FEM, and quadratic GFD. We compare these four method because the sparsity pattern of the coefficient matrix is nearly identical for all the methods. The sparsity pattern determines the amount of storage necessary and also the computational cost of vector-matrix multiplication. In Section 6.2, we assess AES-FEM with quadratic, quartic, and sextic basis functions, and compare it against FEM with linear, quadratic and cubic basis functions. In Section 6.3, we discuss some commonly used linear solvers and preconditioners. Then we compare the runtimes of various pairs of preconditioners and linear solvers for solving the linear systems resulting from the Poisson equation and the convection-diffusion equation using AES-FEM and FEM.

The errors are calculated using the discrete L_2 and L_∞ norms. Let U denote the exact solution and let $\tilde{U}$ denote the numerical solution. Then, we calculate the norms as

$$L_2\,(\text{error}) = \left(\int_\Omega |\tilde{U} - U|^2 \partial\Omega\right)^{1/2} \quad \text{and} \quad L_\infty\,(\text{error}) = \max_i |\tilde{U} - U|. \quad (6.0.1)$$

On a series of meshes of different grid resolution, we calculate the average convergence rate as

$$\text{convergence rate } = -\log_2\left(\frac{\text{error on } m_c}{\text{error on } m_f}\right) \Bigg/ \log_2\left(\sqrt[d]{\frac{\text{nodes in } m_c}{\text{nodes in } m_f}}\right), \quad (6.0.2)$$

where d is the spacial dimension, m_c is the coarsest mesh, and m_f is the finest mesh.

6.1 Numerical Results of Quadratic AES-FEM

6.1.1 2D Results

In this section, we present the results of our 2-dimensional experiments with quadratic AES-FEM, linear FEM, and quadratic GFD. We use two different series of meshes, each series with 4 meshes. The first series of meshes (referred to collectively as "mesh series 1") is generated by placing nodes on a regular grid and then using MATLAB's Delaunay triangularization function to create the elements. The meshes range in size from 64×64 to 512×512 nodes. On the most refined mesh, the minimum angle is 45 degrees and the maximum angle is 90 degrees. The maximum aspect ratio is 1.41, where a triangle's aspect ratio is defined as the ratio of the length of longest edge to the length of the smallest edge. The second series of meshes (referred collectively as "mesh series 2") is generated by using Triangle

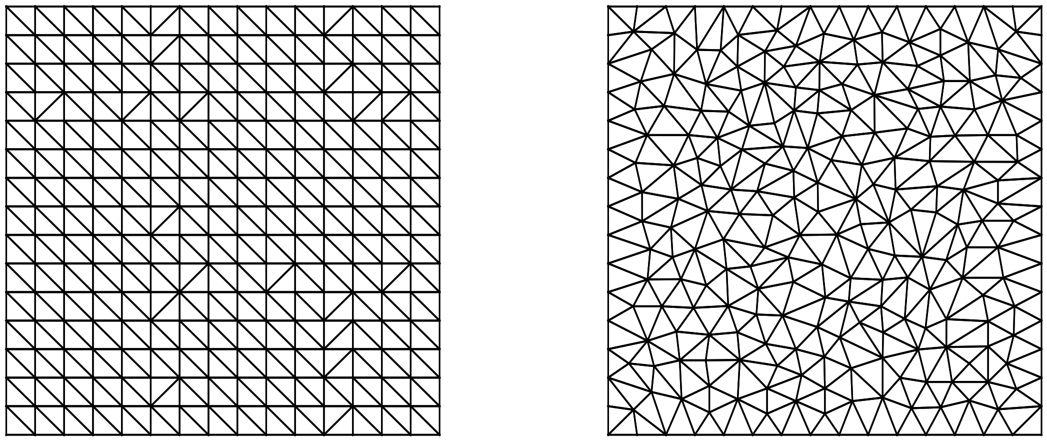


Figure 6.1: The mesh on the left is representative of the meshes used in series 1. The mesh on the right is representative of the meshes used in series 2. Note that the meshes above are coarser than the meshes used in computations so that the details can be seen clearly.

[78]. The number of nodes for each level of refinement is 4,103, 16,401, 65,655, and 262,597, respectively, approximately the same as those in series 1. On the most refined mesh, the maximum angle is 129.6 degrees and the minimum angle is 22.4 degrees. The maximum aspect ratio is 2.61. See Figure 6.1 for a visualization of the types of meshes used.

Poisson Equation

The first set of results we present is for the Poisson equation with Dirichlet boundary conditions on the unit square. That is,

$$-\nabla^2 U = \rho \quad \text{in } \Omega = [0,1]^2, \tag{6.1.1}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.1.2}$$

We consider the following three analytic solutions:

$$U_1 = 16x(1-x)y(1-y), \tag{6.1.3}$$

$$U_2 = \cos(\pi x)\cos(\pi y), \tag{6.1.4}$$

$$U_3 = \frac{1}{\sinh\pi\cosh\pi}\sinh(\pi x)\cosh(\pi y). \tag{6.1.5}$$

The Dirichlet boundary conditions are obtained from the given analytic solutions. The boundary conditions for U_1 are homogeneous and they are non-homogenous for U_2 and U_3 .

The L_∞ and L_2 norm errors for U_1 on mesh series 1 are displayed in Figure 6.2. One can see that the two graphs are fairly similar; this is true for U_2 and U_3 as well and thus we show only the L_∞ norm errors for these two problems; see Figure 6.3. GFD is the most accurate for U_1 and U_2 and AES-FEM 2 is the most accurate for U_3. FEM is the least accurate in all three cases.

For mesh series 2, the L_∞ and L_2 norm errors for U_1 can be seen in Figure 6.4 and the L_∞ norm errors for U_2 and U_3 can be seen in Figure 6.5. On this mesh series, AES-FEM 2 has the lowest error for U_1 and U_2. For U_3, the errors for GFD and AES-FEM 2 are very similar.

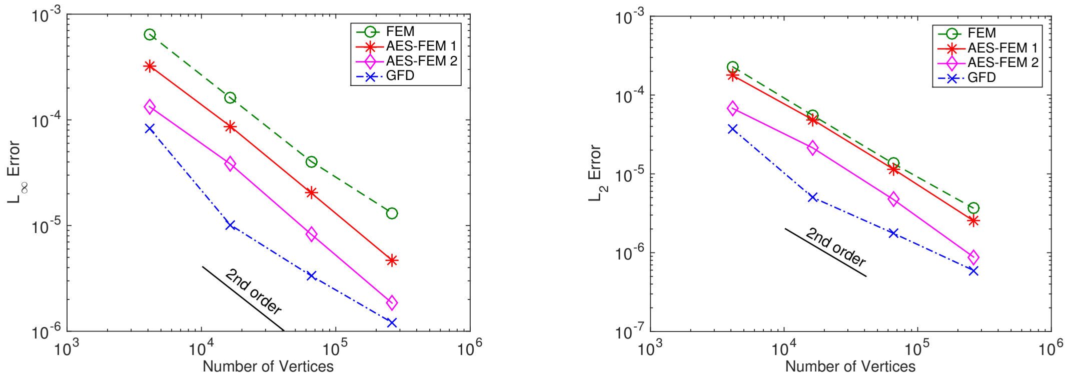


Figure 6.2: The errors for 2D Poisson equation on mesh 1 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).

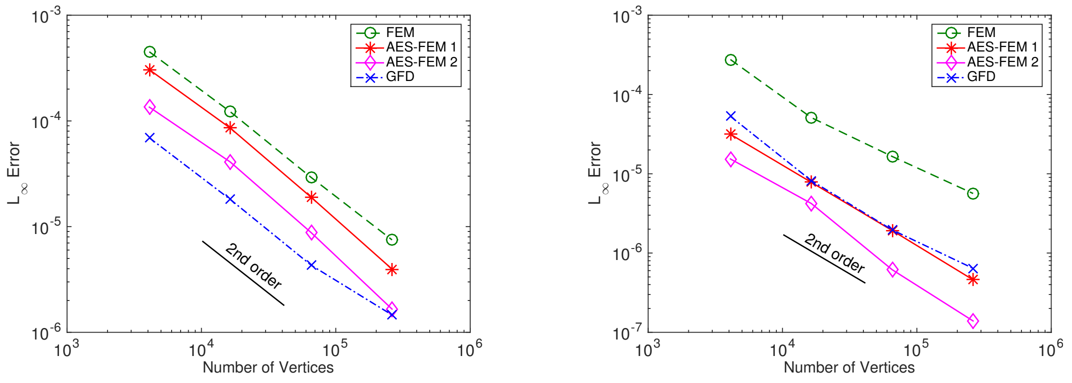


Figure 6.3: The L_∞ norm errors for the 2D Poisson equation on mesh 1 for U_2 (left) and U_3 (right).

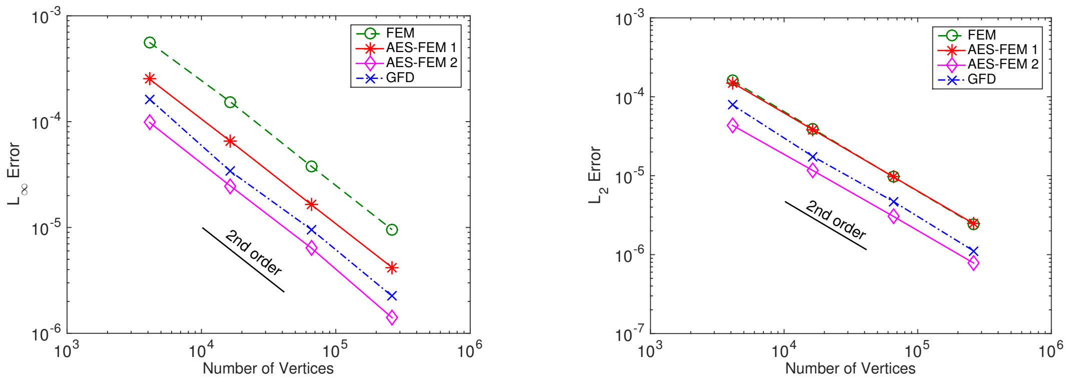


Figure 6.4: The errors for 2D Poisson equation on mesh 2 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).

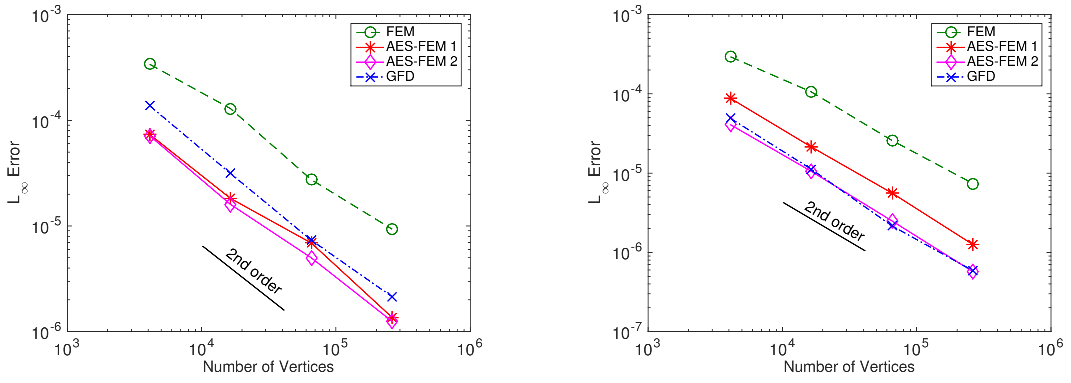


Figure 6.5: The L_∞ norm errors for the 2D Poisson equation on mesh 2 for U_2 (left) and U_3 (right).

Convection-Diffusion Equation

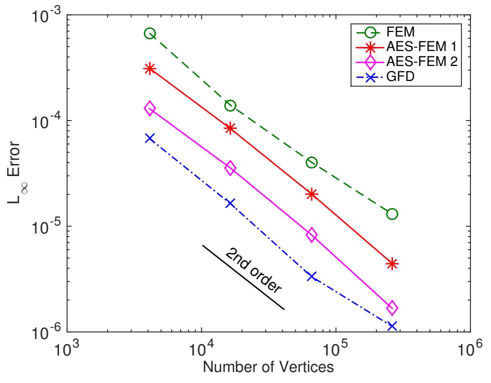
We consider the convection-diffusion equation with Dirichlet boundary conditions on the unit square. That is,

$$-\nabla^2 U + c \cdot \nabla U = \rho \quad \text{in } \Omega, \tag{6.1.6}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.1.7}$$

We take $\boldsymbol{c} = [1, 1]^T$ for all of our tests and we consider the same analytic solutions as for the Poisson equation. Again the boundary conditions are obtained from the given analytic solutions.

The L_∞ and L_2 norm errors obtained from the convection-diffusion equation on mesh series 1 with U_1 are presented in Figure 6.6. The L_∞ norm errors for U_2 and U_3 on mesh series 1 are in Figure 6.7. For all three problems, AES-FEM and GFD are both more accurate than linear FEM. For U_1, GFD is the most accurate. For U_2, the most accurate method is either AES-FEM 2 or GFD depending on the level of refinement. For U_3, AES-FEM 2 is the most accurate.



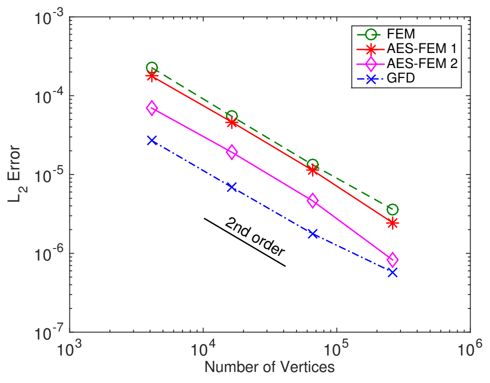
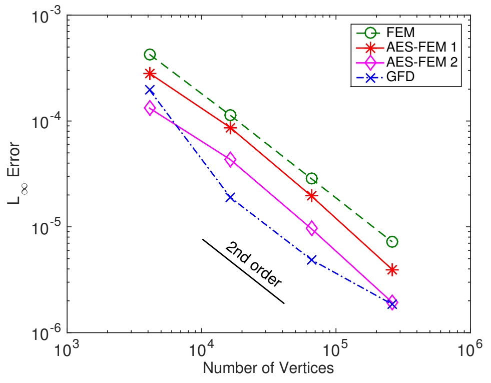


Figure 6.6: The errors for 2D convection-diffusion equation on mesh 1 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).



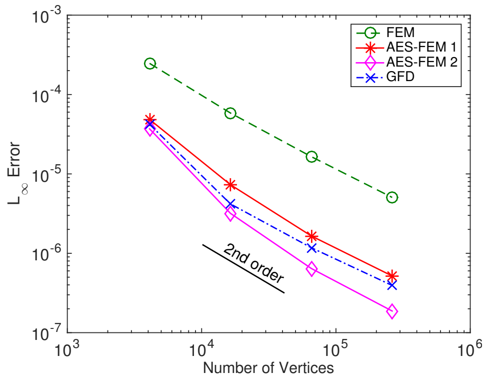


Figure 6.7: The L_∞ norm errors for the 2D convection-diffusion equation on mesh 1 for U_2 (left) and U_3 (right).

On mesh series 2, AES-FEM 2 is the most accurate for U_1, as can be seen in Figure 6.8. For U_2 AES-FEM 1 or AES-FEM 2 is the most accurate, and for U_3 GFD is the most accurate; see Figure 6.9. In all these cases, AES-FEM is more accurate than FEM.

Element-Quality Dependence Test

We test how FEM, AES-FEM, and GFD perform on a series of progressively worse meshes. We begin with the most refined mesh from mesh series 2. We select 6 of the

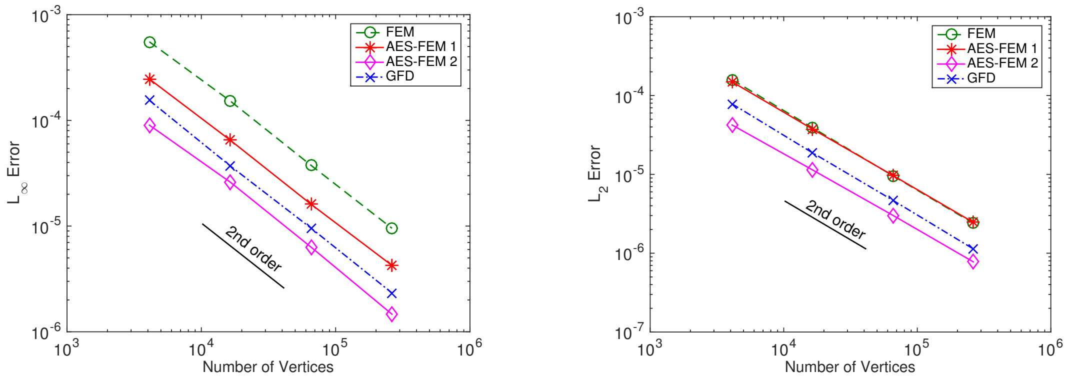


Figure 6.8: The errors for 2D convection-diffusion equation on mesh 2 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).

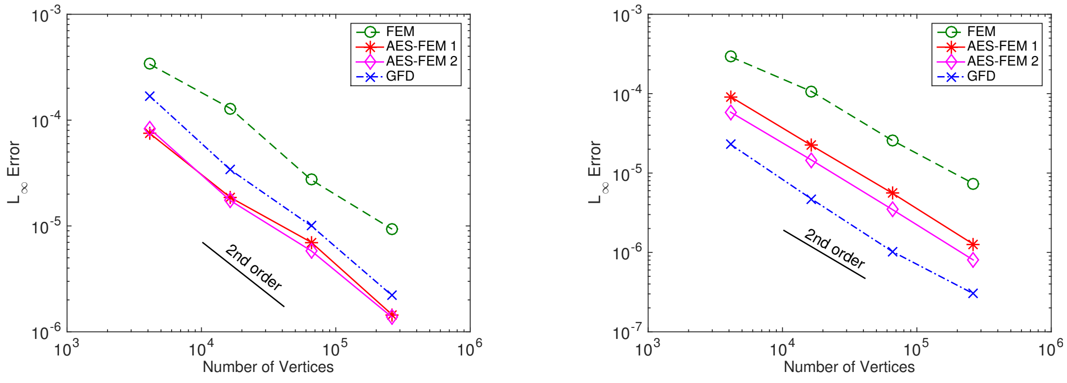


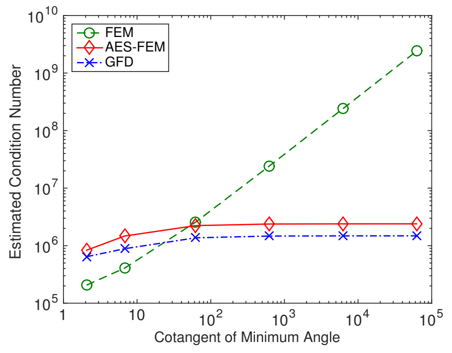
Figure 6.9: The L_∞ norm errors for the 2D convection-diffusion equation on mesh 2 for U_2 (left) and U_3 (right).

523,148 elements and incrementally move one of their nodes towards the opposite edge so as to create flatter triangles. We then solve the Poisson equation with the polynomial analytic solution U_1 in (6.1.3) and record the condition numbers of the coefficient and stiffness matrices and the numbers of iterations required for the solver to converge. Since the stiffness matrix is the same for AES-FEM 1 and AES-FEM 2, the results are just labeled as AES-FEM. We use the conjugate gradient method with incomplete Cholesky preconditioner for FEM and we use GMRES with incomplete LU preconditioner for AES-FEM and GFD. The tolerance for the solvers is 10^{-8} and the drop tolerance for the preconditioners is 10^{-3}. As a measure of the mesh quality, we consider the cotangent of the minimum angle in the mesh; as the minimum angle tends to zero, the cotangent tends towards infinity. For very small angles, the cotangent of the angle is approximately equal to the reciprocal of the angle. We estimate the condition numbers using the MATLAB function `condest`, which computes a lower bound for the 1-norm condition number.

The worse the mesh quality, the higher the condition number of the stiffness matrix resulting from FEM. In contrast, the condition numbers of the GFD coefficient matrix and stiffness matrix from AES-FEM remain almost constant. As the condition number for FEM rises, so does the number of iterations required for the solver to converge, from 102 to 128. The numbers of iterations required to solve the equation for AES-FEM and GFD remain constant, at 73 and 74, respectively. We show the results for 6 meshes. Preconditioned conjugate gradient stagnates when trying to solve the FEM linear system from the 7th mesh, where the minimum angle is approximately 9.1×10^{-5} degrees. Solving the AES-FEM and GFD linear systems from the 7th mesh requires the same numbers of iterations as the other meshes, 73 and 74 respectively. See Figure 6.10 for a comparison of the condition numbers

Table 6.1: Errors in L_2 norm for FEM, AES-FEM 1, AES-FEM 2, and GFD for u_1 on a series of meshes with progressively worse mesh element quality.

	FEM	AES-FEM 1	AES-FEM 2	GFD
Mesh 1	2.42×10^{-6}	2.47×10^{-6}	7.82×10^{-7}	1.11×10^{-6}
Mesh 2	2.42×10^{-6}	2.47×10^{-6}	7.83×10^{-7}	1.10×10^{-6}
Mesh 3	2.42×10^{-6}	2.47×10^{-6}	7.82×10^{-7}	1.10×10^{-6}
Mesh 4	2.42×10^{-6}	2.81×10^{-6}	1.19×10^{-6}	1.10×10^{-6}
Mesh 5	2.42×10^{-6}	2.81×10^{-6}	1.19×10^{-6}	1.10×10^{-6}
Mesh 6	2.42×10^{-6}	2.81×10^{-6}	1.19×10^{-6}	1.10×10^{-6}



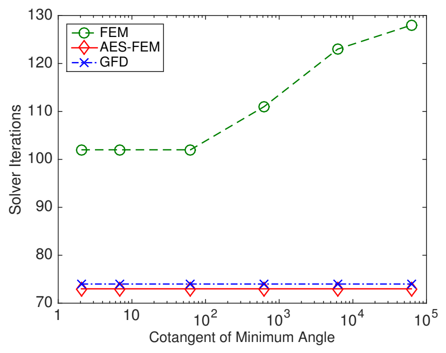


Figure 6.10: The condition numbers of the stiffness matrices for FEM and Adaptive Extended Stencil (AES)-FEM and the coefficient matrix for generalized finite difference (GFD) (left) and the numbers of solver iterations (right). Solvers used are preconditioned conjugate gradient (PCG) for FEM and preconditioned generalized minimal residual (GMRES) for AES-FEM and GFD.

and the numbers of iterations.

The errors for both AES-FEM 1 and AES-FEM 2 rose slightly between the 3rd and the 4th mesh; the errors were then constant for the rest of the meshes. The errors for FEM remained constant and the errors for GFD remained nearly constant over the 6 meshes. AES-FEM 1, AES-FEM 2 and GFD converge on the 7th mesh in the series with the same errors as on Mesh 6, whereas for FEM the solver stagnates. See Table 6.1 for specific errors.

Efficiency

We compare the runtimes for the four methods: AES-FEM 1, AES-FEM 2, FEM, and GFD. We consider the convection-diffusion equation on the most refined mesh of series 2 with the polynomial analytic solution U_1 for this runtime experiment. We decompose the total time into 4 subcategories: *Initialization,* which includes the time to load the mesh and the time to assign the boundary conditions and problem values; *Assembly,* which includes the time to build the stiffness matrix and load vector; *Preconditioner,* which is the time it takes to construction the matrix preconditioner using incomplete LU factorization with a drop tolerance of 10^{-3}; and *Solver,* which is the amount of time for solving the preconditioned system using GMRES with a tolerance of 10^{-8}. See Figure 6.11 for the comparison. The initialization time is minuscule compared to the other categories and is not visible in the figure. FEM requires 76 iterations of GMRES to converge, AES-FEM 1 and AES-FEM 2 both require 74 iterations, and GFD requires 75 iterations.

One can see that FEM has an advantage when it comes to efficiency on the same mesh. In terms of total runtime, it is about 2.1 times faster than AES-FEM 1, 2.1 times faster than AES-FEM 2, and 1.8 times faster than GFD. In terms of assembly time, FEM is about 6.1 times faster than AES-FEM 1, 7.7 times faster than AES-FEM 2, and 2.2 times faster than GFD. Comparing the assembly time of AES-FEM and GFD, we see that GFD is approximately 3.2 and 3.5 times faster than AES-FEM 1 and AES-FEM 2, respectively.

In 2D, assembling the load vector using FEM basis functions (AES-FEM 1) saves some time compared to using GLP basis functions (AES-FEM 2). AES-FEM 1 offers a savings of approximately 1.1 seconds or, in other words, a 10.3%

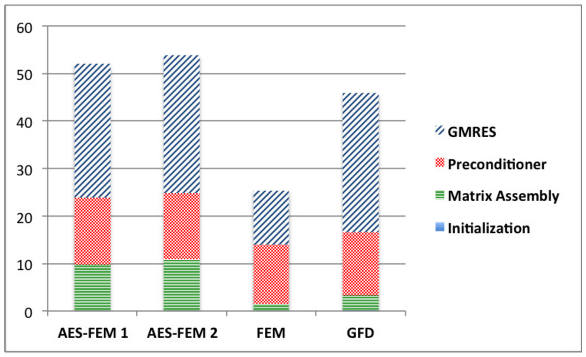


Figure 6.11: Runtimes for a 2D convection-diffusion equation on the most refined mesh in series 2.

reduction of assembly time and a 3.5% reduction of total time compared to AES-FEM 2. We will see in the next section that the efficiency of these two methods varies more in 3D.

However, in terms of error versus runtime, AES-FEM is competitive with, and often is more efficient than, the classical FEM with linear basis functions. In Figure 6.12, we compare the L_∞ norm errors versus runtimes for the four methods on mesh series 2 for the Poisson equation and the convection-diffusion equation with the analytic solution equal to U_2. For the Poisson equation, all four methods are very similar, with AES-FEM 1 being slightly more efficient for finer meshes. For the convection-diffusion equation, AES-FEM 1 and AES-FEM 2 are approximately the same in terms of efficiency and are the most efficient. GFD is also more efficient than FEM.

6.1.2 3D Results

In this section, we present the results from the 3D experiments with quadratic AES-FEM, linear FEM, and quadratic GFD. We consider the Poisson equation and the

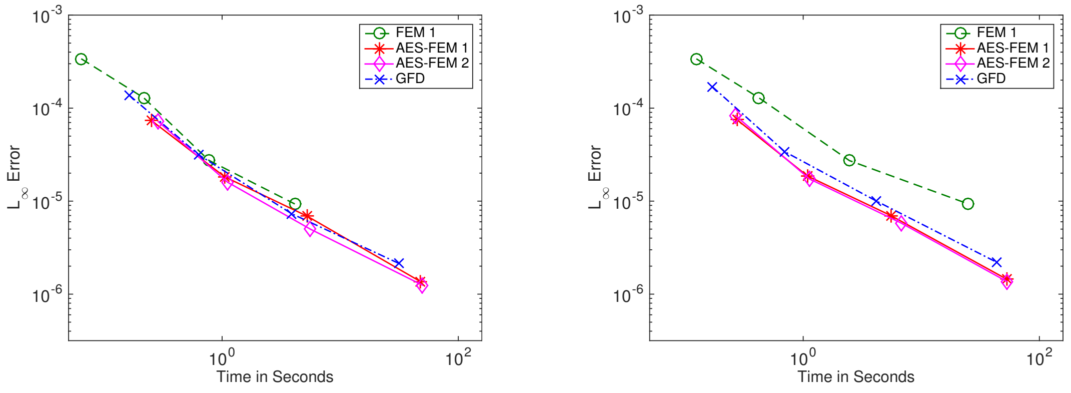


Figure 6.12: L_∞ norm errors versus runtimes for a 2D Poisson equation (left) and convection-diffusion equation (right) on mesh series 2. Lower is better.

convection-diffusion equation in 3D. We test three problems for each equation on two different series of meshes, each with four levels of refinement. The first series of meshes (referred to collectively as "mesh series 1") is created by placing nodes on a regular grid and using MATLAB's Delaunay triangularization to create the elements. The meshes in series 1 range from $8 \times 8 \times 8$ nodes to $64 \times 64 \times 64$ nodes. The minimum dihedral angle in the most refined mesh of series 1 is 35.2 degrees and the maximum dihedral angle is 125.2 degrees. The maximum aspect ratio is 4.9, where the aspect ratio of a tetrahedron is defined as the ratio of the longest edge length to the smallest height. The second series of meshes (referred to collectively as "mesh series 2") is created using TetGen [80]. The number of nodes in mesh series 2 for each level of refinement is 509, 4,080, 32,660, and 261,393, which is approximately the same as the meshes in series 1. The minimum dihedral angle of the most refined mesh in series 2 is 6.7 degrees and the largest dihedral angle is 165.5 degrees. The largest aspect ratio is 15.2.

Poisson Equation

We first consider the Poisson equation with Dirichlet boundary conditions on the unit cube. That is,

$$-\nabla^2 U = \rho \quad \text{in } \Omega, \tag{6.1.8}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.1.9}$$

where $\Omega = [0,1]^3$. We consider three different analytic solutions listed below.

$$U_1 = 64x\,(1-x)\,y\,(1-y)\,z\,(1-z)\,, \tag{6.1.10}$$

$$U_2 = \cos(\pi x)\cos(\pi y)\cos(\pi z), \tag{6.1.11}$$

$$U_3 = \frac{1}{\sinh \pi \cosh \pi \cosh \pi} \sinh(\pi x)\cosh(\pi y)\cosh(\pi z). \tag{6.1.12}$$

The Dirichlet boundary conditions are derived from the analytic solutions. They are homogeneous for U_1 and non-homogeneous for U_2 and U_3.

The L_∞ and L_2 norm errors for the Poisson equation on mesh series 1 for U_1 can be seen in Figure 6.13 and the L_∞ norm errors for U_2 and U_3 can be seen in Figure 6.14. GFD is the most accurate for U_1 and U_2. AES-FEM 2 is the most accurate for U_3.

For mesh series 2, AES-FEM 2 is the most accurate and is close to an order of magnitude more accurate than FEM. See Figure 6.15 for the L_∞ and L_2 norm errors for U_1 and Figure 6.16 for the L_∞ norm errors for U_2 and U_3.

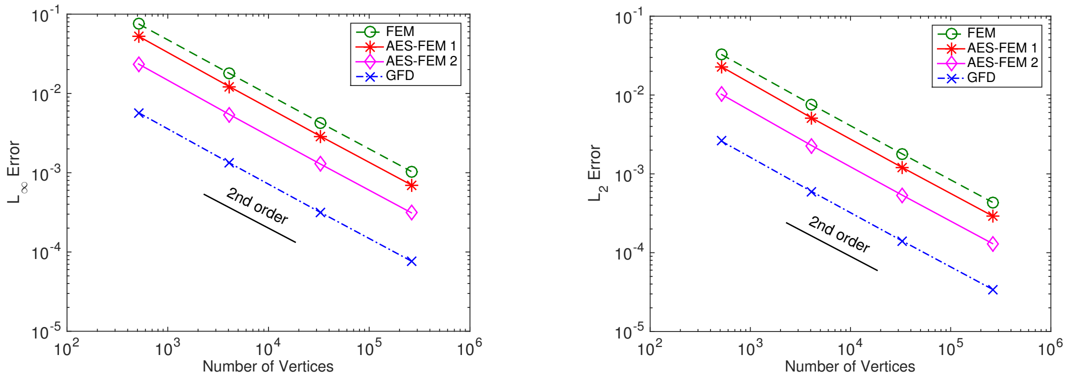


Figure 6.13: The errors for 3D Poisson equation on mesh 1 with U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).

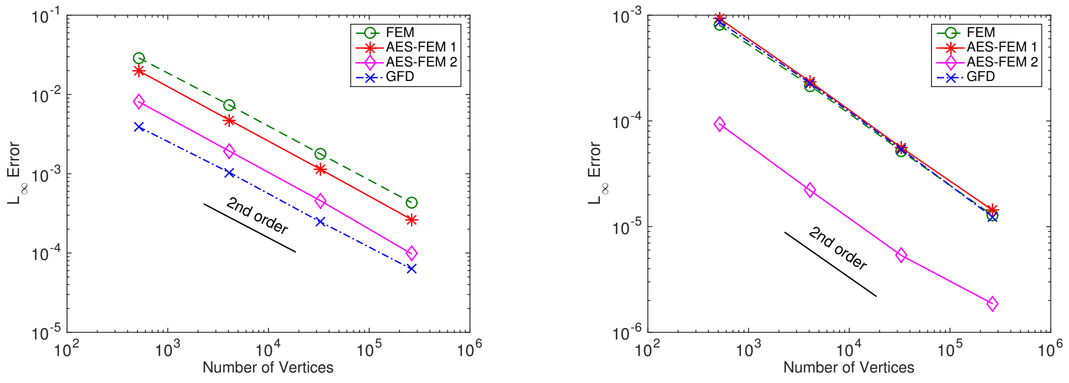


Figure 6.14: The L_∞ norm errors for the 3D Poisson equation on mesh 1 for U_2 (left) and U_3 (right).

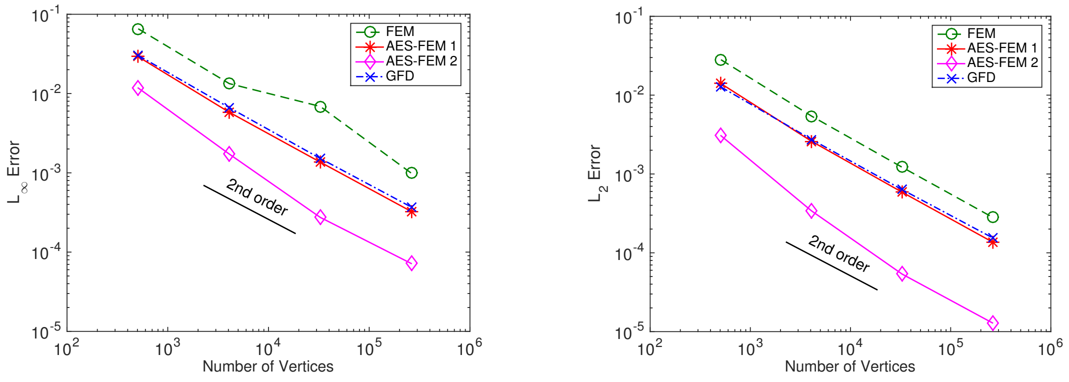


Figure 6.15: The errors for 3D Poisson equation on mesh 2 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).

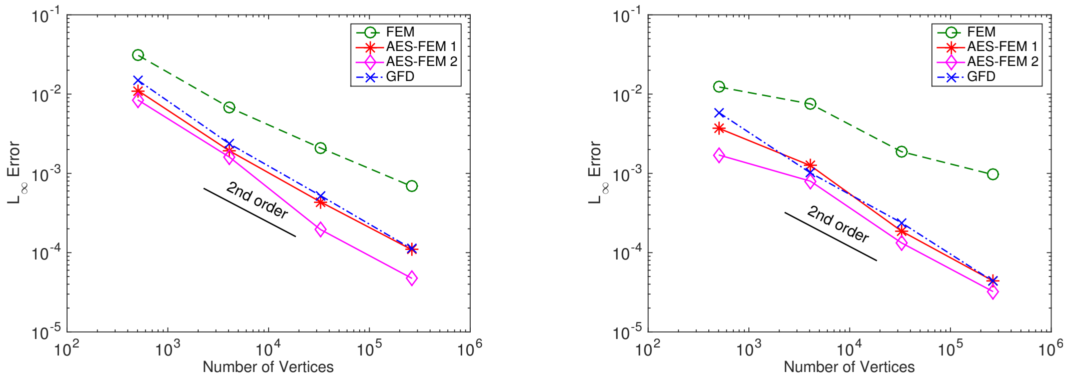


Figure 6.16: The L_∞ norm errors for the 3D Poisson equation on mesh 2 for U_2 (left) and U_3 (right).

Convection-Diffusion Equation

We consider the convection-diffusion equation with Dirichlet boundary conditions on the unit cube, $\Omega = [0,1]^3$.

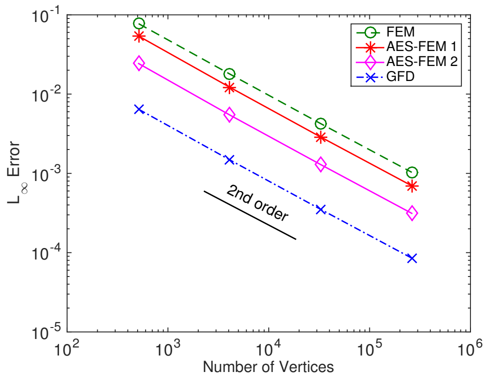
$$-\nabla^2 U + c \cdot \nabla U = \rho \quad \text{in } \Omega, \tag{6.1.13}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.1.14}$$

We take $\boldsymbol{c} = [1,1,1]^T$ and we consider the same analytic solutions as in the previous section. Again, the Dirichlet boundary conditions are derived from the analytic solutions U_1, U_2, and U_3.

The L_∞ and L_2 norm errors for the 3D convection-diffusion equation on mesh series 1 for U_1 can be seen in Figure 6.17, see Figure 6.18 for the L_∞ norm errors on mesh series 1 for U_2 and U_3. As with the Poisson equation, GFD is the most accurate for U_1 and u_2. For u_3, either GFD or AES-FEM 2 is the most accurate depending on the level of refinement.

On mesh series 2, AES-FEM 2 is the most accurate for U_1 and U_2, as can be



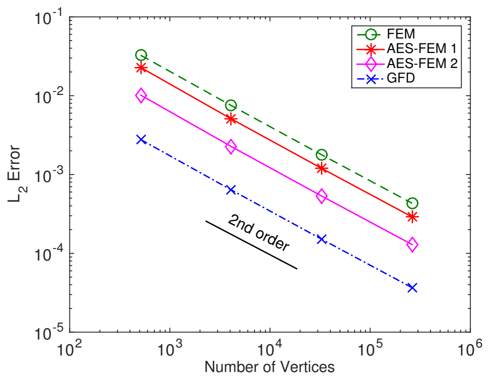
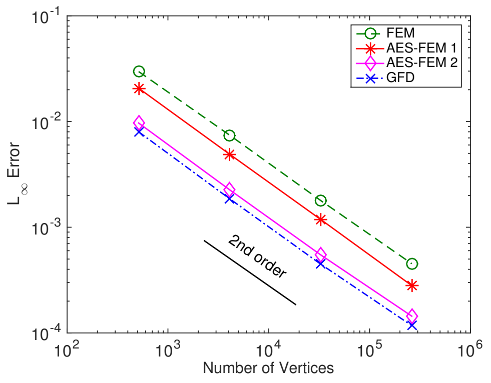


Figure 6.17: The errors for 3D convection-diffusion equation on mesh 1 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).



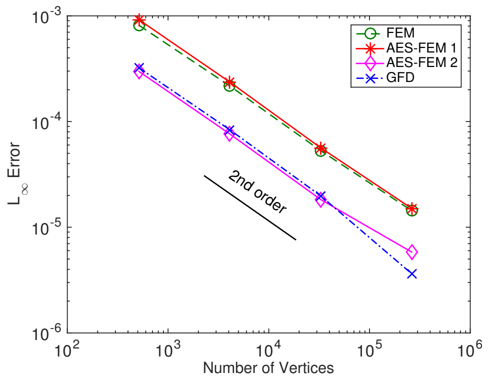
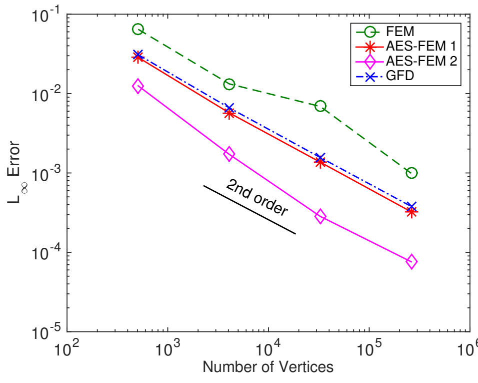


Figure 6.18: The L_∞ norm errors for the 3D convection-diffusion equation on mesh 1 for U_2 (left) and U_3 (right).

seen in Figure 6.19 and the left panel of Figure 6.20. For U_3, either GFD or AES-FEM 2 is the most accurate, as can be seen in the right panel of Figure 6.20. Similar to 2D results, AES-FEM is more accurate than FEM in all these cases.

Element-Quality Dependence Test

We test how FEM, AES-FEM, and GFD perform on a series of meshes with progressively worse element shape quality. We begin with the most refined mesh from



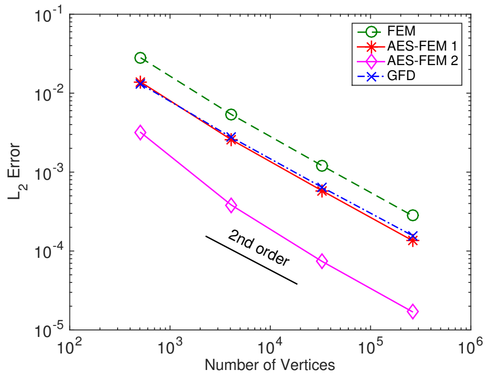
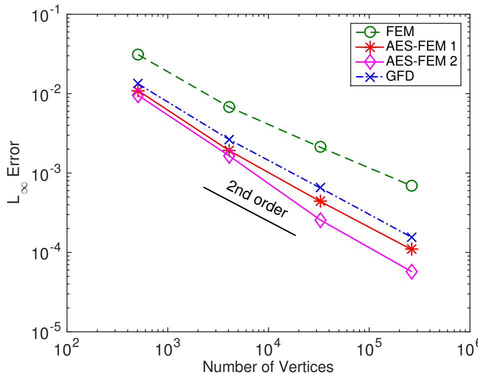


Figure 6.19: The errors for 3D convection-diffusion equation on mesh 2 for U_1. The errors were computed using the L_∞ norm (left) and the L_2 norm (right).



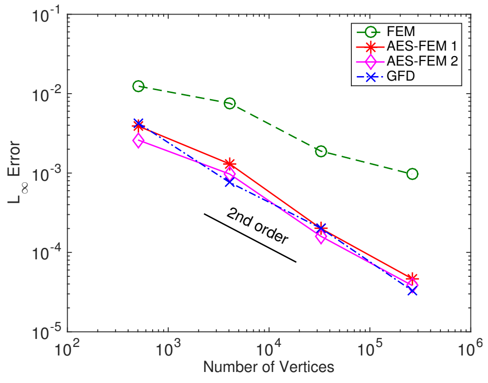


Figure 6.20: The L_∞ errors for the 3D convection-diffusion equation on mesh 2 for U_2 (left) and U_3 (right).

mesh series 1. We select 69 out of the 1,500,282 elements and incrementally move one of their nodes towards the opposite side so as to create sliver tetrahedra. We then solve the Poisson equation with the polynomial analytic solution U_1 in (6.1.10) and record the condition numbers of the coefficient and stiffness matrices and the numbers of iterations required for the solver to converge. We use the conjugate gradient method with Gauss-Seidel preconditioner for FEM and we use GMRES with Gauss-Seidel preconditioner for AES-FEM and GFD. We use a tolerance of 10^{-5} for both solvers. As a measure of the mesh quality, we consider the cotangent of the minimum dihedral angle in the mesh; as the minimum angle tends to zero, the cotangent tends towards infinity. For very small angles, the cotangent of the angle is approximately equal to the reciprocal of the angle. We estimate the condition numbers using the MATLAB function `condest`, which computes a lower bound for the 1-norm condition number.

The worse the mesh quality, the higher the condition number of the stiffness matrix resulting from FEM. The condition numbers of the stiffness matrix from AES-FEM and the coefficient matrix from GFD remain almost constant. As the condition number of the matrix rises so does the number of iterations required for the solver to converge. For FEM the number of iterations increases from 69 for the best mesh to 831 for the most deformed mesh. The numbers of iterations for AES-FEM and GFD remain almost constant, increasing from 56 to 59 and from 56 to 60, respectively. See Figure 6.21 for a comparison of the condition numbers and iteration counts of the solvers.

For each of the four methods, the errors were nearly constant over the series of meshes. For FEM, the L_2 error on the 1st mesh was 4.36×10^{-4} and on the 6th mesh, the error was 4.37×10^{-4}. For AES-FEM 1, the L_2 error on all the meshes

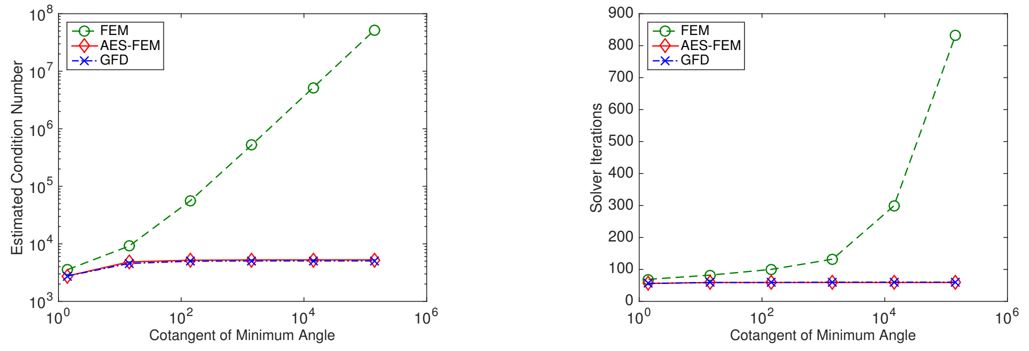


Figure 6.21: The condition numbers of the stiffness matrices for FEM and AES-FEM and the coefficient matrix for GFD (left) and the numbers of solver iterations (right). Solvers used are preconditioned CG for FEM and preconditioned GMRES for AES-FEM and GFD.

was 2.92×10^{-4}. For AES-FEM 2, the L_2 error on all the meshes was 1.30×10^{-4}. For GFD, the L_2 error on the 1st mesh was 3.43×10^{-5} and on the 6th mesh, the error was 3.40×10^{-5}.

Efficiency

We compare the runtimes of the four methods for solving the convection-diffusion equation with the polynomial analytic solution U_1 on the most refined mesh of series 2. As in 2D, the total time is decomposed into 4 subcategories: *Initialization, Assembly*, *Preconditioner*, and *Solver*. The preconditioner used is incomplete LU with a drop tolerance of 10^{-1}. GMRES with a tolerance of 10^{-8} is used as the solver. AES-FEM 1 and AES-FEM 2 each require 142 iterations to converge, FEM requires 128 iterations, and GFD requires 138 iterations. The majority of the time is spent assembling the matrix and solving the system. See Figure 6.22 for the comparison.

As in 2D, FEM is the most efficient method on a given mesh. Overall, the

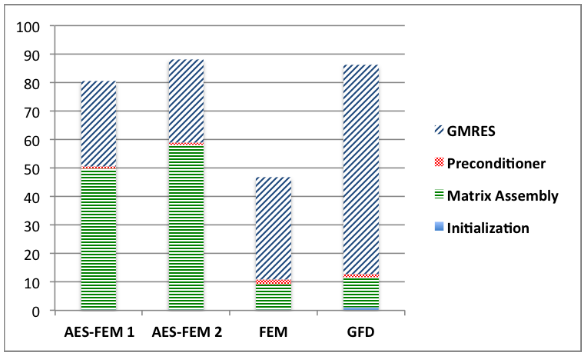
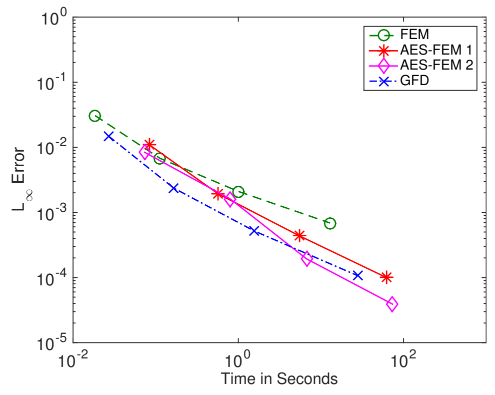


Figure 6.22: Runtimes for a 3D convection-diffusion equation on the most refined mesh in series 2.

total runtime of FEM is approximately 1.7 times faster than AES-FEM 1, 1.9 times faster than AES-FEM 2, and 1.8 times faster than GFD. The assembly of FEM is approximately 5.6 times faster than AES-FEM 1, 6.5 times faster than AES-FEM 2, and 1.2 times faster than GFD.

In 3D, the difference of assembling the load vector using FEM basis functions (AES-FEM 1) versus using GLP basis functions (AES-FEM 2) is more pronounced than in 2D. The assembly in AES-FEM 1 is 8.3 seconds shorter than that of AES-FEM 2. This means the assembly of AES-FEM 1 uses 14.4% less time than that of AES-FEM 1 and the total runtime is 8.7% shorter.

However, similar to 2D, AES-FEM is competitive with, and most of time more efficient than, the classical FEM with linear basis functions in terms of error versus runtime. Figure 6.23 shows the L_∞ norm errors versus runtimes for the four methods on mesh series 2 for the Poisson equation and the convection-diffusion equation for u_2. For the Poisson equation, GFD is more efficient on coarser meshes and AES-FEM 2 is more efficient for finer meshes. For the convection-diffusion equation, GFD is more efficient on smaller meshes and AES-FEM 2 is more efficient for finer meshes. AES-FEM 1 is also more efficient than FEM.



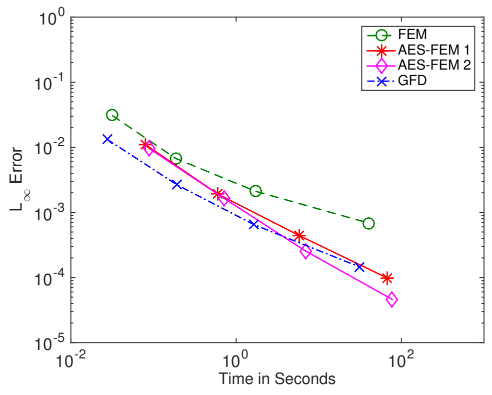


Figure 6.23: L_∞ norm errors versus runtimes for a 3D Poisson equation (left) and convection-diffusion equation (right) on mesh series 2. Lower is better.

6.2 Numerical Results of High-Order AES-FEM

In this section, we assess the accuracy, efficiency, and element-quality dependence of AES-FEM with quadratic, quartic, and sextic basis functions, and compare it against FEM with linear, quadratic and cubic basis functions.

6.2.1 2D Results

We first assess AES-FEM in 2D over the unit square and the unit disc, which are representative for geometries with flat and curved boundaries, respectively. We triangulated the domains using Triangle [78] for linear meshes and using Gmsh [36] for quadratic and cubic meshes. See Figure 6.24 for some examples meshes with linear elements, which are representative in terms of mesh quality but are coarser than those used in actual computations. The numbers of nodes for the unit square range from 1,027 to 146,077, and those for the unit disc range from 544 to 79,417. Since isoparametric FEM requires good mesh quality, we ensured that these meshes all have good element shapes for our comparative study: For linear

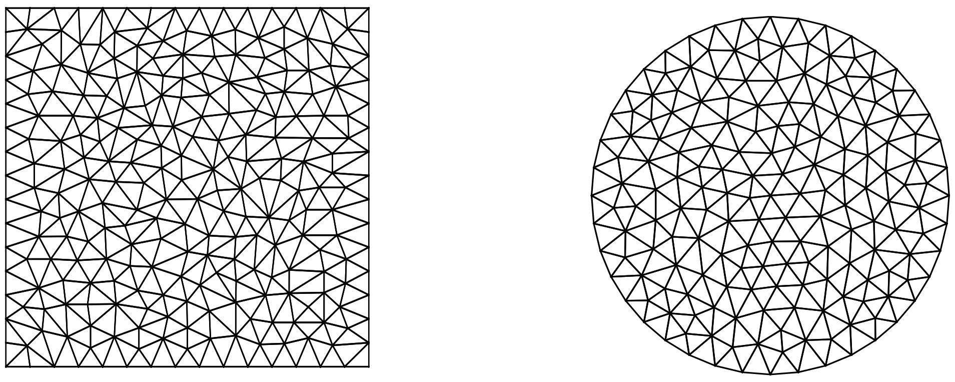


Figure 6.24: Example 2D meshes with linear elements.

meshes, the minimum angle is 24.04 degrees and the maximum angle is 128.17 degrees; for high-order meshes, all elements have positive Jacobians everywhere.

We consider the Poisson equation and convention-diffusion equation. For both cases, we use GMRES with the ILU preconditioner to solve the linear systems arising from AES-FEM. For FEM, we use conjugate gradient (CG) with incomplete Cholesky as the preconditioner for the Poisson equation, and use GMRES with ILU for the convection-diffusion equation. To demonstrate the accuracy of high-order methods, we set the tolerance of the iterative solvers to 10^{-12}. The drop tolerance for the incomplete factorization is set as 10^{-4} by default, unless otherwise noted.

Poisson Equation

We first present results for the Poisson equation with Dirichlet boundary conditions on the unit square and on the unit disc. That is,

$$-\nabla^2 U = \rho \quad \text{in } \Omega, \tag{6.2.1}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.2.2}$$

For the unit square $\Omega = [0,1]^2$, we consider the following three analytic solutions:

$$U_1 = 16x^3(1-x^3)y^3(1-y^3), \tag{6.2.3}$$

$$U_2 = \cos(\pi x)\cos(\pi y), \tag{6.2.4}$$

$$U_3 = \frac{\sinh(\pi x)\cosh(\pi y)}{\sinh\pi\cosh\pi}. \tag{6.2.5}$$

For the unit disc $\Omega = \{(x,y)|x^2+y^2 \le 1\}$, we consider U_3 and also

$$U_4 = \cos\left(\frac{\pi}{2}\left(x^2+y^2\right)\right). \tag{6.2.6}$$

For each problem, the right-hand side ρ and the Dirichlet boundary condition g are obtained from the given analytic solutions. For all the cases, the iterative solvers converged to the desired tolerance for AES-FEM. For FEM, the solver stagnated for the finest meshes in some cases without achieving the specified tolerance, even after we reduced the drop tolerance to 10^{-6} in incomplete Cholesky. However, the resulting errors were small enough not to affect the comparison qualitatively.

Figure 6.25 shows the L_∞ and L_2 norm errors for U_1 on the unit square. The L_2 norm errors for U_2 and U_3 on the unit square and for U_3 and U_4 on the unit disc are shown in Figures 6.26 and 6.27, respectively. In all cases, quadratic AES-FEM and linear FEM have similar errors, and quartic AES-FEM has similar or better results compared to cubic FEM. Both of the above pairs have similar sparsity patterns and similar numbers of nonzeros in the coefficient matrices. Furthermore, sextic AES-FEM is far more accurate than all the other methods, achieving sixth-order accuracy despite the use of linear elements. This result confirms our accuracy analysis in Section 4.3 for 2D problems.

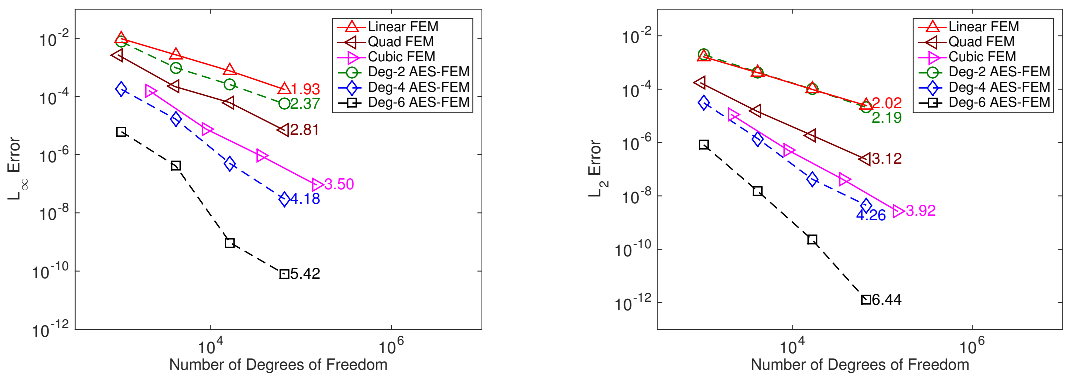


Figure 6.25: The errors for 2D Poisson equation on the unit square for U_1 in L_∞ (left) and L_2 norms (right). The number to the right of each curve indicates the average convergence rate.

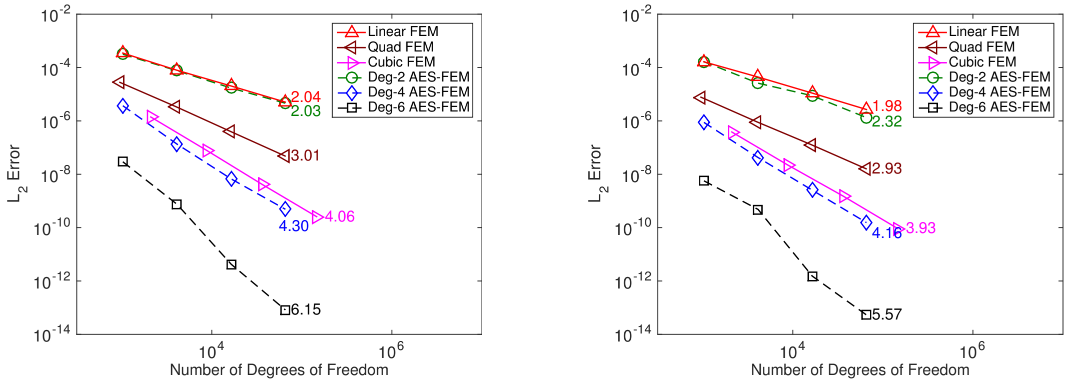


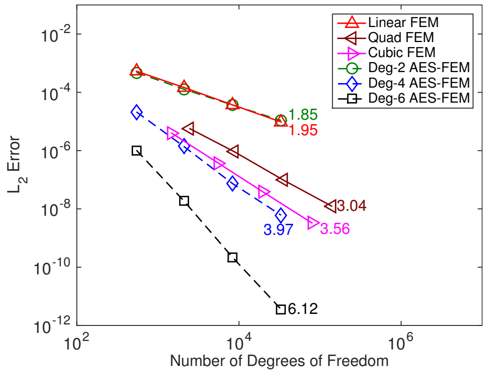
Figure 6.26: The L_2 norm errors for 2D Poisson equation on the unit square for U_2 (left) and U_3 (right).

Convection-Diffusion Equation

In our second example, we consider the time-independent convection-diffusion equation with Dirichlet boundary conditions, that is,

$$-\nabla^2 U + \boldsymbol{c} \cdot \boldsymbol{\nabla} U = \rho \quad \text{in } \Omega, \tag{6.2.7}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.2.8}$$



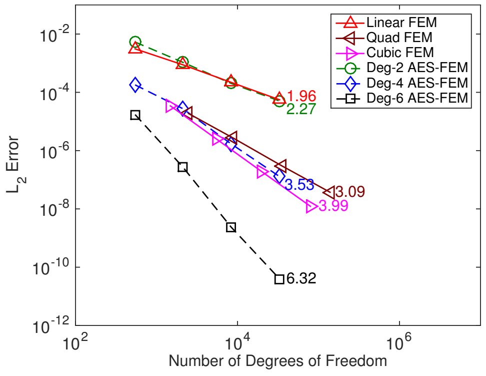


Figure 6.27: The L_2 norm errors for 2D Poisson equation on the unit disc for U_3 (left) and U_4 (right).

We take $\boldsymbol{c} = [1,1]^T$ for all of our tests, and we consider the same analytic solutions over the unit square and on the unit disc as for the Poisson equation.

Figure 6.28 shows the L_∞ and L_2 norm errors for U_1 on the unit square. The L_2 norm errors for U_2 and U_3 on the unit square and for U_3 and U_4 on the unit disc are shown in Figures 6.29 and 6.30, respectively. Similar to the Poisson equation, quadratic AES-FEM has similar convergence rate as linear FEM, but slightly lower errors. Quartic AES-FEM is more accurate than cubic FEM in all cases, and sextic AES-FEM again delivers superior accuracy, achieving about sixth-order convergence.

Assessment of Element-Quality Dependence

To assess the dependence of AES-FEM and FEM on mesh quality, we use a series of meshes for the unit square with progressively worse element quality, which we obtain by distorting a good-quality mesh. For AES-FEM and linear FEM, we use a mesh with 130,288 elements and 65,655 nodes and distort four elements by moving

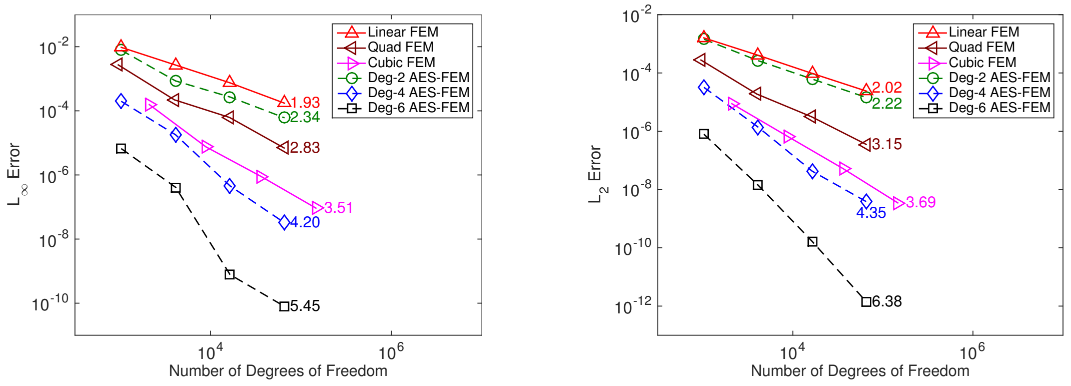


Figure 6.28: The errors for 2D convection-diffusion equation on the unit square for U_1 in the infinity norm (left) and the L_2 norm (right).

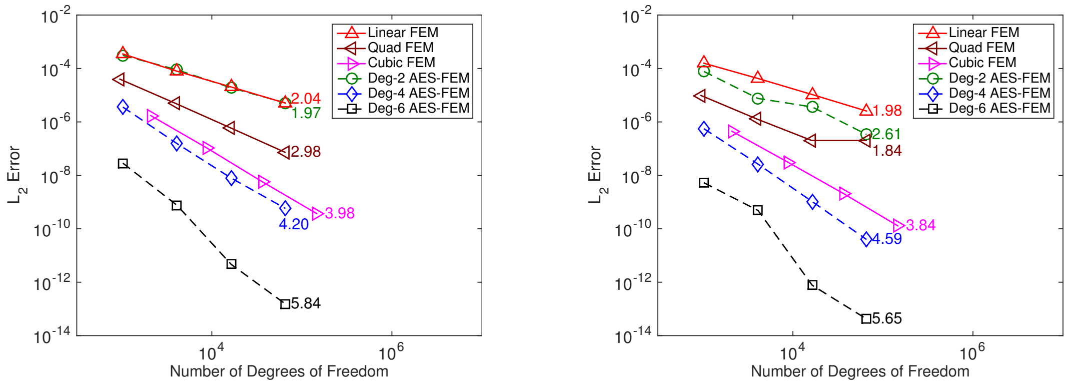


Figure 6.29: The L_2 norm errors for 2D convection-diffusion equation on the unit square for U_2 (left) and U_3 (right).

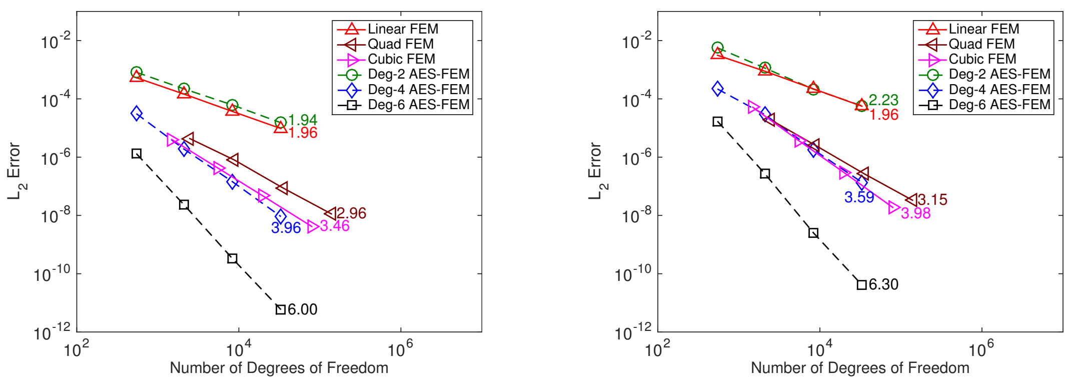
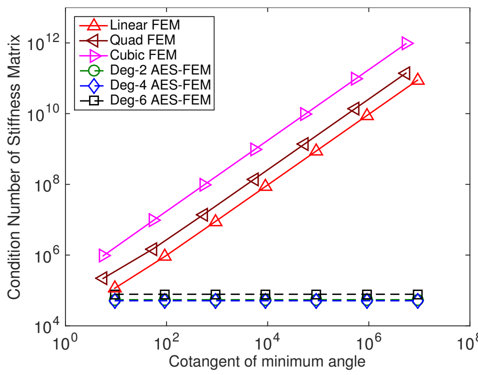


Figure 6.30: The L_2 norm errors for 2D convection-diffusion equation on the unit disc for U_3 (left) and U_4 (right).



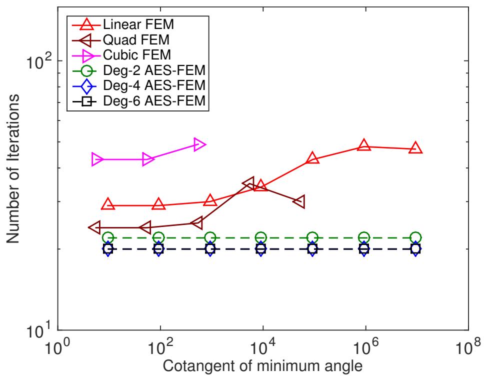


Figure 6.31: Left: the lower bound of the condition numbers of the stiffness matrices of FEM and AES-FEM method on a series of progressive worse meshes in 2D. Right: the number of iterations required for the iterative solver to converge when solving the corresponding systems.

one vertex of each of these elements incrementally towards its opposite edge. For quadratic FEM, we use a mesh with 32,292 elements and 65,093 nodes and distort a single element by moving one vertex and its adjacent mid-edge nodes. For cubic FEM, we use a mesh with 32,292 elements and 146,077 nodes, and distort a single element by moving one vertex and its adjacent mid-edge nodes. On each distorted mesh, we solve the Poisson equation with the exact solution U_2.

Figure 6.31 shows the condition numbers of the stiffness matrices of FEM and AES-FEM and the numbers of iterations required to solve the linear systems. It can be seen that the condition numbers of FEM increase inversely proportional to the minimum angle, while the condition numbers of AES-FEM remain constant. In terms of the linear solver, the numbers of iteration increase significantly for linear FEM as the mesh is distorted, and preconditioned CG fails for poorly-shaped quadratic and cubic meshes due to nonpositive pivot during incomplete Cholesky or due to stagnation of CG. In contrast, the number of iterations remains constant for AES-FEM, independently of the element quality.

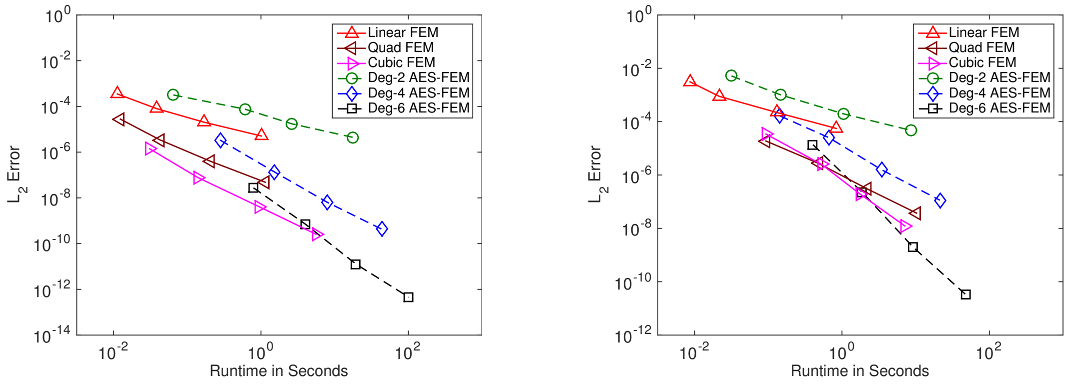


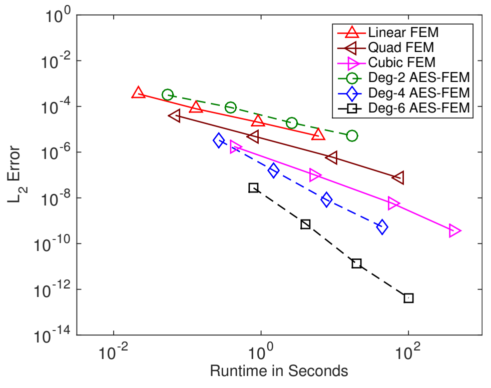
Figure 6.32: The errors verses runtime for 2D Poisson equation on the unit square for exact solution U_2 (left) and on the unit disc for exact solution U_4 (right).

Efficiency

To compare the efficiency of AES-FEM and FEM, Figures 6.32 and 6.33 show the errors versus runtimes for the 2D Poisson equation and convection-diffusion equation, respectively. It is evident that for the convention-diffusion equation, both quartic and sextic AES-FEM outperform cubic FEM over relatively fine meshes. For the Poisson equation, sextic AES-FEM outperforms cubic FEM over finer meshes. We expect AES-FEM will perform even better with further optimization of its implementation.

6.2.2 3D Results

We now assess AES-FEM in 3D over the unit cube and the unit ball, which are representative for geometries with flat and curved boundaries, respectively. We mesh the domains using TenGen [80] for the linear meshes and using Gmsh for the quadratic and cubic meshes. See Figure 6.34 for some example meshes with linear elements, which are representative in terms of mesh quality but are coarser than



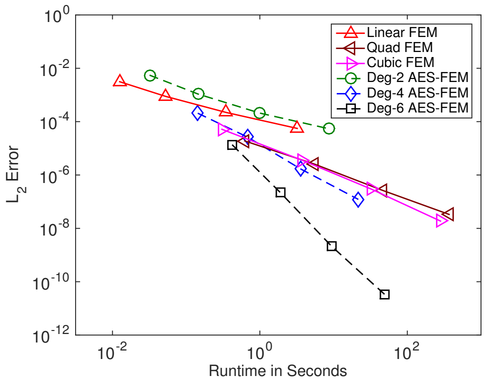


Figure 6.33: The errors verses runtime for 2D convection-diffusion equation on the unit square for exact solution U_2 (left) and on the unit disc for exact solution U_4 (right).

those used in actual computations. The numbers of nodes for the unit cube range from 509 to 7,272,811, and those for the unit ball range from 1,011 to 2,834,229. As in 2D, since isoparametric FEM requires good mesh quality, we ensured that these meshes all have reasonable element shapes: For linear meshes, the minimum dihedral angle is 6.09 degrees and the maximum angle is 166.05 degrees; for high-order meshes, all elements have positive Jacobians everywhere.

We consider the Poisson equation and the convection-diffusion equation. For both cases, we use GMRES with the Gauss-Seidel preconditioner to solve the linear systems arising from AES-FEM. For FEM, we use CG with incomplete Cholesky as the preconditioner for the Poisson equation, and use GMRES with Gauss-Seidel for the convection-diffusion equation. We set the tolerance of the iterative solvers to 10^{-12}. The drop tolerance for incomplete Cholesky is 10^{-3} on the cube and 10^{-6} on the ball.

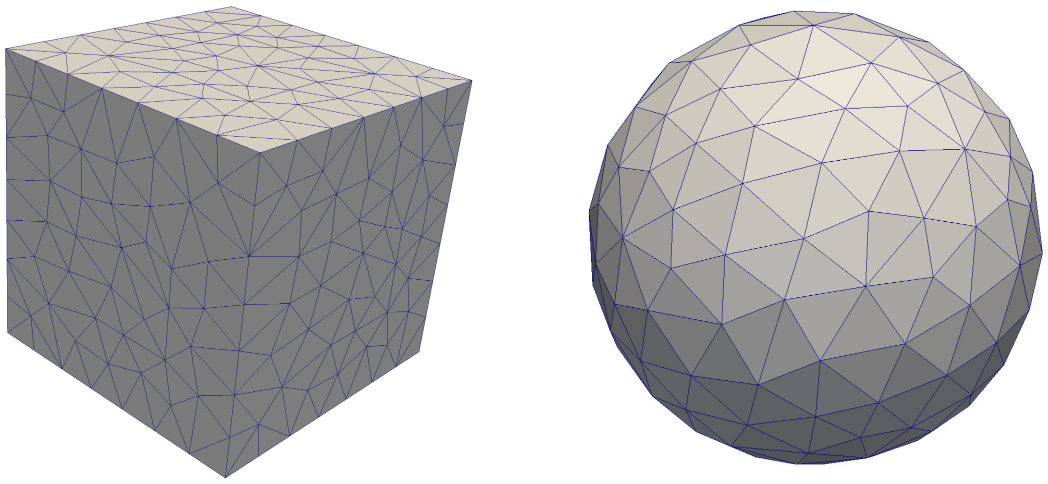


Figure 6.34: Example 3D meshes with linear elements.

Poisson Equation

We first present results for the Poisson equation with Dirichlet boundary conditions on the unit cube and on the unit ball. That is,

$$-\nabla^2 U = \rho \quad \text{in } \Omega, \tag{6.2.9}$$

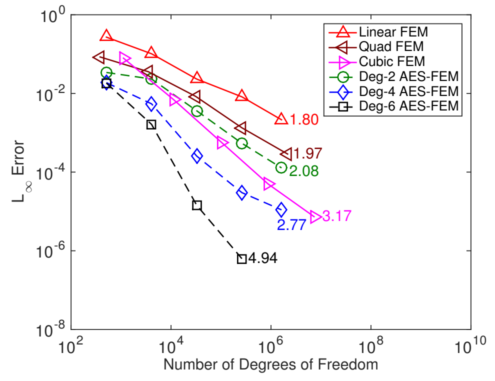
$$U = g \quad \text{on } \partial\Omega. \tag{6.2.10}$$

For the unit cube, where $\Omega = [0,1]^3$, we consider the following three analytic solutions:

$$U_1 = 64x^3\left(1-x^3\right)y^3\left(1-y^3\right)z^3\left(1-z^3\right), \tag{6.2.11}$$

$$U_2 = \cos(\pi x)\cos(\pi y)\cos(\pi z), \tag{6.2.12}$$

$$U_3 = \frac{\sinh(\pi x)\cosh(\pi y)\cosh(\pi z)}{\sinh\pi\cosh^2\pi}. \tag{6.2.13}$$



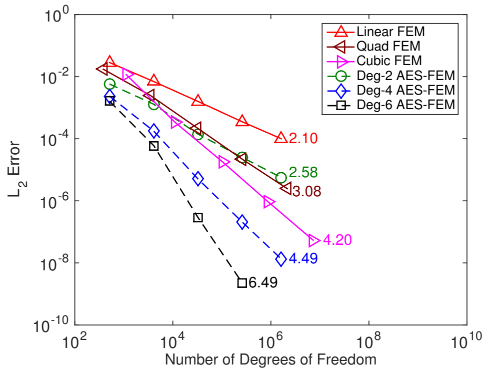


Figure 6.35: The errors for 3D Poisson equation on the unit cube for U_1 in the infinity norm (left) and L_2 norm (right).

For the unit ball $\Omega = \{(x, y, z)|x^2 + y^2 + z^2 \leq 1\}$, we consider the analytic solution U_3 and also

$$U_4 = \cos\left(\frac{\pi}{2}\left(x^2 + y^2 + z^2\right)\right). \tag{6.2.14}$$

For each problem, the right-hand side ρ and the Dirichlet boundary conditions g are obtained from the given analytic solutions.

Figure 6.35 shows the L_∞ and L_2 norm errors for U_1 on the unit cube. The L_2 norm errors for U_2 and U_3 on the unit cube and for U_3 and U_4 on the unit ball are in Figures 6.36 and 6.37, respectively. In all cases, quadratic AES-FEM converges at similar or better rates than linear FEM and has lower errors, and quartic AES-FEM has similar or lower errors than cubic FEM. As in 2D, both of the aforementioned pairs have similar sparsity patterns and similar numbers of nonzeros in the coefficient matrices. Furthermore, sextic AES-FEM is far more accurate than all the other meshes, achieving sixth-order accuracy despite the use of linear elements. This further confirms our accuracy analysis in Section 4.3 for 3D problems.

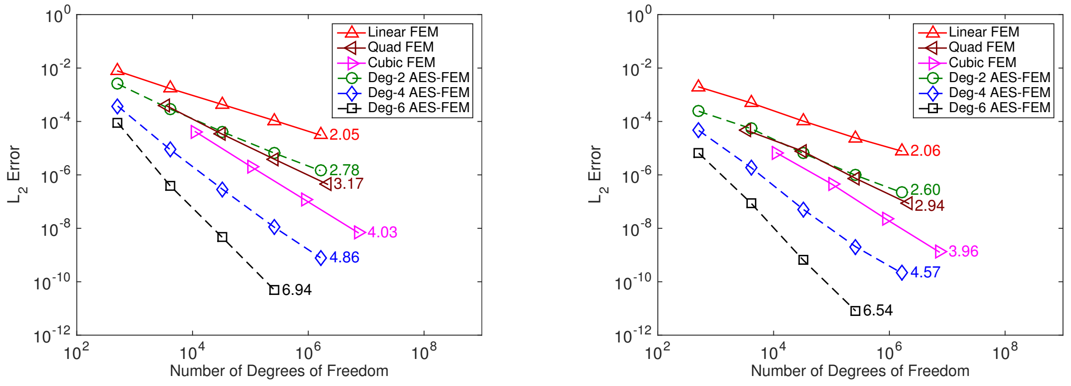


Figure 6.36: The L_2 norm errors for 3D Poisson equation on the unit cube for U_2 (left) and U_3 (right).

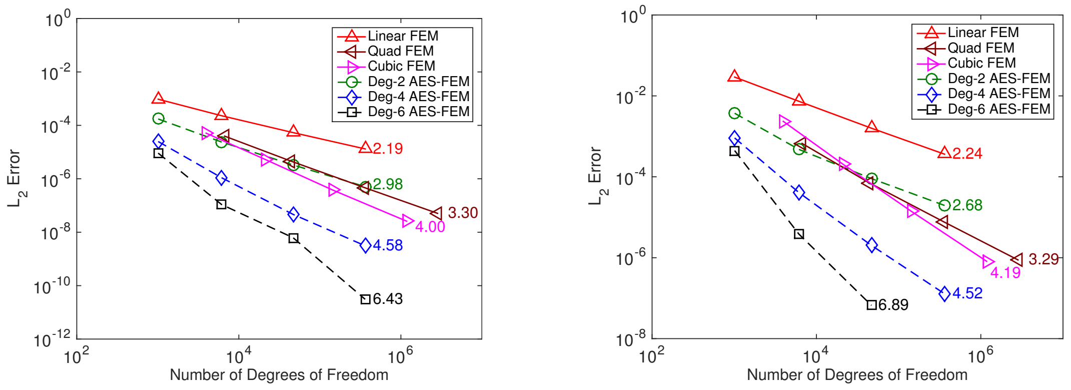


Figure 6.37: The L_2 norm errors for 3D Poisson equation on the unit ball for U_3 (left) and U_4 (right).

Convection-Diffusion Equation

We consider the time-independent convection-diffusion equation with Dirichlet boundary conditions on the unit cube and the unit ball, that is,

$$-\nabla^2 U + \boldsymbol{c} \cdot \boldsymbol{\nabla} U = \rho \quad \text{in } \Omega, \tag{6.2.15}$$

$$U = g \quad \text{on } \partial\Omega. \tag{6.2.16}$$

We take $\boldsymbol{c} = [1, 1, 1]^T$ and we consider the same analytic solutions over the unit cube and unit ball as for the Poisson equation.

Figure 6.38 shows the L_∞ and L_2 norm errors for U_1 on the unit cube. The L_2 norm errors for U_2 and U_3 on the unit cube and for U_3 and U_4 on the unit ball are in Figures 6.39 and 6.40, respectively. Similar to the Poisson equation, quadratic AES-FEM and linear FEM converge at similar rates with quadratic AES-FEM having slightly lower errors. Quartic AES-FEM is more accurate than the cubic FEM in all cases, and sextic AES-FEM is again the most accurate, with about sixth-order convergence. For FEM, the linear solver stagnated for the same problems on the finest mesh, but the resulting errors were small enough not to affect the comparison qualitatively.

Assessment of Element-Quality Dependence

To assess the dependence of AES-FEM and FEM on mesh quality in 3D, we use a series of meshes on the unit cube with progressively worse element quality, which we obtain by distorting a good-quality mesh. For AES-FEM and linear FEM, we

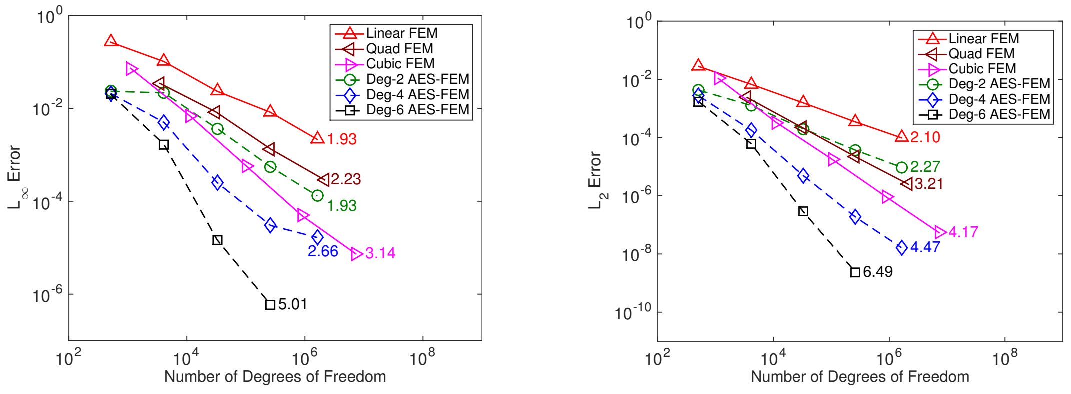


Figure 6.38: The errors for 3D convection-diffusion equation on the unit cube for U_1 in the infinity norm (left) and L_2 norm (right).

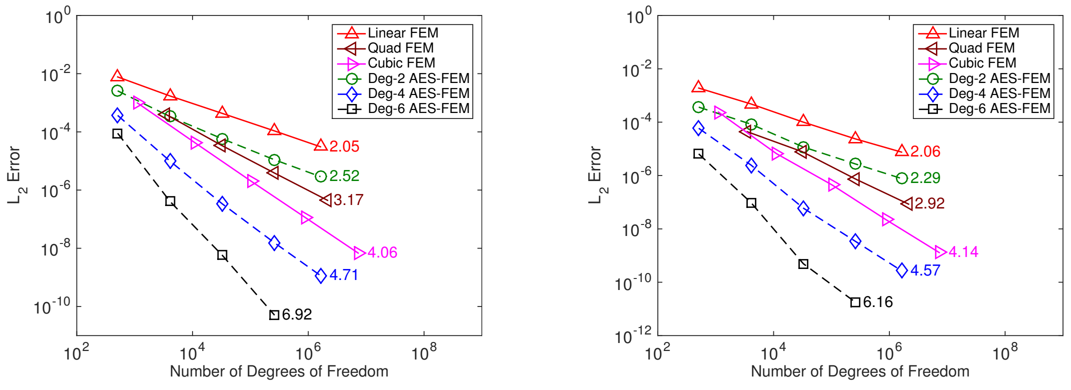


Figure 6.39: The L_2 norm errors for 3D convection-diffusion equation on the unit cube for U_2 (left) and U_3 (right).

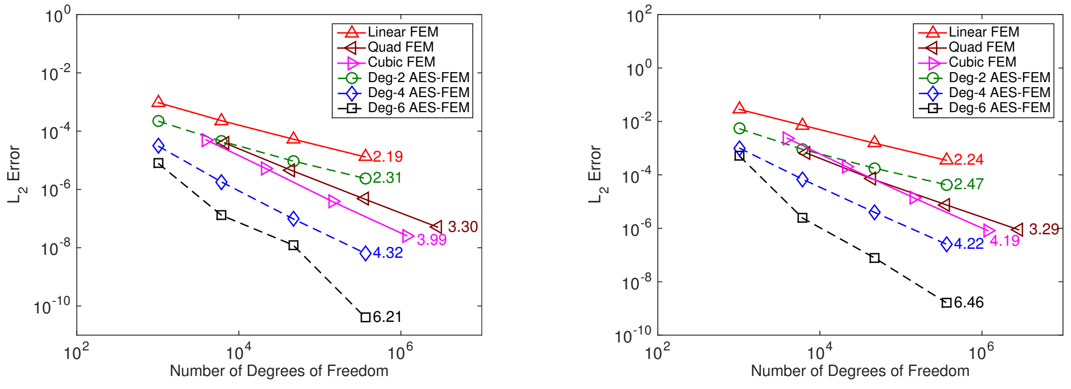


Figure 6.40: The L_2 norm errors for 3D convection-diffusion equation on the unit ball for U_3 (left) and U_4 (right).

use a mesh with 1,604,418 elements and 190,978 nodes and distort 74 elements by moving one vertex of each of these elements incrementally towards its opposite face. For quadratic FEM, we use a mesh with 178,746 elements and 250,047 nodes and distort nine elements by moving one vertex of each of these elements and their adjacent mid-edge and mid-face nodes incrementally towards its opposite face. For cubic FEM, a mesh with 20,250 elements and 97,336 nodes is used and distort a single element by moving one vertex and its adjacent mid-edge, mid-face, and mid-cell nodes incrementally towards its opposite face.

On each distorted mesh, we solve the Poisson equation with the exact solution U_2. The iterative solvers used are GMRES for AES-FEM and CG for FEM. The tolerance for both solvers is 10^{-8}. The preconditioner used for both methods is Gauss-Seidel.

Figure 6.41 shows the condition numbers of the stiffness matrices of FEM and AES-FEM and the numbers of iterations required to solve the linear systems. It can be seen that the condition numbers of FEM increase inversely proportional to the minimum angle, while the condition numbers of quadratic and quartic AES-FEM remain constant. The condition numbers of sextic AES-FEM increase slightly for the second and third meshes, but then dropped back to the original number. In terms of the linear solver, the number of iterations increases significantly for FEM while remaining constant for AES-FEM, independent of element quality.

Efficiency

To compare the efficiency of AES-FEM and FEM in 3D, Figures 6.42 and 6.43 show the errors versus runtimes for the 3D Poisson equation and convection-diffusion equation, respectively. It is evident that on relatively finer meshes, sextic AES-FEM

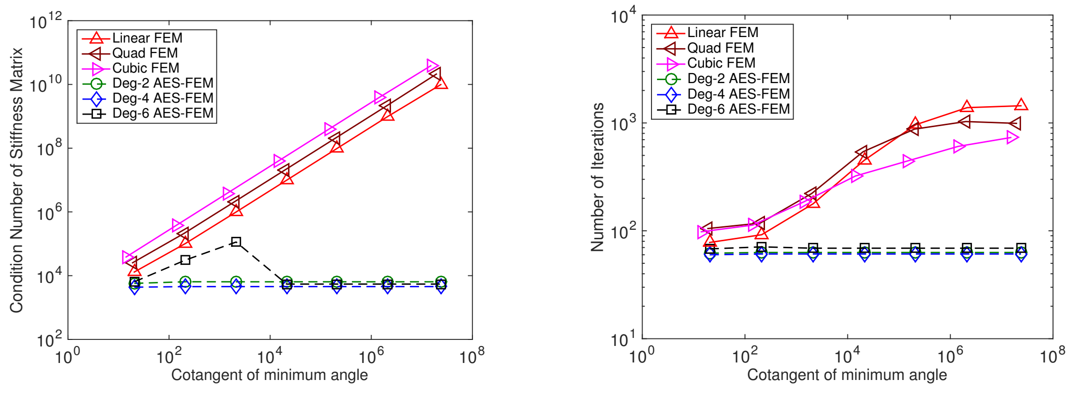


Figure 6.41: Left: the estimated condition numbers of the stiffness matrices of FEM and AES-FEM method on a series of progressive worse meshes in 3D. Right: the number of iterations required for the iterative solver to converge when solving the corresponding systems.

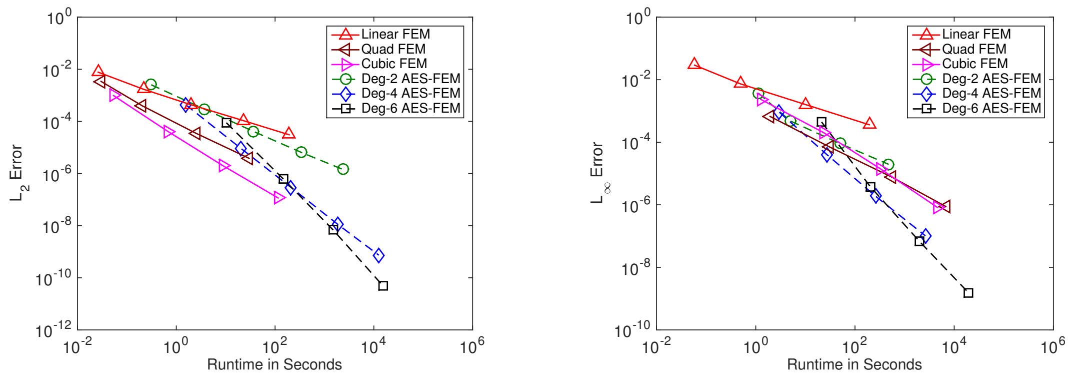
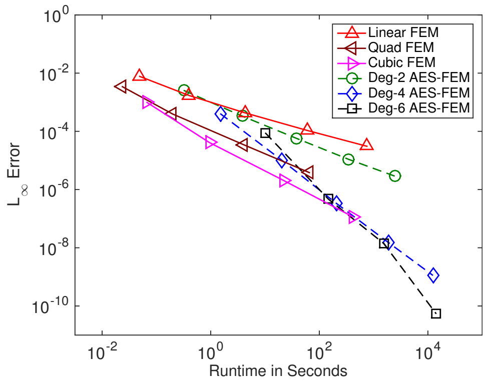


Figure 6.42: The errors verses runtime for 3D Poisson equation on the unit cube for exact solution U_2 (left) and on the unit ball for exact solution U_4 (right).

outperforms cubic FEM, and for the Poisson equation on the unit ball, quartic AES-FEM does as well. We expect AES-FEM will perform even better with further code optimizations.



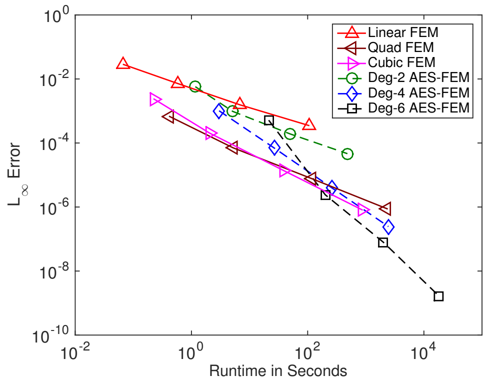


Figure 6.43: The errors verses runtime for 3D convection-diffusion equation on the unit cube for exact solution U_2 (left) and on the unit ball for exact solution U_4 (right).

6.3 Linear Solvers

The stiffness matrix resulting from AES-FEM is a large, non-symmetric, sparse system. Krylov subspace methods are often used for iteratively solving such systems [73]. For symmetric positive definite systems, the conjugate gradient method (CG) [40] is preferred and for symmetric indefinite systems, the minimum residual method (MINRES) [65] is the typical choice [29]. CG and MINRES both use the Lanczos process. CG has a two-term recurrence and MINRES has a three-term recurrence. CG minimizes the energy norm of the residual and MINRES minimizes the two-norm. Both CG and MINRES require one matrix-vector multiplication per iteration.

For non-symmetric systems, the choice of which solver to use is less clear than for the symmetric case. Popular methods include the generalized minimal residual method (GMRES) [74], the biconjugate gradient stabilized method (Bi-CGSTAB) [86], and the transpose-free quasi-minimal residual method (TFQMR) [31]. GMRES uses the Arnoldi iteration to find the vector that minimizes the residual. Each iteration requires one matrix-vector multiplication. GMRES often requires a restart due to its k-term recurrence, where k is the iteration count. Bi-CGSTAB uses the non-symmetric Lanczos iteration. Each iteration requires two matrix-vector multiplications. Bi-CGSTAB has a three-term recurrence. TFQMR also uses the non-symmetric Lanczos iteration and has a three-term recurrence. Like Bi-CGSTAB, each iteration of TFQMR requires two matrix-vector multiplications.

Krylov subspace methods are typically used in conjunction with preconditioners, which can improve the robustness and efficiency of the solver. When solving a symmetric system, the preconditioner should preserve the symmetry and thus a two-sided preconditioner is used. If we are solving the system $\boldsymbol{Ax} = \boldsymbol{b}$, then the corresponding preconditioned system will be $\boldsymbol{SASy} = \boldsymbol{Sb}$ where $\boldsymbol{x} = \boldsymbol{Sy}$. For symmetric systems, incomplete Cholesky factorization [87], shifted incomplete Cholesky factorization [58], or Gauss-Seidel [47] are common choices for a preconditioner. When solving a non-symmetric system $\boldsymbol{Ax} = \boldsymbol{b}$, either a left preconditioner or a right preconditioner may be applied, that is, $\boldsymbol{MAx} = \boldsymbol{Mb}$ or $\boldsymbol{AMy} = \boldsymbol{Mb}$ with $\boldsymbol{x} = \boldsymbol{My}$, respectively. Using a left preconditioner changes the norm in which the residual is minimized. If a right preconditioner is used, the norm is unchanged. MATLAB uses left preconditioners. For non-symmetric systems, incomplete LU factorization [72], algebraic multigrid [13], or Gauss-Seidel can be used as preconditioners. Another option for solving this system would be multigrid methods [85], and we will explore their use in future work.

The performance of any method depends on the number of matrix-vector multiplications and inner-product, whether the solver uses a 3-term or k-term recurrence, the condition number of the matrix, and what preconditioner (if any) is used.

Here, we conduct a comparison of various combinations for solvers and preconditioners for solving the linear systems obtained from quadratic, quartic, and sextic AES-FEM and linear, quadratic, and cubic FEM. In Table 6.2, we list the size, the number of non-zero entries, and the condition number for each matrix. Applying FEM to the Poisson equation results in a symmetric matrix, and all other matrices are non-symmetric. For solving non-symmetric systems, we consider GMRES, BiCGSTAB, and TFQMR. We use Gauss Seidel as a left and a right preconditioner. We compare two variants of incomplete LU: no-fill, which we use as a left and a right preconditioner, and ilutp, which we use as a left preconditioner with drop-tolerances of $\tau = 10^{-3}$, $\tau = 10^{-6}$, and $\tau = 10^{-9}$ in 2D and $\tau = 10^{-1}$ and $\tau = 10^{-2}$ in 3D. We also include an algebraic multigrid preconditioner, both as a left and a right preconditioner. When using a left precondition, other than AMG, we used the builtin MATLAB solvers. For solvers with a right preconditioner and for the solvers with left or right AMG, we used our own implementation. For symmetric systems, we include a comparison of conjugate gradient and MINRES, with Gauss Seidel and incomplete Cholesky preconditioners. We consider two variants of incomplete Cholesky, one with no-fill and one with the same drop tolerances as ILU. We rank the method by their runtimes on one core of a 2.6 GHz Xeon E5-2670 CPU. We use a tolerance of 10^{-12} for the solvers. For GMRES, we use restart equal to 20.

For solving AES-FEM systems from the 2D Poisson equation, the three most efficient methods are BiCGSTAB with right no-fill ILU is the most efficient, GMRES with right AMG, and then TFQMR with left no-fill ILU. See Table 6.3 for the runtimes. For solving FEM systems from the 2D Poisson equation, the three most efficient methods are CG with incomplete Cholesky with $\tau = 10^{-3}$, MINRES with incomplete Cholesky with $\tau = 10^{-3}$, and then CG with incomplete Cholesky with $\tau = 10^{-6}$. See Table 6.4 for the runtimes. The matrices for linear FEM and quadratic AES-FEM are the same size, although in 2D the number of non-zeros is greater for AES-FEM and the sparsity patterns are different. The fastest method for linear FEM required 0.7188 seconds and the fastest for quadratic AES-FEM requires 1.964 seconds. We avoid comparing the run times of the high-order methods since the matrices have different size, sparsity patterns, and number of non-zeros.

For solving AES-FEM systems from the 2D convection-diffusion equation, the three most efficient methods are BiCGSTAB with left no-fill ILU, GMRES with right AMG, and then BiCGSTAB with right no-fill ILU. See Table 6.5 for the runtimes. For solving FEM systems from the 2D convection-diffusion equation, the three most efficient methods are GMRES with right AMG, BiCGSTAB with right AMG, and then BiCGSTAB with left AMG. See Table 6.6 for the runtimes. The fastest method for linear FEM required 1.838 seconds and the fastest for quadratic AES-FEM requires 2.131 seconds.

For solving AES-FEM systems from the 3D Poisson equation, the three most efficient methods are BiCGSTAB with right no-fill ILU, BiCGSTAB with left no-fill ILU, and then TFQMR with left no-fill ILU. See Table 6.7 for the runtimes. For solving FEM systems from the 3D Poisson equation, the most efficient methods are CG with incomplete Cholesky with $\tau = 10^{-3}$. Incomplete Cholesky with $\tau = 10^{-2}$ encountered a non-positive pivot for quadratic FEM, but for linear and cubic FEM, this preconditioner coupled with CG is also very efficient. See Table 6.8 for the runtimes. The matrices for linear FEM and quadratic AES-FEM are the same size with the same number of non-zeros and sparsity patterns. The fastest method for linear FEM required 6.971 seconds and the fastest method for AES-FEM required 8.284 seconds. Again, we avoid comparing the run times of the high-order methods since the matrices have different size, sparsity patterns, and number of non-zeros.

For solving AES-FEM systems from the 3D convection-diffusion equation, the three most efficient methods are BiCGSTAB with left no-fill ILU, BiCGSTAB with right no-fill ILU, and then TFQMR with left no-fill ILU. See Table 6.9 for the runtimes. For solving FEM systems from the 3D Poisson equation, the three most efficient methods are BiCGSTAB with left no-fill ILU, BiCGSTAB with right no-fill ILU, and then BiCGSTAB with $\tau = 10^{-1}$. See Table 6.10 for the runtimes. The fastest method for linear FEM required 11.191 seconds and the fastest method for AES-FEM required 8.790 seconds.

Table 6.2: Description of the matrices that we use in testing the runtimes.

			Size	#Non-Zero	Cond. Num.
2D	Poisson	Deg-2 AES-FEM	64,635	830,379	5.4729×10^4
		Deg-4 AES-FEM	64,635	1,971,551	5.1220×10^4
		Deg-6 AES-FEM	64,635	3,534,311	7.8084×10^4
		Linear FEM	64,635	450,381	5.0979×10^4
		Quad FEM	64,077	730,335	8.6221×10^4
		Cubic FEM	35,407	592,755	7.9266×10^4
	Conv-Diff	Deg-2 AES-FEM	64,635	830,379	5.3261×10^4
		Deg-4 AES-FEM	64,635	1,971,551	5.0594×10^4
		Deg-6 AES-FEM	64,635	3,534,311	7.5688×10^4
		Linear FEM	64,635	450381	5.0524×10^4
		Quad FEM	64,077	730,515	8.5397×10^4
		Cubic FEM	35,407	592,765	7.8595×10^4
3D	Poisson	Deg-2 AES-FEM	237,737	3,678,999	6.0501×10^3
		Deg-4 AES-FEM	237,737	11,954,665	4.3106×10^3
		Deg-6 AES-FEM	237,737	30,460,354	3.9280×10^4
		Linear FEM	237,737	3,678,771	9.0463×10^3
		Quad FEM	239,099	6,641,095	1.9284×10^4
		Cubic FEM	91,088	4,086,480	1.7085×10^4
	Conv-Diff	Deg-2 AES-FEM	237,737	3,678,999	6.2140×10^3
		Deg-4 AES-FEM	237,737	11,954,665	4.2780×10^3
		Deg-6 AES-FEM	237,737	30,460,354	2.4835×10^5
		Linear FEM	237,737	3,678,771	8.9887×10^3
		Quad FEM	239,099	6,641,095	1.9133×10^4
		Cubic FEM	91,088	4,086,480	1.6975×10^4

Table 6.3: Runtime in seconds for solving AES-FEM systems from 2D Poisson equation.

AES-FEM					
			Quad	Quartic	Sextic
GMRES	GS	Left	17.04	20.29	26.78
		Right	10.19	16.15	21.89
	ILU	Left $\tau = 10^{-3}$	5.77	9.82	19.29
		Left $\tau = 10^{-6}$	97.38	184.83	304.79
		Left $\tau = 10^{-9}$	280.01	585.38	989.64
		Left no-fill	14.07	11.79	7.42
		Right no-fill	8.41	8.15	6.18
	AMG	Left	3.01	-	9.69
		Right	1.96	3.65	5.86
BiGCSTAB	GS	Left	3.39	4.79	7.01
		Right	3.31	6.09	10.59
	ILU	Left $\tau = 10^{-3}$	4.80	8.86	17.84
		Left $\tau = 10^{-6}$	97.81	182.32	303.30
		Left $\tau = 10^{-9}$	278.03	583.30	975.45
		Left no-fill	2.95	3.16	3.75
		Right no-fill	2.99	4.23	5.15
	AMG	Left	2.98	5.87	9.70
		Right	2.13	3.59	6.40
TFQMR	GS	Left	3.34	5.18	7.32
		Right	4.76	10.91	17.42
	ILU	Left $\tau = 10^{-3}$	5.10	8.91	17.97
		Left $\tau = 10^{-6}$	95.50	182.17	301.78
		Left $\tau = 10^{-9}$	280.99	593.73	962.78
		Left no-fill	2.92	3.67	4.02
		Right no-fill	4.30	7.24	9.28
	AMG	Left	3.19	6.09	9.33
		Right	3.14	6.50	9.72

Table 6.4: Runtime in seconds for solving FEM systems from 2D Poisson equation.

FEM					
			Linear	Quad	Cubic
CG	Gauss Seidel	symmetric	2.89	4.11	2.94
	Incomplete Cholesky	$\tau = 10^{-3}$	0.72	0.95	0.79
		$\tau = 10^{-6}$	1.44	2.57	1.43
		$\tau = 10^{-9}$	6.42	18.53	6.61
		no-fill	3.06	3.87	3.03
MINRES	GS	symmetric	3.30	4.18	3.27
	Incomplete Cholesky	$\tau = 10^{-3}$	0.80	0.99	0.85
		$\tau = 10^{-6}$	1.50	2.59	1.49
		$\tau = 10^{-9}$	6.66	18.61	6.64
		no-fill	3.63	4.57	3.97

Table 6.5: Runtime in seconds for solving AES-FEM systems from 2D convection-diffusion equation.

AES-FEM					
			Quad	Quartic	Sextic
GMRES	GS	Left	16.14	21.789	32.95
		Right	11.42	18.06	26.19
	ILU	Left $\tau = 10^{-3}$	5.30	9.28	19.34
		Left $\tau = 10^{-6}$	89.28	179.23	304.26
		Left $\tau = 10^{-9}$	270.98	591.11	975.91
		Left no-fill	15.23	10.74	8.19
		Right no-fill	8.79	7.68	6.03
	AMG	Left	3.21	5.78	9.69
		Right	2.13	3.71	6.01
BiGCSTAB	GS	Left	3.07	5.30	6.36
		Right	3.41	7.08	10.25
	ILU	Left $\tau = 10^{-3}$	4.66	8.59	18.19
		Left $\tau = 10^{-6}$	90.02	187.08	298.89
		Left $\tau = 10^{-9}$	268.00	605.10	974.37
		Left no-fill	2.99	3.43	3.57
		Right no-fill	2.65	4.29	5.35
	AMG	Left	3.27	5.30	9.44
		Right	2.25	3.73	6.56
TFQMR	GS	Left	3.45	5.33	8.21
		Right	5.48	11.37	18.84
	ILU	Left $\tau = 10^{-3}$	4.83	8.68	17.67
		Left $\tau = 10^{-6}$	92.24	183.83	298.05
		Left $\tau = 10^{-9}$	272.80	598.31	985.59
		Left no-fill	3.25	3.98	3.76
		Right no-fill	5.06	8.06	9.22
	AMG	Left	3.17	5.88	10.10
		Right	3.48	6.60	10.07

Table 6.6: Runtime in seconds for solving FEM systems from 2D convection-diffusion equation.

FEM					
			Linear	Quad	Cubic
GMRES	GS	Left	40.21	38.79	38.84
		Right	24.25	31.00	23.82
	ILU	Left $\tau = 10^{-3}$	2.93	27.94	2.67
		Left $\tau = 10^{-6}$	30.03	475.87	29.28
		Left $\tau = 10^{-9}$	89.28	1988.43	86.29
		Left no-fill	31.74	43.60	33.92
		Right no-fill	16.56	24.63	17.32
	AMG	Left	3.12	3.55	2.95
		Right	2.04	2.10	1.69
BiGCSTAB	GS	Left	4.66	6.04	4.50
		Right	3.90	6.01	3.84
	ILU	Left $\tau = 10^{-3}$	2.04	26.61	2.03
		Left $\tau = 10^{-6}$	29.45	483.68	29.21
		Left $\tau = 10^{-9}$	88.36	2015.26	88.54
		Left no-fill	4.07	5.50	4.05
		Right no-fill	3.44	5.15	3.48
	AMG	Left	2.79	3.57	2.76
		Right	1.84	2.26	1.80
TFQMR	GS	Left	5.07	7.02	5.37
		Right	6.01	10.28	6.16
	ILU	Left $\tau = 10^{-3}$	2.16	27.26	2.25
		Left $\tau = 10^{-6}$	29.19	489.79	29.95
		Left $\tau = 10^{-9}$	87.01	2015.69	91.15
		Left no-fill	4.77	5.55	4.52
		Right no-fill	16.55	8.03	16.61
	AMG	Left	3.23	3.69	3.30
		Right	3.37	3.72	3.31

Table 6.7: Runtime in seconds for solving AES-FEM systems from 3D Poisson equation.

AES-FEM					
			Quad	Quartic	Sextic
GMRES	GS	Left	17.21	61.45	1644.59
		Right	14.13	51.73	927.43
	ILU	Left $\tau = 10^{-1}$	28.41	70.62	214.59
		Left $\tau = 10^{-2}$	58.58	350.47	1515.67
		Left no-fill	14.98	59.67	50.92
		Right no-fill	9.42	36.36	47.98
	AMG	Left	74.20	224.11	659.47
		Right	46.97	85.87	694.64
BiGCSTAB	GS	Left	9.70	28.84	161.72
		Right	9.06	24.72	174.34
	ILU	Left $\tau = 10^{-1}$	10.83	42.99	83.96
		Left $\tau = 10^{-2}$	57.46	316.41	1458.80
		Left no-fill	8.88	22.35	39.62
		Right no-fill	8.28	21.59	40.58
	AMG	Left	63.14	146.47	20065.70
		Right	44.31	82.68	8109.98
TFQMR	GS	Left	11.32	31.87	265.95
		Right	15.63	48.42	449.45
	ILU	Left $\tau = 10^{-1}$	12.87	43.57	126.19
		Left $\tau = 10^{-2}$	91.72	402.21	53.48
		Left no-fill	8.44	22.90	41.70
		Right no-fill	13.32	46.15	68.16
	AMG	Left	75.87	163.45	-
		Right	81.72	163.27	7022.07

Table 6.8: Runtime in seconds for solving FEM systems from 3D Poisson equation.

FEM					
			Linear	Quad	Cubic
CG	Gauss Seidel	symmetric	14.93	22.65	16.31
	Incomplete Cholesky	$\tau = 10^{-1}$	11.66	23.06	11.89
		$\tau = 10^{-2}$	7.03	-	6.67
		$\tau = 10^{-3}$	6.97	11.98	7.05
		no-fill	8.59	12.57	9.81
MINRES	Gauss Seidel	symmetric	15.95	22.74	17.13
	Incomplete Cholesky	$\tau = 10^{-1}$	12.19	23.26	13.03
		$\tau = 10^{-2}$	7.43	-	7.63
		$\tau = 10^{-3}$	8.18	12.54	7.85
		no-fill	9.15	12.17	9.22

Table 6.9: Runtime in seconds for solving AES-FEM systems from 3D convection-diffusion equation.

AES-FEM					
			Quad	Quartic	Sextic
GMRES	GS	Left	16.61	39.37	303.28
		Right	17.34	39.41	350.79
	ILU	Left $\tau = 10^{-1}$	29.12	81.04	148.93
		Left $\tau = 10^{-2}$	88.96	293.27	1114.39
		Left no-fill	20.07	27.17	62.28
		Right no-fill	12.58	22.63	47.78
	AMG	Left	84.19	207.78	-
		Right	48.22	91.09	552.05
BiGCSTAB	GS	Left	9.83	25.54	133.30
		Right	9.70	32.11	165.81
	ILU	Left $\tau = 10^{-1}$	11.47	41.83	79.67
		Left $\tau = 10^{-2}$	81.81	279.21	1113.45
		Left no-fill	9.58	18.84	44.75
		Right no-fill	8.79	19.19	44.29
	AMG	Left	63.14	124.51	-
		Right	40.51	78.51	-
TFQMR	GS	Left	10.66	28.19	182.43
		Right	20.28	60.90	314.12
	ILU	Left $\tau = 10^{-1}$	12.86	43.39	76.99
		Left $\tau = 10^{-2}$	80.80	285.52	1012.30
		Left no-fill	9.49	19.16	48.95
		Right no-fill	17.01	30.52	82.37
	AMG	Left	63.38	130.78	-
		Right	76.50	136.64	-

Table 6.10: Runtime in seconds for solving FEM systems from 3D convection-diffusion equation.

FEM					
			Linear	Quad	Cubic
GMRES	GS	Left	29.27	35.49	24.40
		Right	25.45	36.36	26.06
	ILU	Left $\tau = 10^{-1}$	27.61	66.97	26.14
		Left $\tau = 10^{-2}$	79.01	726.72	63.57
		Left no-fill	24.50	31.68	25.67
		Right no-fill	18.11	25.89	18.74
	AMG	Left	75.34	46.93	70.97
		Right	34.32	19.36	12.30
BiGCSTAB	GS	Left	15.26	17.16	12.31
		Right	15.67	22.00	15.54
	ILU	Left $\tau = 10^{-1}$	11.27	19.89	11.16
		Left $\tau = 10^{-2}$	77.28	702.59	59.65
		Left no-fill	11.66	15.70	11.89
		Right no-fill	11.19	18.50	11.60
	AMG	Left	43.16	32.48	38.69
		Right	28.83	22.80	12.63
TFQMR	GS	Left	14.61	17.17	12.95
		Right	25.54	35.41	21.44
	ILU	Left $\tau = 10^{-1}$	13.56	19.97	12.43
		Left $\tau = 10^{-2}$	78.67	697.63	59.39
		Left no-fill	12.98	16.32	12.56
		Right no-fill	22.22	31.81	20.30
	AMG	Left	46.11	32.56	41.60
		Right	52.32	37.81	21.86

Chapter 7

Conclusion and Further Work

The primary contribution of this dissertation is the presentation of the adaptive extended stencil finite element method (AES-FEM), which uses generalized Lagrange polynomial basis functions constructed via weighted least squares approximations to overcome the element-dependency of the traditional finite element method. Generalized Lagrange polynomial basis functions extend the concept of interpolary Lagrange basis functions to weighted least-squares approximations, while retaining two critical properties, i.e., function value as coefficient and partition of unity. By utilizing generalized Lagrange basis functions, AES-FEM is insensitive to element-quality. Additionally, AES-FEM can reach high-order accuracy with linear meshes, even for domains with curved boundaries. AES-FEM preserves the theoretical framework of the classical FEM and the simplicity in imposing essential boundary conditions and integrating the stiffness matrix.

In this dissertation, we presented the formulation of AES-FEM, extended the method to high-order, showed that the method is consistent, and discussed both the local and global stability of the method. We described the implementation, including the mesh data structure and the numerical algorithms. We compared the accuracy of AES-FEM against the classical FEM with linear basis functions and the quadratic generalized finite difference method for the Poisson and convection-diffusion equations in both 2D and 3D. We showed improved accuracy and stability of quadratic AES-FEM over linear FEM, and demonstrated that the condition number of AES-FEM, and hence the convergence rate of iterative solvers, are independent of the element quality of the mesh. We compared the accuracy of quadratic, quartic and sextic AES-FEM against linear, quadratic and cubic FEM for the Poisson equation and the time-independent convection-diffusion equation in 2D and 3D, including on domains with curved boundaries. We showed improved accuracy and stability of high-order AES-FEM over high-order FEM. We demonstrated up to sixth order convergence rates of AES-FEM, despite the use of linear elements. Our experiments also showed that AES-FEM is more efficient than the classical FEM in terms of error versus runtime, while having virtually the same sparsity patterns of the stiffness matrices. Since AES-FEM results in a non-symmetric matrix, we compared the efficiency of linear solvers and preconditioners.

The future work for AES-FEM can be thought of in two broad categories: improvements to the method and applications of the method. We discuss the first in the paragraph. While AES-FEM is efficient in terms of error versus runtime, it is slower than the classical FEM on a given mesh due to the more expensive computation of the basis functions and the non-symmetry of the stiffness matrix. The efficiency can be improved substantially by exploring p-adaptivity, leveraging the parallelism and the efficient multigrid solvers, which we will report in the future. Our present implementation of AES-FEM uses the standard hat functions as the weight functions, which may lead to large errors when applied to tangled meshes with inverted elements. We will report the resolution of tangled meshes in a future publication. This work has focused on elliptic or diffusion-dominant PDEs with solutions with smooth solutions. Another future research direction is the resolution of discontinuities from hyperbolic problems.

Another area of future work is exploring applications. Because AES-FEM performs well on poor-quality meshes, it would be highly applicable for applications with deforming meshes and could reduce, or possibly eliminate, the need for remeshing. One area of interesting is fluid structure interactions, including the modeling of thin-shells and membranes. For these models, it is well known that the C^0-continuous finite elements may not converge on poor-quality elements. AES-FEM can overcome this problem. Another area of interest is linear and nonlinear elasticity, including fracture mechanics problems. A third area of interest is superconductivity, especially as applied to super conductor digital circuits.

Bibliography

[1] M. Ainsworth and B. Senior. An adaptive refinement strategy for hp-finite element computations. *Appl. Numer. Math.*, 26(1):165–178, 1998.

[2] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2002.

[3] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Anal.*, 13(2):214–226, 1976.

[4] I. Babuška and M. Suri. The p and $h-p$ versions of the finite element method, basic principles and properties. *SIAM Review*, 36(4):578–632, 1994.

[5] I. Babuška, B. A. Szabo, and I. N. Katz. The p-version of the finite element method. *SIAM J. Numer. Anal.*, 18(3):515–545, 1981.

[6] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. *J. Comput. Phys.*, 131(2):267–279, 1997.

[7] F. Bassi and S. Rebay. High-order accurate discontinuous finite element solution of the 2D Euler equations. *J. Comput. Phys.*, 138(2):251–285, 1997.

[8] T. Belytschko, R. Gracie, and G. Ventura. A review of extended/generalized finite element methods for material modeling. *Model. Simul. Mater. Sci. Eng.*, 17(4):043001, 2009.

[9] T. Belytschko, Y. Y. Lu, and L. Gu. Element free Galerkin methods. *Int. J. Numer. Meth. Engrg.*, 37(2):229–256, 1994.

[10] J. Benito, F. Ureña, and L. Gavete. Influence of several factors in the generalized finite difference method. *Appl. Math. Model.*, 25:1039–1053, 2001.

[11] J. Benito, F. Ureña, and L. Gavete. Solving parabolic and hyperbolic equations by the generalized finite difference method. *J. Comput. Appl. Math.*, 209(2):208–233, 2007.

[12] P. B. Bochev and M. D. Gunzburger. Finite element methods of least-squares type. *SIAM Rev.*, 40(4):789–837, 1998.

[13] A. Brandt, S. McCoruick, and J. Huge. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and its Applications*, 257:257–284, 1985.

[14] S. C. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods*, volume 15. Springer Science & Business Media, College Park, MD, 2008.

[15] C. Cantwell, S. Sherwin, R. Kirby, and P. Kelly. From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements. *Comput. Fluids*, 43(1):23–28, 2011.

[16] C. Canuto, M. Y. Hussaini, A. M. Quarteroni, A. Thomas Jr, et al. *Spectral methods in fluid dynamics*. Springer Science & Business Media, Berlin Heidelberg, 2012.

[17] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Comput. Aid. Geom. Des.*, 22(2):121–146, 2005.

[18] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. SIAM, Philadelphia, 2nd edition, 2002.

[19] P. G. Ciarlet and P.-A. Raviart. Interpolation theory over curved elements, with applications to finite element methods. *Comput. Methods in Appl. Mech. Eng*, 1(2):217–249, 1972.

[20] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM J. Numer. Anal.*, 47(2):1319–1365, 2009.

[21] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. *The Development of Discontinuous Galerkin Methods*. Springer, Berlin Heidelberg, 2000.

[22] R. Conley, T. J. Delaney, and X. Jiao. High-order adaptive extended stencils FEM with linear elements. *SIAM J. Sci. Comput.*, 2016. Submitted. Preprint availabe at http://arxiv.org/abs/1603.09325.

[23] R. Conley, T. J. Delaney, and X. Jiao. Overcoming element quality dependence of finite elements with adaptive extended stencil FEM (AES-FEM). *Int. J. Numer. Meth. Engng.*, 2016. doi:10.1002/nme.5246.

[24] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy. Toward a universal hp adaptive finite element strategy, part 1. constrained approximation and data structure. *Comput. Method Appl. M.*, 77(1):79–112, 1989.

[25] L. Demkowicz, W. Rachowicz, and P. Devloo. A fully automatic hp-adaptivity. *SIAM J. Sci. Comput.*, 17(1):117–142, 2002.

[26] V. Dyedov, N. Ray, D. Einstein, X. Jiao, and T. J. Tautges. AHF: Array-based half-facet data structure for mixed-dimensional and non-manifold meshes. In *Proceedings of the 22nd International Meshing Roundtable*, pages 445–464. Springer, 2014.

[27] I. Ergatoudis, B. Irons, and O. Zienkiewicz. Curved, isoparametric, "quadrilateral" elements for finite element analysis. *Int. J. Solids Struct.*, 4(1):31–42, 1968.

[28] B. A. Finlayson. *The Method of Weighted Residuals and Variational Principles*. Academic Press, New York, 1973.

[29] D. C.-L. Fong and M. A. Saunders. CG versus MINRES: An empirical comparison. *SQU Journal for Science*, 17(1):44–62, 2012.

[30] G. E. Forsythe and W. R. Wasow. *Finite-Difference Methods for Partial Differential Equations*. John Wiley & Sons, Inc, New York - London, 1960.

[31] R. W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14(2):470–482, 1993.

[32] I. Fried. Accuracy of complex finite elements. *AIAA Journal*, 10(3):347–349, 1972.

[33] T.-P. Fries and T. Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *Int. J. Numer. Meth. Engrg.*, 84(3):253–304, 2010.

[34] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Defining quality measures for validation and generation of high-order tetrahedral meshes. In *Proceedings of the 22nd International Meshing Roundtable*, pages 109–126. Springer, 2014.

[35] L. Gavete, M. Gavete, and J. Benito. Improvements of generalized finite difference method and comparison with other meshless methods. *Appl. Math. Model.*, 27:831–847, 2003.

[36] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *Int. J. Numer. Meth. Engrg.*, 79(11):1309–1331, 2009.

[37] L. N. Gifford. More on distorted isoparametric elements. *Int. J. Numer. Meth. Engrg.*, 14(2):290–291, 1979.

[38] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins, Baltimore, 4th edition, 2013.

[39] W. Gui and I. Babuska. The h, p and hp versions of the finite element method in 1 dimension. part 1. the error analysis of the p-version. Technical report, Maryland Univ College Park Lab for Numerical Analysis, 1985.

[40] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Stand.*, 49(6):409–436, 1952.

[41] N. J. Higham. A survey of condition number estimation for triangular matrices. *SIAM Rev.*, 29(4):575–596, 1987.

[42] T. J. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Meth. Appl. Mech. Engrg.*, 194(39):4135–4195, 2005.

[43] B. M. Irons. Engineering applications of numerical integration in stiffness methods. *AIAA Journal*, 4(11):2035–2037, 1966.

[44] P. S. Jensen. Finite difference techniques for variable grids. *Comput. Struct.*, 2(1):17–29, 1972.

[45] X. Jiao and H. Zha. Consistent computation of first-and second-order differential quantities for surface meshes. In *ACM Symposium on Solid and Physical Modeling*, pages 159–170. ACM, 2008.

[46] A. Johnen, J.-F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. *J. Comput. Phys.*, 233:359–372, 2013.

[47] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[48] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Math. Comp.*, 37(155):141–158, 1981.

[49] N.-S. Lee and K.-J. Bathe. Effects of element distortions on the performance of isoparametric elements. *Int. J. Numer. Meth. Engng.*, 36(20):3553–3576, 1993.

[50] M. Lenoir. Optimal isoparametric finite elements and error estimates for domains involving curved boundaries. *SIAM J. Numer. Anal.*, 23(3):562–580, 1986.

[51] X. Li, M. S. Shephard, and M. W. Beall. Accounting for curved domains in mesh adaptation. *Int. J. Numer. Meth. Engrg.*, 58(2):247–276, 2003.

[52] T. Liszka and J. Orkisz. The finite difference method at arbitrary irregular grids and its application in applied mechanics. *Comput. Struct.*, 11(1):83–95, 1980.

[53] H. Liu and X. Jiao. WLS-ENO: Weighted least squares based essentially non-oscillatory schemes on unstructured meshes. *J. Comput. Phys.*, 2016 (In press). DOI: 10.1016/j.jcp.2016.03.039.

[54] S. Lo. A new mesh generation scheme for arbitrary planar domains. *Int. J. Numer. Meth. Engrg.*, 21(8):1403–1426, 1985.

[55] S. H. Lui. *Numerical Analysis of Partial Differential Equations*. John Wiley & Sons, Hoboken, 2012.

[56] X. Luo, M. S. Shephard, J.-F. Remacle, R. M. O'Bara, M. W. Beall, B. Szabó, and R. Actis. p-version mesh generation issues. In *International Meshing Roundtable*, pages 343–354, Sandia National Laboratories, 2002.

[57] X.-J. Luo, M. S. Shephard, L.-Z. Yin, R. M. O'Bara, R. Nastasi, and M. W. Beall. Construction of near optimal meshes for 3d curved domains with thin sections and singularities for p-version method. *Eng. Comput.*, 26(3):215–229, 2010.

[58] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 34(150):473–497, 1980.

[59] J. M. Melenk and I. Babuška. The partition of unity finite element method: Basic theory and applications. *Comput. Method. Appl. M.*, 139(1):289–314, 1996.

[60] S. Milewski. Meshless finite difference method with higher order approximation applications in mechanics. *Arch. Comput. Method. E.*, 19:1–49, 2012.

[61] S. Milewski. Selected computational aspects of the meshless finite difference method. *Numer. Algorithms*, 63(1):107–126, 2013.

[62] W. F. Mitchell and M. A. McClain. A comparison of hp-adaptive strategies for elliptic partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 41(1), 2014.

[63] D. Moxey, M. Green, S. Sherwin, and J. Peiró. An isoparametric approach to high-order curvilinear boundary-layer meshing. *Comput. Methods in Appl. Mech. Eng.*, 283:636–650, 2015.

[64] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Comput. Mech.*, 10(5):307–318, 1992.

[65] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[66] A. T. Patera. A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *J. Comput. Phys.*, 54(3):468–488, 1984.

[67] J. Peraire and P.-O. Persson. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM J. Sci. Comput.*, 30(4):1806–1824, 2008.

[68] N. Perrone and R. Kao. A general finite difference method for arbitrary meshes. *Comput. Struct.*, 5(1):45–57, 1975.

[69] P.-O. Persson and J. Peraire. Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, volume 204, 2009.

[70] F. U. Prieto, J. J. Benito Muñoz, and L. G. Corvinos. Application of the generalized finite difference method to solve the advection–diffusion equation. *J. Comput. Appl. Math.*, 235(7):1849–1855, 2011.

[71] W. Reed and T. Hill. Triangular mesh methods for the neutron transport equation. *Los Alamos Report LA-UR-73-479*, 1973.

[72] Y. Saad. ILUT: A dual threshold incomplete LU factorization. *Numer. Linear Algebr.*, 1(4):387–402, 1994.

[73] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2003.

[74] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7(3):856–869, 1986.

[75] C. Schwab. *p- and hp- Finite Element Methods: Theory and Applications in Solid and Fluid Mechanics*. Oxford University Press, Oxford, 1998.

[76] R. Sevilla, S. Fernández-Méndez, and A. Huerta. NURBS-enhanced finite element method (NEFEM). *Int. J. Numer. Meth. Engrg.*, 76(1):56–83, 2008.

[77] M. S. Shephard and M. K. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *Int. J. Numer. Meth. Engrg.*, 32(4):709–749, 1991.

[78] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering*, pages 203–222. Springer, 1996.

[79] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Comput. Geom.*, 22(1):21–74, 2002.

[80] H. Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Software*, 41(2):11:1 – 11:36, 2015.

[81] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences. Technical report, NASA, 2014.

[82] P. Šolín, J. Červenỳ, and I. Doležel. Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM. *Math. Comput. Simulat.*, 77(1):117–132, 2008.

[83] J. Stricklin, W. Ho, E. Richardson, and W. Haisler. On isoparametric vs linear strain triangular elements. *Int. J. Numer. Meth. Engng.*, 11(6):1041–1043, 1977.

[84] B. Szabo and A. Mehta. p-convergent finite element approximations in fracture mechanics. *Int. J. Numer. Meth. Engng.*, 12(3):551–560, 1978.

[85] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, San Diego, 2000.

[86] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, 13(2):631–644, 1992.

[87] R. S. Varga. Factorization and normalized iterative methods. Technical report, Westinghouse Electric Corp. Bettis Plant, Pittsburgh, 1959.

[88] P. Vincent and A. Jameson. Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists. *Math. Model. Nat. Phenom.*, 6(3):97–140, 2011.

[89] D. Wang, B. Clark, and X. Jiao. An analysis and comparison of parameterization-based computation of differential quantities for discrete surfaces. *Comput. Aid. Geom. Des.*, 26(5):510–527, 2009.

[90] Z. Wang. High-order methods for the Euler and Navier–Stokes equations on unstructured grids. *Prog. Aerosp. Sc.*, 43(1):1–41, 2007.

[91] D. S. Watkins. *Fundamentals of Matrix Computations*. John Wiley & Sons, New York, 2004.

[92] N. Zander, T. Bog, S. Kollmannsberger, D. Schillinger, and E. Rank. Multi-level hp-adaptivity: high-order mesh adaptivity without the difficulties of constraining hanging nodes. *Comput. Mech.*, 55(3):499–517, 2015.

[93] O. Zienkiewicz, R. Taylor, and J. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann, Oxford, 2013.

[94] M. Zlámal. Curved elements in the finite element method. I. *SIAM J. Numer. Anal.*, 10(1):229–240, 1973.

[95] M. Zlámal. The finite element method in domains with curved boundaries. *Int. J. Numer. Meth. Engrg.*, 5(3):367–373, 1973.