A Flexible Galerkin Finite Element Method with an *A Posteriori* Discontinuous Finite Element Error Estimation for Hyperbolic Problems

Thomas Christopher Massey

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mathematics

Slimane Adjerid, Chair
Christopher Beattie
Jeffrey T. Borggaard
Lee Johnson
Robert C. Rogers

Date: July 3, 2002
Blacksburg, Virginia

Keywords: Finite Elements, Flexible Discontinuous Galerkin,
A Posteriori Error Estimation, Hyperbolic Partial Differential Equations.

A Flexible Galerkin Finite Element Method with an *A Posteriori* Discontinuous Finite Element Error Estimation for Hyperbolic Problems

Thomas Christopher Massey

ABSTRACT

A Flexible Galerkin Finite Element Method (FGM) is a hybrid class of finite element methods that combine the usual continuous Galerkin method with the now popular discontinuous Galerkin method (DGM). A detailed description of the formulation of the FGM on a hyperbolic partial differential equation, as well as the data structures used in the FGM algorithm is presented. Some *hp*-convergence results and computational cost are included. Additionally, an a posteriori error estimate for the DGM applied to a two-dimensional hyperbolic partial differential equation is constructed. Several examples, both linear and nonlinear, indicating the effectiveness of the error estimate are included.

Acknowledgments

First and foremost, I want to give thanks to God, without whom I could have never achieved this goal.

I want to thank my advisor, Slimane Adjerid. He has always encouraged and supported me in the most positive of ways. He has been a kind, faithful and wise mentor, and I appreciate him greatly.

I want to thank the other members of my committee for their efforts in reviewing my work, especially Lee Johnson. I also want to thank all the teachers who have been a part of my life and encouraged me to achieve my potential.

I also want to thank my family and friends who have believed in me and encouraged me over these last six years, I love you all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Introduction and Background

Computer hardware technology and algorithm design has improved greatly over the past several years. As a result, the numerical models of physical phenomena that were being used can now give results faster and cheaper than ever before. Beyond this, with the increased speeds and storage capacities, we are able to include more variables in our models and in so doing are able to model more realistic scenarios. For example, in meteorology a few years ago the mathematical models and necessary data were in place to give a fairly reliable weather forecast one week into the future; however due to the slowness of the computers, it took longer than a week to run the numerical model. Today we are given a weeks forecast nightly on the local news. Science and engineering has relied on numerical modeling to simulate and analyze complicated systems for many years. The mathematical models most often used in the scientific arena come in the form of differential or integral equations. The finite element method is a general technique for numerically approximating the solution of such equations and was introduced by engineers in the late 1950's to solve the partial differential equations in structural mechanics. The name finite elements comes from the idea that in the original models for beams, plates, etc., the structure was subdivided into finitely many small parts or elements. Since that time, the method has been developed into a general way of approximately solving partial differential equations.

Several finite element formulations can exist for a given differential equation, but as Johnson [59] states, in general one must i) transform the differential equation into a variational problem, ii) discretize the infinite variable problem to a finite dimensional space, iii) solve the discrete problem using some kind of optimization method, and iv) implement the method on a computer. The term Galerkin finite element method is used when the trial and test spaces used in steps i) and ii) are the same space. The original Galerkin finite element methods were designed in such a way that the approximate solution was forced to be continuous across element boundaries and as a result are sometimes called conforming methods. We will, however, refer to such methods as continuous Galerkin methods (CGM). These meth-

ods have been used successfully in many applications including meteorology, oceanography, structural mechanics and the oil industry to name a few. In contrast to the CGM, a discontinuous Galerkin method (DGM) was developed in 1973 by Reed and Hill [72] for the neutron transport equation

$$\sigma u + \nabla \cdot (\boldsymbol{\alpha} u) = f, \tag{1.1}$$

where σ is a real number and $\boldsymbol{\alpha}$ is a constant vector. The DGM does not require the approximate solutions to be continuous across element boundaries, it instead has a flux term to account for the discontinuities. Because of this formulation the DGM can be computed explicitly. In fact the DGM has several features that make it very appealing:

- DGM can very easily handle complicated geometries, including hanging nodes.
- DGM are highly parallelizable since only neighboring elements are required to compute the solution on the current element.
- DGM are readily suited to adaptive strategies since refining and derefining the mesh can be done without regard to the continuity required by the CGM.
- DGM require no special treatment of the boundary conditions to maintain high order accuracy.

The idea of allowing discontinuous solutions across elements was not unique to hyperbolic problems, it was independently developed and used for parabolic and elliptic problems in the so called internal penalty methods, see [42]. To the present, the DGM has been used successfully to solve hyperbolic [15, 17, 16, 18, 34, 32, 39, 51], parabolic [44, 45], and elliptic [13, 12, 88] partial differential equations, to list just a few references.

Regardless of the type of finite element method, we need to know how well our computed solution will approximate the exact solution to our problem. For these reasons, *a priori* error estimates have been developed for both CGM and DGM and provide some initial guidance for deciding on the degree of the approximation to use and the size of the mesh to use, to maintain a prescribed level of accuracy. However, realistic problems in science and engineering have events that develop, evolve, and decay on diverse temporal and spatial scales and a simple implementation of the methods mentioned above would require excessive computing resources or fail to accurately resolve nonuniform behavior. The preferred approach is to use an adaptive *hp*- method. Both the CGM and the DGM have adaptive strategies which automatically refine, coarsen, or relocate meshes (and hence the $h-$ refinement) and vary the order of the numerical approximation (and hence the $p-$ refinement). As Adjerid *et al.* [2] state, these adaptive methods offer greater efficiency, reliability and robustness over the usual methods.

A posteriori error estimates have been an integral part of every adaptive method and are used to guide the adaptive process by indicating regions where more or less resolution is needed, to assess the quality of the solution and to stop the adaptive process. Flaherty [50] divides *a posteriori* error estimates into four broad classifications:

- *Residual error estimates* are found by solving local finite element problems created on either an element or a subdomain.
- *Flux-projection error estimates* are found by calculating a new flux via a post processing of the finite element solution and then taking the difference of the new smoother flux and the original flux.
- *Extrapolation error estimates* are found by taking the difference of finite element solutions having different orders or different meshes.
- *Interpolation error estimates* are used with estimates of the unknown constants.

These four techniques are not necessarily independent of one another, and regardless of the classification, a good *a posteriori* error estimation procedure should:

- give an accurate measure of the discretization error for a wide range of mesh spacing and polynomial degrees.
- be inexpensive to compute relative to the cost of obtaining the finite element solution.
- provide an estimate that is computable in several norms.

The *a posteriori* error estimate developed in this dissertation is a residual error estimate.

We are interested in hyperbolic problems, so we will give a brief summary of the development of the DGM along with some *a posteriori* error estimates and superconvergence results as they relate to the work presented here. A more complete review can be found in [31] and the references therein. In 1974 LeSaint and Raviart [70] published the first mathematical analysis of the DGM in the form of an *a priori* error estimate. They established a rate of convergence of $O(h^p)$ in the $L_2(\Omega)$-norm for general triangulations and a rate of $O(h^{p+1})$ for tensor products of polynomials of degree p in one variable defined on Cartesian grids (assuming a smooth solution). In 1986, Johnson and Pitkaränta [60] proved a rate of convergence of $O(h^{p+\frac{1}{2}})$ for general triangular meshes. In 1988, Richter [74] successfully demonstrated the optimal rate of convergence $O(h^{p+1})$ for some structured two-dimensional non-Cartesian grids. Since then, studies have shown convergence to the weak solution assuming much less regularity in the exact solution, see references in [31]. In the early 1990's investigations of *a posteriori* error estimates for the DGM were undertaken and the first *hp- a posteriori* error estimate for the DGM was given by Bey and Oden [15]. In 1995 Bey *et al.* [17] used these error estimates to devise parallel adaptive strategies for the DGM applied to linear scalar hyperbolic conservation laws.

For nonlinear systems of conservation laws, variations of the original DGM such as the Runge-Kutta discontinuous Galerkin methods (RKDG) were introduced by Cockburn and Shu [35, 34]. They give an explicit scheme and use a discontinuous finite element formulation in space and a special TVD Runge-Kutta time discretization, all in conjunction with a

generalized slope limiter, where TVD means total variation diminishing. In 1998, Cockburn and Shu [36] introduced the local discontinuous Galerkin method, LDG, for time-dependent convection-diffusion systems. Cockburn and Shu [36] state that the LDG method is an extension of the RKDG methods for purely hyperbolic systems. They further state that the basic idea for constructing the LDG method is to suitably rewrite the convection-diffusion system into a larger, degenerate, first-order system and then discretize it by the RKDG method.

In 1994 Biswas *et al.* [18] developed a parallel finite element method for hyperbolic conservation laws in one and two dimensions using the DGM to discretize in space with a basis of piecewise Legendre polynomials. In this work they also discovered evidence of superconvergence of the DGM approximate solution at the Radau points and used this result along with p-refinement to construct efficient *a posteriori* error estimates in space. Lowrie [67] showed numerical evidence of an order $O(h^{2p+1})$ accurate component of the DGM approximate solution. Cockburn *et al.* [33] used a post-processing technique, involving a convolution with a kernel, applied only once to the approximate solution, to obtain a rate of convergence of $O(h^{2p+1})$ on Cartesian grids when the exact solution is globally smooth. Adjerid *et al.* [5] proved that smooth DGM solutions of one-dimensional hyperbolic problems using p-degree polynomial approximations have an $O(h^{p+2})$ superconvergence rate at the roots of a Radau polynomial of degree $p+1$. They used this result to construct asymptotically correct *a posteriori* error estimates. They further established a strong $O(h^{2p+1})$ superconvergence at the downwind end of every element. Krivodonova *et al.* [63] proved a superconvergence result on average on the outflow edge of every element of unstructured triangular meshes and constructed *a posteriori* error estimates that converge to the true error under mesh refinement.

Recently, Alotto *et al.* [9], Perugia and Schötzau [71], and Dawson and Proft [38] have coupled continuous and discontinuous Galerkin methods. Alotto *et al.* [9] developed these coupled methods for use in the simulation of rotating electrical machines. Here during time stepping, the relative movement of two meshes associated with the two different regions of the electrical device (rotor and stator) results in the generation of hanging nodes on the slip surface. They wanted to easily handle the hanging nodes located on the slip surface by using the LDG and then use a CGM on the rest of the parts. Perugia and Schötzau [71] furthered this work in the context of an elliptical problem and developed an optimal *a priori* error estimate for arbitrary meshes with hanging nodes. They coupled the LDG and CGM by using DGM techniques to create a suitable numerical flux. Dawson and Proft [38] developed a coupled method for the numerical solution of convection-diffusion (transport) equations, where convection may be dominant. Again, the idea was to decompose the domain into two regions, one where the LDG was used and another where the CGM was used and then to define a suitable coupling to transmit solution information across the dividing boundaries. They were able to derive stability and *a priori* error estimates.

Others application where the DGM and CGM are combined exist and include, for example, modeling of coastal circulation where currently the Navy uses a program called ADCIRC, see Luettich *et al.* [68] that uses a continuous finite element discretization in space. According to

Blain and Cobb [19] this model performs well in most circumstances. However when trying to model rip currents where the flow is highly advective, or intertidal zones where the geometry is complicated and the water very shallow, Blain and Cobb [19] report that the ADCIRC model does not conserve mass quantities well. As a result, new models using a DGM in space have been developed for costal modeling, see Chippada *et al.* [26]. These models have shown promise in solving the problem of mass conservation. The Navy is considering a coupled method in the same spirt as the method in [38] where they will run the ADCIRC model from "safe" water regions toward the coast up to a specified depth, say, and then switch over and use the derived solution from ADCIRC to start up a DGM method which would complete the computation up onto the shorelines.

The flexible Galerkin finite element method (FGM) for solving partial differential equations is a hybrid of the traditional Galerkin finite element methods (CGM) which require the discrete solution to be continuous across elements and the now popular discontinuous Galerkin method (DGM) which does not have such a requirement. The FGM allows the user to prescribe for each element both the degree of the finite element approximation to be used and the level of continuity to be enforced between elements. Therefore, if one wants to run the FGM so that full continuity is enforced, (so that the approximations are continuous across elements and in so doing effectively having a CGM), this is not only possible but easily achieved. Likewise, implementing the FGM so that no continuity is enforced between elements, i.e., the DGM, is easily achieved. In this way, the FGM encompasses both the DGM and the CGM. The real novelty of the FGM is in allowing varying levels of continuity to be enforced. For example, suppose the true solution develops a discontinuity somewhere in the domain. Then it would make sense to allow the approximation used in that region to be discontinuous (thus potentially mimicking the solution itself) while allowing the rest of the approximation to be continuous. As another example, suppose we want the approximate solution to be continuous in one dimension (say time) and discontinuous in the second (say space). This also is achieved easily by appropriately defining the input for the data structures. Additionally, running test scenarios to see how not enforcing continuity in time, or enforcing continuity in space changes the solution is very simple. In particular, we need only to adjust the input for the FGM data structures and we need not reformulate the problem or run a different routine.

The examples from [9, 71, 38] show the usefulness of a model that can automatically adjust between continuous and discontinuous approximations within a region, as does the FGM. But again the FGM is more than just a coupling method. It also has the flexibility to allow for levels of continuity to be enforced between elements. For example, suppose that we are using a fourth degree finite element approximation on our mesh. Then with the FGM it is possible to assign the level of continuity enforced across the boundary adjoining two elements to be quadratic. This means that the constant, linear and quadratic components of the approximation on each involved element are continuous, while the cubic and quartic components have no continuity enforced on them.

1.2 Organizational Outline

This dissertation is organized as follows: The remaining parts of Chapter 1 specify the types of problems we consider, introduce most of our notation, and recall some theory on the method of characteristics. In Chapter 2 we formulate the flexible Galerkin method and describe the basis functions. We also provide examples illustrating the error patterns, rates of convergence under mesh refinement, and cost comparisons for the various levels of continuity enforced. In Chapter 3 we present some computational aspects of the FGM, including the necessary data structures and numerical integration techniques, and a discussion of how to compute the $\mathcal{L}_2$ norm of the error on elements where a discontinuous true solution exists. In Chapter 4 we extend the one-dimensional results of Adjerid *et al.* [5] to multi-dimensional hyperbolic problems on rectangular meshes, prove some superconvergence results, and construct *a posteriori* error estimates for the discontinuous Galerkin method. Chapter 5 presents several numerical examples to illustrate the theory from Chapter 4. Chapter 6 states the contributions of this work and suggests future work.

1.3 Problem Specification and Notation

The motivation for the present work is to develop a new class of finite element methods, called the flexible Galerkin methods, FGM, that can be applied to approximately solve general systems of conservation laws where we adopt most of Evans' [47] notation

$$\mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) = 0 \quad \text{in } \mathbb{R}^n \times (0, \infty), \tag{1.2a}$$

$$\mathbf{u} = \mathbf{g} \quad \text{on } \mathbb{R}^n \times t = 0. \tag{1.2b}$$

In (1.2) the components of

$$\mathbf{u} = \mathbf{u}(x,t) = (u^1(x,t), \ldots, u^m(x,t)) \quad (\mathbf{x} \in \mathbb{R}^n, t \geq 0) \tag{1.3}$$

are the densities of various conserved quantities and $\nabla \cdot \mathbf{F}(\mathbf{u})$ is the divergence of $\mathbf{F}(\mathbf{u})$. The rate of change of these densities within a bounded region Ω is given by the flux function

$$\mathbf{F} : \mathbb{R}^m \to \mathbb{R}^m \tag{1.4}$$

which controls the rate of change of $\mathbf{u}$ through the boundary of Ω. The function $\mathbf{g}$ describes the initial condition of $\mathbf{u}$.

Evans [47] lists some examples of such systems of conservation laws that are of particular importance to us since they represent nontrivial instances in which the FGM could be used to approximate their solution. For instance *Euler's equations* for compressible gas flow in

one dimension can be written as

$$\rho_t + (\rho v)_x = 0 \quad \text{(conservation of mass)} \tag{1.5a}$$

$$(\rho v)_t + (\rho v^2 + p)_x = 0 \quad \text{(conservation of momentum)} \tag{1.5b}$$

$$(\rho E)_t + (\rho E v + p v)_x = 0 \quad \text{(conservation of energy)} \tag{1.5c}$$

in $\mathbb{R} \times (0, \infty)$, where ρ is the mass density, v the velocity, E the energy density per unit mass. Evans assumes $E = e + \frac{v^2}{2}$ where e is the internal energy per unit mass and also that the pressure $p = p(\rho, e)$ is a known function of ρ and e.

We will confine our discussion in this work to two types of model problems, one linear with constant coefficients and the other nonlinear, but both two-dimensional first order hyperbolic problems. We will use the following formulations:

The model linear problem is a constant coefficient first order problem that takes the form:

$$\boldsymbol{\alpha} \cdot \nabla u + \gamma u = f(x, y) \qquad \text{in } \Omega = [0, 1] \times [0, 1], \tag{1.6a}$$

subject to the boundary conditions at the inflow boundaries

$$u(x, 0) = g_1(x), \quad u(0, y) = g_2(y) \tag{1.6b}$$

where γ is a scalar constant and $\boldsymbol{\alpha}$ is a constant vector with $\alpha_i > 0, i = 1, 2$. We assume that $f(x, y)$, $g_1(x)$, and $g_2(y)$ are such that the problem (1.6) admits a unique solution $u(x, y) \in C^\infty(\Omega)$.

The model nonlinear problem is also a first-order problem and takes the form:

$$f_x(u) + g_y(u) = h(x, y), \qquad \text{on} \quad \Omega = [0, 1] \times [0, 1], \tag{1.7a}$$

subject to the boundary conditions at the inflow boundaries

$$u(x, 0) = \phi_1(x) \qquad \text{and} \qquad u(0, y) = \phi_2(y). \tag{1.7b}$$

Again we assume that if $h(x, y)$, $\phi_1(x)$ and $\phi_2(y)$ are such that (1.7) admits a unique solution $u(x, y) \in C^\infty(\Omega)$.

In both model problems we denote the inflow boundary by $\partial\Omega_{\text{in}} \subset \partial\Omega$ and the outflow boundary by $\partial\Omega_{\text{out}} \subset \partial\Omega$ and note that $\partial\Omega_{\text{in}} \cup \partial\Omega_{\text{out}} = \partial\Omega$.

To determine inflow and outflow let $\boldsymbol{n}$ denote the unit outward normal to $\partial\Omega$. Then for the linear model,

$$\partial\Omega_{\text{in}} = \{(x, y) \in \partial\Omega \mid \boldsymbol{\alpha} \cdot \boldsymbol{n} < 0\}, \tag{1.8}$$

$$\partial\Omega_{\text{out}} = \{(x, y) \in \partial\Omega \mid \boldsymbol{\alpha} \cdot \boldsymbol{n} \geq 0\}. \tag{1.9}$$

and for the nonlinear model,

$$\partial\Omega_{\text{in}} = \{(x,y) \in \partial\Omega \mid [f'(u), g'(u)] \cdot \boldsymbol{n} < 0\}, \tag{1.10}$$

$$\partial\Omega_{\text{out}} = \{(x,y) \in \partial\Omega \mid [f'(u), g'(u)] \cdot \boldsymbol{n} \geq 0\}. \tag{1.11}$$

We now present most of the other notation used in this dissertation. We begin with the gradient of a function $U = U(x,y)$ which is

$$\nabla U = \frac{\partial U}{\partial x}\mathbf{i} + \frac{\partial U}{\partial y}\mathbf{j}. \tag{1.12}$$

The divergence of a differentiable vector function $\mathbf{F}(x,y) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix}$ is

$$\nabla \cdot \mathbf{F} = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y}. \tag{1.13}$$

The Jacobian of a differentiable vector function $\mathbf{F}(x,y)$ is

$$\mathbf{J}[\mathbf{F}(x,y)] = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}. \tag{1.14}$$

The determinant of a matrix function $\mathbf{M}$ is denoted $\det(\mathbf{M})$.

The L_2 inner product of two integrable functions f and g is

$$\langle f, g \rangle_2 = \int_{-1}^{1} f(x)g(x)dx \tag{1.15}$$

and the subsequent induced norm is

$$\|f\|_2 = \sqrt{\langle f, f \rangle_2}. \tag{1.16}$$

The weighted L_2 inner product of two integrable functions f and g is

$$\langle f, g \rangle_{2,w} = \int_{-1}^{1} w(x)f(x)g(x)\ dx \tag{1.17}$$

and subsequent induced norm is

$$\|f\|_{2,w} = \sqrt{\langle f, f \rangle_{2,w}}. \tag{1.18}$$

The $\mathcal{L}_2$ norm of a function $f(x,y)$ over a region Ω is

$$\|f\|_{\mathcal{L}_2(\Omega)} = \left(\iint_{\Omega} f^2(x,y)\ dxdy \right)^{\frac{1}{2}}. \tag{1.19}$$

The jump in a function f at a point x_* is

$$[[f(x_*)]] = (f(x_*^+) - f(x_*^-)) \tag{1.20a}$$

where

$$x_*^+ = \lim_{\delta x \to 0} x_* + \delta x, \tag{1.20b}$$

$$x_*^- = \lim_{\delta x \to 0} x_* - \delta x. \tag{1.20c}$$

The unit outward normal to a boundary is $\boldsymbol{n}$.

The solution U^- from an element's inflow boundary Γ_{in} is

$$U^- = \lim_{s \to 0^+} U((x,y) + s\boldsymbol{n}). \tag{1.21}$$

The greatest integer function is

$$ceil(x) = n, \quad \text{such that } n-1 < x \leq n, \quad n \text{ is an integer.} \tag{1.22}$$

The space of monic polynomials of degree n is

$$\bar{\Pi}_n = \{p \mid p(\xi) = \xi^n + c_{n-1}\xi^{n-1} + \cdots + c_0\}. \tag{1.23}$$

The space of all polynomials of degree k or less is

$$\mathcal{P}_k = \{q | q = \sum_{m=0}^{k} \sum_{i=0}^{m} c_i^m x^i y^{m-i}\}. \tag{1.24}$$

The smallest space $\mathcal{V}_p$ of polynomial functions such that

$$\mathcal{P}_{p+1} \subset \mathcal{V}_p \cup \{x^{p+1}, y^{p+1}\}, \qquad p \geq 0 \tag{1.25}$$

is

$$\mathcal{V}_p = \{V | V = \sum_{k=0}^{p} \sum_{i=0}^{k} c_i^k x^i y^{k-i} + \sum_{i=1}^{p} c_i^{p+1} x^i y^{p+1-i}\}. \tag{1.26}$$

We will define any other notation as needed.

1.4 Method of Characteristics

First order hyperbolic partial differential equations are used to model conservation of physical quantities such as mass, momentum, and energy in a fluid flow. For instance, consider a conservation law in one space dimension of the form

$$\frac{d}{dt}\int_{\alpha}^{\beta} u\ dx = -f(u)|_{\alpha}^{\beta}, \tag{1.27}$$

where u and $f(u)$ are density and flux, respectively. This conservation law states that the rate of change of u within the volume $\alpha \leq x \leq \beta$, is equal to the change in flux through the boundaries, $x = \alpha$ and $x = \beta$. If both f and u are smooth, then (1.27) can be written as

$$\int_{\alpha}^{\beta} (u_t + f_x(u))dx = 0. \tag{1.28}$$

Since this must hold for all possible volumes, the integrand must vanish, so that

$$u_t + f_x(u) = 0. \tag{1.29}$$

We can rewrite (1.29) into its convective form as

$$u_t + a(u)u_x = 0, \tag{1.30a}$$

where

$$a(u) = \frac{df(u)}{du}. \tag{1.30b}$$

The term u_x is called the convection term and represents the movement of the quantity by means of the movement of the medium. The term $a(u)$ gives the velocity of the medium while the flux of a quantity (from left to right) across a point due to convection is given by $f_x(u) = a(u)u_x$. For example, Farlow [48] considers dumping some pollutants into a clean river moving at a velocity given by $a(u)$ and then observing the concentration of the pollutant at a distance x downstream. If the pollutant does not diffuse with the running water, the concentration of the pollutant is given by the solution of (1.29) subject to

$$u(0,t) = P \qquad \text{constant input of the pollutants into the river} \tag{1.31}$$

$$u(x,0) = 0 \qquad \text{no pollutants in the river initially.} \tag{1.32}$$

Back to the general case, for simplicity we consider the initial value problem in convective form

$$u_t + a(u)u_x = 0, \tag{1.33a}$$

subject to

$$u(x,0) = \phi(x), \qquad -\infty < x < \infty. \tag{1.33b}$$

Now consider the rate of change of $u(x(t),t)$ as measured by a moving observer, $x = x(t)$. Then the total derivative of u would be

$$du = u_t dt + u_x dx = \left(u_t + \frac{dx}{dt} u_x \right) dt \tag{1.34}$$

or, by the chain rule,

$$\frac{du}{dt} = u_t + \frac{dx}{dt} u_x. \tag{1.35}$$

If the observer moves with velocity $a(u)$ then of course

$$\frac{dx}{dt} = a(u). \tag{1.36}$$

Using (1.33a) in (1.35) leads to

$$\frac{du}{dt} = u_t + \frac{dx}{dt} u_x = 0 \tag{1.37}$$

which implies that u is a constant along the curve $\frac{dx}{dt} = a(u)$. This curve is called a characteristic curve, characteristic line, or simply a characteristic of the differential equation (1.33a). Having (1.37), we have replaced the partial differential equation (1.33a) with two coupled ordinary differential equations,

$$\frac{dx}{dt} = a(u), \tag{1.38a}$$

$$\frac{du}{dt} = 0. \tag{1.38b}$$

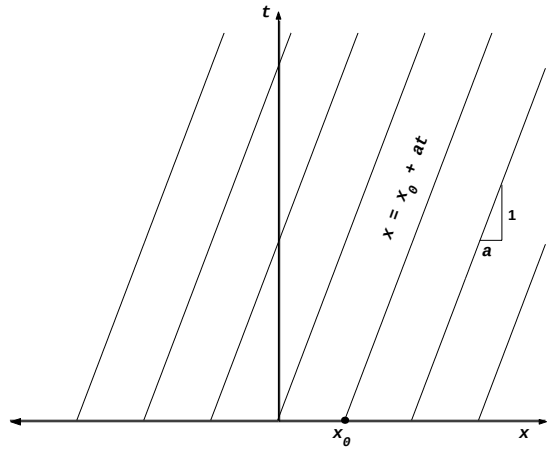
Since u is constant along a characteristic then it must have the same value it had initially. So consider a characteristic that passes through the point $(x_0, 0)$. From the initial conditions we have $u(x_0, 0) = \phi(x_0)$ and thus $u = \phi(x_0)$ along the characteristic. This particular characteristic satisfies the ODE

$$\frac{dx}{dt} = a(\phi(x_0)) = a_0, \tag{1.39}$$

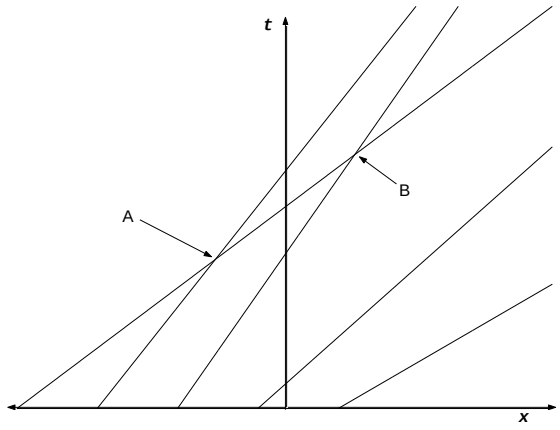
which states that the slope of the characteristic is a constant, a_0. Thus the characteristic is a straight line given by

$$x(t) = x_0 + a_0 t. \tag{1.40}$$

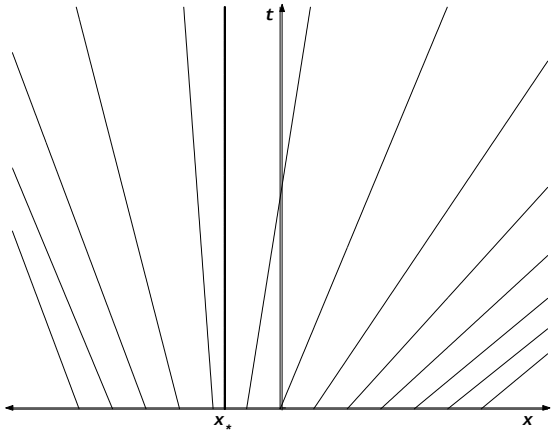
In fact, each characteristic is a straight line but with different slopes resulting from different initial values. This means that characteristic lines can be parallel, can intersect one another, or diverge from one another. We get parallel characteristics when $a(u)$ is a constant. For example, if $a(u) = \frac{1}{2}$, then all the characteristics are parallel and move with constant velocity $\frac{1}{2}$. Such characteristic lines have slope 2 as shown in Figure 1.1(A). Figure 1.1(B) shows intersecting characteristics lines at the points A and B. Figure 1.1(C) shows characteristic



(A)



(B)



(C)

Figure 1.1: Different characteristic lines.

lines that "fan out", since the slope of the characteristics, $a(u)$, take on both positive and negative values as well as a value of zero at the point x_*.

When characteristics do not intersect, the unique and continuous solution of (1.33a) is given implicitly by $u(x,t) = \phi(x - a(\phi(x_0))t)$ where the space-time point (x,t) lies on the characteristic with parameter x_0. Accordingly, to find the solution at a point (x,t) we must determine on which characteristic it lies and then trace that characteristic back to the initial value. Then, since u is constant along characteristics, the value of u at (x,t) is the same as u at $(x_0, 0)$.

But what happens if characteristic lines intersect? Which characteristic do you follow back to $t = 0$? Can the solution u be multivalued at a point where characteristics intersect? To answer this, consider modeling traffic flow of cars, can the density of cars at a single point take on more than one value? Physically this can not happen. How do we solve such a problem? The answer is simply that the solution u must break or be discontinuous. When characteristics intersect the solution is said to have a moving discontinuity, called a shock wave or simply a shock. These shocks can form immediately (as is the case where the initial condition is discontinuous) or they can form at some finite time later. Using the integral form (1.27) of (1.29) allows us to assume that u and $f(u)$ can have jump discontinuities. To see how this works, let x_s be the position where the shock forms and propagates in time, so that $x_s(t)$ is a smooth function of time with a shock velocity given by $\frac{dx_s}{dt}$. Let x_{s-} and x_{s+} be the position of the shock on the left and right of the discontinuity. We note that on either side of the shock (1.33a) holds, since u and f are assumed to be smooth there. Now for each time $t > 0$, we assume that $\alpha < x_s(t) < \beta$ and from (1.27) we get

$$\frac{d}{dt}\int_\alpha^\beta u dx = \frac{d}{dt}\left[\int_\alpha^{x_{s-}} u dx + \int_{x_{s-}}^\beta u dx\right] = -f(u)|_\alpha^\beta \,. \tag{1.41}$$

By differentiating we get

$$\int_\alpha^{x_{s-}} u_t dx + u(x_{s-})\frac{dx_{s-}}{dt} + \int_{x_{s-}}^\beta u_t dx + u(x_{s+})\frac{dx_{s+}}{dt} = -f(u)|_\alpha^\beta \,. \tag{1.42}$$

On either side of the shock the solution is assumed to be smooth. Thus (1.29) is valid and can replace the integrals. Using the smoothness of $x_s(t)$, $\frac{dx_{s-}}{dt} = \frac{dx_{s+}}{dt} = \frac{dx_s}{dt}$, we get

$$-f(u)|_\alpha^{x_{s-}} + u(x_{s-})\frac{dx_s}{dt} + -f(u)|_{x_{s+}}^\beta + u(x_{s+})\frac{dx_s}{dt} = -f(u)|_\alpha^\beta \tag{1.43}$$

which simplifies to

$$\frac{dx_s}{dt}\left(u(x_{s+}) - u(x_{s-})\right) = \left(f(u(x_{s+})) - f(u(x_{s-}))\right). \tag{1.44}$$

Using jump notation, as defined in (1.20), we get

$$[[u]]\frac{dx_s}{dt} = [[f]], \tag{1.45}$$

which is known as the *Rankine-Hugoniot* jump condition and can be used to give a discontinuous solution u. Notice that the shock has velocity

$$\frac{dx_s}{dt} = \frac{[[f]]}{[[u]]}. \tag{1.46}$$

In order to determine where the shock forms, x_s, we use the fact that shocks form where characteristic lines intersect. Considering two neighboring characteristics, one derived from x_0, the other from $x_0 + \Delta x$, both at time $t = 0$, we obtain

$$x = x_0 + a(\phi(x_0))t, \tag{1.47a}$$

$$x = x_0 + \Delta x + a(\phi(x_0 + \Delta x))t. \tag{1.47b}$$

These two characteristics only intersect at a positive time if $a(\phi(x_0)) > a(\phi(x_0 + \Delta x))$. Solving for the intersection point by setting (1.47a) equal to (1.47b) and solving for t we get

$$t = \frac{1}{(a(\phi(x_0)) - a(\phi(x_0 + \Delta x)))/\Delta x} \tag{1.48}$$

Taking the limit as $\Delta x \to 0$ we have

$$t = \frac{-1}{\frac{da}{dx_0}}. \tag{1.49}$$

From this we see that characteristics only intersect for $t > 0$ provided $\frac{da(\phi(x_0))}{dx_0} < 0$. To find the first time t that a shock forms we must minimize (1.49) over all times $t > 0$. This can be accomplished by solving

$$\frac{d^2}{dx_0^2}(a(\phi(x_0))) = 0. \tag{1.50}$$

Characteristics theory extends readily to more general nonhomogeneous partial differential equations and systems of conservation laws, see [47]. We use this theory to gain information about the problem, such as which direction information flows and the possibility of shocks forming. Knowing the direction of flow tells us where in the mesh to begin our computations since we need to progress through the elements in the direction of information flow. When we know *a priori* that a shock forms, we can employ techniques to improve the quality of the solution, such as doing an $hp-$refinement near the shock or using slope limiters to control spurious oscillations that can form and pollute the approximate solution. Additionally, adaptive methods can incorporate this theory to predict the location of a shock and follow its path, so that the adjustments mentioned earlier can be done automatically.

Chapter 2

Flexible Galerkin Formulation

2.1 Introduction

We begin the development of the FGM by first discretizing the domain Ω of either of the model problems defined in (1.6) and (1.7). To start we subdivide Ω into rectangular elements, Δ, by partitioning $[0,1]$ along the x-axis into n subintervals (x_{i-1}, x_i) and partition $[0,1]$ along the y-axis into m subintervals (y_{j-1}, y_j). We define the ij^{th} element Δ_{ij} by

$$\Delta_{ij} = (x_{i-1}, x_i) \times (y_{j-1}, y_j) \qquad i = 1, \ldots, n; j = 1, \ldots, m. \tag{2.1}$$

Clearly there will be $N = nm$ elements such that

$$\bigcup_{i,j=1}^{n,m} \bar{\Delta}_{ij} = \bar{\Omega}, \tag{2.2}$$

where $\bar{\Omega}$ denotes the closure of Ω. Furthermore, two elements Δ_{ij} and Δ_{kl} can intersect at a single vertex, along a single edge, or not at all.

Once the elements have been defined, we must then consider our approximation $U(x,y)$ to $u(x,y)$. The approximation $U(x,y)$ is a piecewise polynomial function. On each element, Δ_{ij}, $U(x,y)$ belongs to the finite dimensional space $\mathcal{V}_p$ of polynomial functions such that

$$\mathcal{P}_{p+1} \subset \mathcal{V}_p \cup \{x^{p+1}, y^{p+1}\}, \qquad p \geq 0. \tag{2.3}$$

In (2.3), $\mathcal{P}_k$ is the space of polynomials of degree k

$$\mathcal{P}_k = \{q \mid q = \sum_{m=0}^{k} \sum_{i=0}^{m} c_i^m x^i y^{m-i}\}. \tag{2.4}$$

These spaces are suboptimal but they lead to a very simple *a posteriori* error estimator which we present in Chapter 4. For efficiency reasons we consider the smallest spaces that

satisfy (2.3),

$$\mathcal{V}_p = \{\ V \mid V = \sum_{k=0}^{p} \sum_{i=0}^{k} c_i^k x^i y^{k-i} + \sum_{i=1}^{p} c_i^{p+1} x^i y^{p+1-i} \}. \tag{2.5}$$

We note that the polynomial degree p is allowed to vary from one element to the next and that tensor product elements satisfy (2.3). Thus $\mathcal{V}_p$ is just the space $\mathcal{P}_{p+1}$ with the terms x^{p+1} and y^{p+1} missing. The space $\mathcal{V}_p$ is easily visualized using Pascal's triangle, see Figures 2.1 and 2.2.

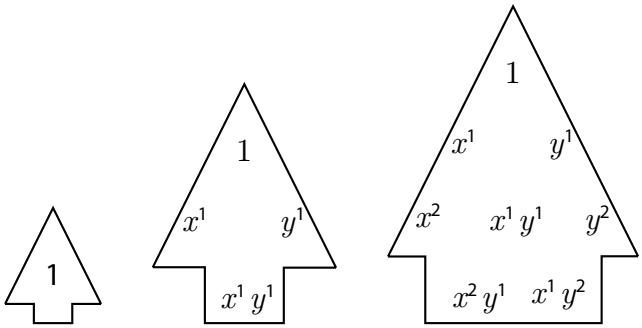


Figure 2.1: The spaces $\mathcal{V}_0$ to $\mathcal{V}_2$ (left to right).

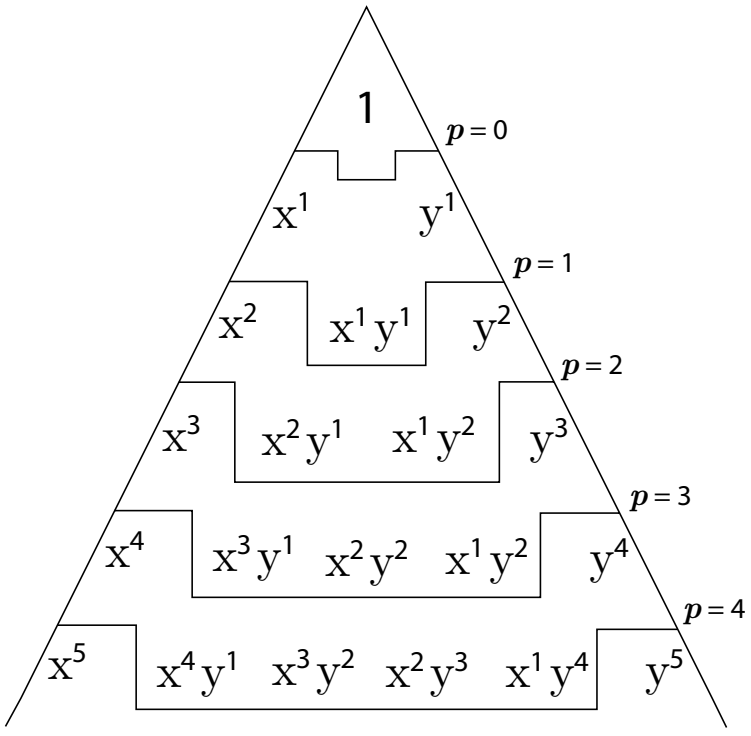


Figure 2.2: Different $\mathcal{V}_p$ shown as nested arrowheads inside of Pascal's triangle.

Our FGM is constructed on meshes having quadrilateral elements as shown in Figure 2.3. Thus on an element where our approximation $U(x, y)$ is not piecewise constant, it will consist

of three distinct components or modes: 1) vertex modes, 2) edge modes, and 3) interior modes, see Figure 2.3. Vertex and edge modes are the only modes involved when classifying continuity between elements. This is because the interior modes are constructed from shape functions that are zero on the boundaries and thus do not contribute to the approximation there. In the case where an element's approximation is just a constant, there is only a single value associated with that element. For each of the modes, we associate particular shape functions which are used to construct a basis for the approximating space $\mathcal{V}_p$. We discuss these shape modes and the basis functions in the following section.

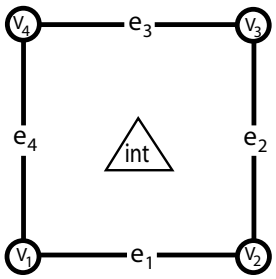


Figure 2.3: A quadrilateral element with its components labeled.

2.2 Basis Functions

In this section we mostly follow Flaherty's [50] construction of the standard hierarchical shape functions of Szabó and Babuška [85]. Using hierarchical shape functions to span the approximating space $\mathcal{V}_p$ has many advantages over other basis functions. Specifically, when compared to Lagrangian basis functions, we can construct a basis of degree $p+1$ by using our existing degree p basis and adding correction terms, namely the $(p+1)^{th}$ terms, instead of constructing an entirely new space. Additionally, hierarchical basis functions offer computational advantages in regard to round-off error associated with increasing polynomial degrees, see Szabó and Babuška [85].

The particular hierarchical shape functions we use are built from Legendre polynomials, $P_n(x)$. Legendre polynomials are orthogonal and are easily constructed using a three term recurrence. Also $P_n(-x) = (-1)^n P_n(x)$ and thus whenever n is odd, $P_n(x)$ is an odd function and hence

$$\int_{-1}^{1} P_n(x)dx = 0.$$

Legendre polynomials $P_n(x)$ of degree $n = 0, 1, 2, \cdots$, are solutions to Legendre's differential equation

$$(1-x^2)y'' - 2xy' + n(n+1)y = 0,$$

which has a solution given by Rodrigue's formula,

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n.$$

Table 2.1 list the first six Legendre polynomials and their derivatives. A more useful formulation of the Legendre polynomials is given by the three term recurrence formula,

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x).$$

Their derivatives are given by

$$P_{n+1}^{'}(x) = (2n+1)P_n(x) + P_{n-1}^{'}(x).$$

As stated earlier, the Legendre polynomials are orthogonal in that

$$\langle P_n, P_m \rangle_2 \triangleq \int_{-1}^{1} P_n(x)P_m(x)dx = \begin{cases} 0 & n \neq m \\ \frac{2}{2n+1} & n = m \end{cases}. \tag{2.6}$$

The following three term recurrence (which holds for orthogonal polynomials in general) can be used to compute the Legendre polynomials:

$$P_n(x) = (x - a_n)P_{n-1}(x) - b_nP_{n-2}(x), \qquad n \geq 2, \tag{2.7}$$

with $P_0(x) = 1$, and $P_1(x) = x - a_1$,

$$a_n = \frac{\langle xP_{n-1}(x), P_{n-1} \rangle_2}{\langle P_{n-1}(x), P_{n-1} \rangle_2}, \tag{2.8}$$

$$b_n = \frac{\langle xP_{n-1}(x), P_{n-2} \rangle_2}{\langle P_{n-2}(x), P_{n-2} \rangle_2}. \tag{2.9}$$

The computed values of these a_n's and b_n's for Legendre polynomials and other orthogonal polynomials are tabulated and can be found in mathematical reference tables. Furthermore, for coding purposes, the a_n's and b_n's can be computed accurately using numerical quadrature and then stored in a lookup table for later use. In terms of numerical quadrature, the Gaussian nodes used for Gaussian quadratures are the roots of the Legendre polynomials.

Lobatto polynomials are obtained from Legendre polynomials by

$$\phi_n(\xi) = \sqrt{\frac{2n-1}{2}} \int_{-1}^{\xi} P_{n-1}(t)dt, \qquad n \geq 1. \tag{2.10}$$

The Lobatto polynomials have the following recurrence relation

$$\phi_n(\xi) = \frac{P_n(\xi) - P_{n-2}(\xi)}{\sqrt{2(2n-1)}}, \qquad n \geq 2, \tag{2.11}$$

and the property that

$$\phi_n(-1) = \phi_n(1) = 0, \quad n \geq 2. \tag{2.12}$$

n	$P_n(x)$	$P_n^{'}(x)$
0	1	0
1	x	1
2	$\frac{1}{2}(3x^2-1)$	$3x$
3	$\frac{1}{2}(5x^3-3x)$	$\frac{3}{2}(5x^2-1)$
4	$\frac{1}{8}(35x^4-30x^2+3)$	$\frac{5}{2}(7x^3-3x)$
5	$\frac{1}{8}(63x^5-70x^3+15x)$	$\frac{15}{8}(21x^4-14x^2+1)$

Table 2.1: Legendre polynomials and their derivatives up to degree 5.

n	$\phi_n(\xi)$
1	$\sqrt{\frac{1}{2}}\ \ (\xi+1)$
2	$\sqrt{\frac{3}{8}}\ \ (\xi^2-1)$
3	$\sqrt{\frac{5}{8}}\ \ \xi(\xi^2-1)$
4	$\sqrt{\frac{7}{128}}\ \ (5\xi^4-6\xi^2+1)$
5	$\sqrt{\frac{9}{128}}\ \ \xi(7\xi^4-10\xi^2+3)$
6	$\sqrt{\frac{11}{512}}\ \ (21\xi^6-35\xi^4+15\xi^2-1)$

Table 2.2: Lobatto polynomials of degree $n = 1$ to 6.

Note that (2.12) means the function vanishes at the endpoints. As a result of the orthogonality property of Legendre polynomials, the Lobatto polynomials satisfy

$$\int_{-1}^{1} \frac{d\phi_n}{d\xi}\frac{d\phi_m}{d\xi}d\xi = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases}. \tag{2.13}$$

The first six Lobatto polynomials appear in Table 2.2 and their graphs in Figure 2.4.

The basis functions for the discontinuous Galerkin discretization error estimate, see §4.3, are computed using Radau polynomials. The Radau polynomials are defined using Legendre polynomials and are classified as either right Radau polynomials or left Radau polynomials. Right Radau polynomials are zero at the right endpoint of their domain and the left Radau polynomials are zero at the left endpoint of their domain. Let $P_n(\xi)$ be the usual Legendre polynomial of degree n. Define $R_n^+(\xi)$ to be the right Radau polynomial of degree n and $R_n^-(\xi)$ to be the left Radau polynomial of degree n,

$$R_n^{\pm}(\xi) = P_n(\xi) \mp P_{n-1}(\xi), \qquad n \geq 1. \tag{2.14}$$

We plot the first six right Radau polynomials in Figure 2.5. We also give the zeros of the right Radau polynomials up to degree 12, in Table 2.3. The values for these zeros were found using *Mathematica*'s *NSolve* function with maximum precision. We use these values when we perform Lagrange interpolation of the boundary conditions at the Radau points, as in (2.40).

As stated earlier, there are 3 types of modes in the hierarchical shape functions. Each of these modes is described here and the local indexing is given in Figure 2.3. We describe these shape modes on the canonical element $\{(\xi,\eta) \mid -1 \leq \xi,\eta \leq 1\}$ using the one dimensional linear functions

$$\bar{N}_1(\xi) = \frac{(1-\xi)}{2}, \tag{2.15}$$

$$\bar{N}_2(\xi) = \frac{(1+\xi)}{2}. \tag{2.16}$$

i) Vertex Modes

There are four vertex modes

$$\text{for vertex 1,} \qquad \Phi^1_{v_1}(\xi,\eta) = \bar{N}_1(\xi)\bar{N}_1(\eta), \tag{2.17}$$

$$\text{for vertex 2,} \qquad \Phi^1_{v_2}(\xi,\eta) = \bar{N}_2(\xi)\bar{N}_1(\eta), \tag{2.18}$$

$$\text{for vertex 3,} \qquad \Phi^1_{v_3}(\xi,\eta) = \bar{N}_2(\xi)\bar{N}_2(\eta), \tag{2.19}$$

$$\text{for vertex 4,} \qquad \Phi^1_{v_4}(\xi,\eta) = \bar{N}_1(\xi)\bar{N}_2(\eta). \tag{2.20}$$

We plot the vertex modes in Figure 2.6.

ii) Edge Modes

Let $\phi^r(\xi)$ be the r^{th} degree Lobatto polynomials defined earlier. For $r_k \geq 2$, there are $\sum_{k=1}^{4}(r_k-1)$ edge modes for each element. For a given edge k, the edge modes are a product of the linear functions $\bar{N}_i,\ i = 1$ or 2 and a hierarchical polynomial of degree r_k, such that the linear function is a function of the variable normal to edge k and the hierarchical polynomial is a function of the variable associated with the current edge k. In this way, the linear functions "blend" the hierarchical functions onto the element. We show the quadratic and cubic edge modes in Figures 2.7 and 2.8 respectively.

$$\text{on edge 1} \quad \Phi^r_{e_1}(\xi,\eta) = \bar{N}_1(\eta)\phi^r(\xi), \qquad r = 2,3,\ldots,r_1, \tag{2.21}$$

$$\text{on edge 2} \quad \Phi^r_{e_2}(\xi,\eta) = \bar{N}_2(\xi)\phi^r(\eta), \qquad r = 2,3,\ldots,r_2, \tag{2.22}$$

$$\text{on edge 3} \quad \Phi^r_{e_3}(\xi,\eta) = \bar{N}_2(\eta)\phi^r(\xi), \qquad r = 2,3,\ldots,r_3, \tag{2.23}$$

$$\text{on edge 4} \quad \Phi^r_{e_4}(\xi,\eta) = \bar{N}_1(\xi)\phi^r(\eta), \qquad r = 2,3,\ldots,r_4. \tag{2.24}$$

iii) Interior Modes

For the usual finite element hierarchical function bases, interior shape modes are used for polynomial approximations of degree four or higher. However, with the space $\mathcal{V}_p$, the interior shape modes are used for polynomials of degree three or higher. Thus for an element using a degree $s \geq 3$ approximation there will be $\frac{(s-1)(s-2)}{2}$ interior shape modes or interior degrees of freedom. The interior shape functions are designed to vanish at the element's boundaries. We begin by defining the bi-quadratic function $\bar{\phi}_\Delta$ which is the "bubble" function:

$$\bar{\phi}_\Delta(\xi,\eta) = 16\prod_{i=1}^{2}\bar{N}_i(\xi)\bar{N}_i(\eta) = (1-\xi^2)(1-\eta^2). \tag{2.25}$$

The bubble function is indeed the first interior shape function and is used in all the other interior shape modes as a blending function.

For an approximation degree $s \geq 3$ the interior shape modes will be defined as

$$\Phi^{s,\lambda,\mu}_\Delta(\xi,\eta) = \bar{\phi}_\Delta(\xi,\eta)P_\lambda(\xi)P_\mu(\eta), \qquad \lambda+\mu = s-3, \tag{2.26}$$

where s, λ, μ are respectively, the total approximation degree, the degree of $P_\lambda(\xi)$, and the degree of $P_\mu(\eta)$.

The first few interior shape modes are

$$\Phi_{\Delta}^{3,0,0}(\xi,\eta) = \bar{\phi}_{\Delta}(\xi,\eta), \tag{2.27}$$

$$\Phi_{\Delta}^{4,1,0}(\xi,\eta) = \bar{\phi}_{\Delta}(\xi,\eta)P_1(\xi), \tag{2.28}$$

$$\Phi_{\Delta}^{4,0,1}(\xi,\eta) = \bar{\phi}_{\Delta}(\xi,\eta)P_1(\eta), \tag{2.29}$$

$$\Phi_{\Delta}^{5,2,0}(\xi,\eta) = \bar{\phi}_{\Delta}(\xi,\eta)P_2(\xi), \tag{2.30}$$

$$\Phi_{\Delta}^{5,1,1}(\xi,\eta) = \bar{\phi}_{\Delta}(\xi,\eta)P_1(\xi)P_1(\eta), \tag{2.31}$$

$$\Phi_{\Delta}^{5,0,2}(\xi,\eta) = \bar{\phi}_{\Delta}(\xi,\eta)P_2(\eta), \tag{2.32}$$

and are displayed in Figure 2.9.

We present a summary view of all the shape modes in the space $\mathcal{V}_p$ for $p = 1$ to 6, in Figure 2.10.

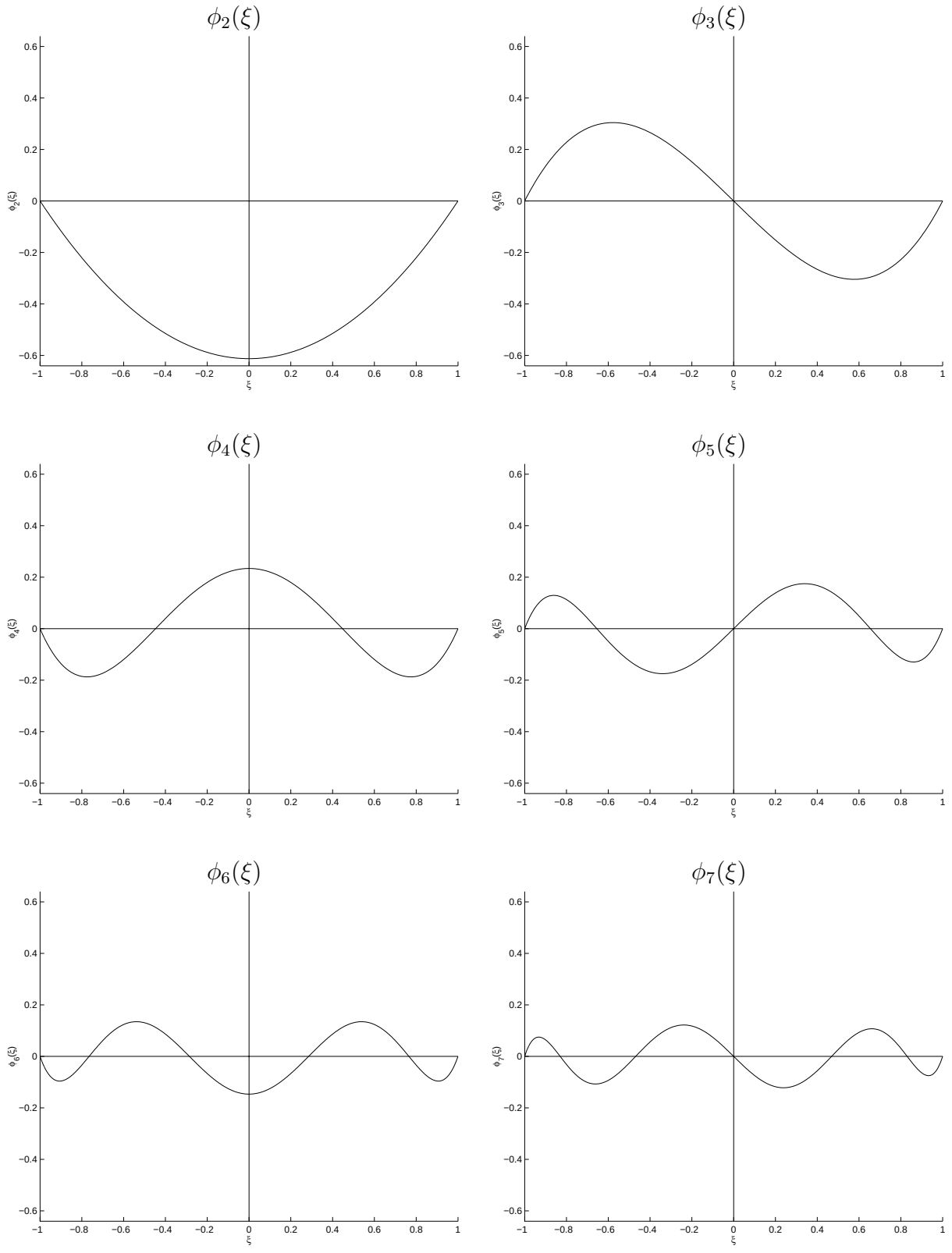


Figure 2.4: Lobatto polynomials for degrees 2 to 7 (upper left to lower right).

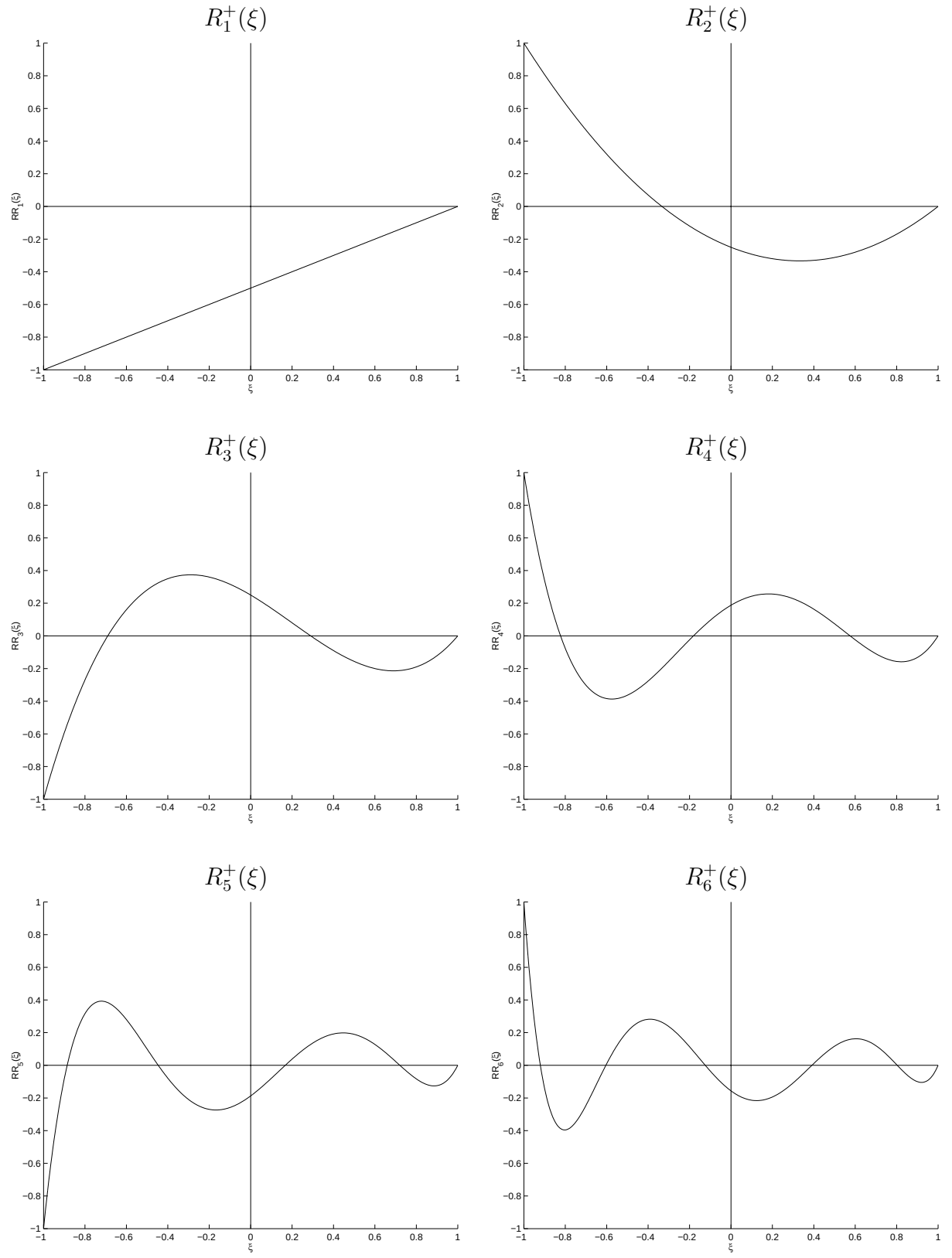


Figure 2.5: Right Radau polynomials from degree 1 to 6 (upper left to lower right).

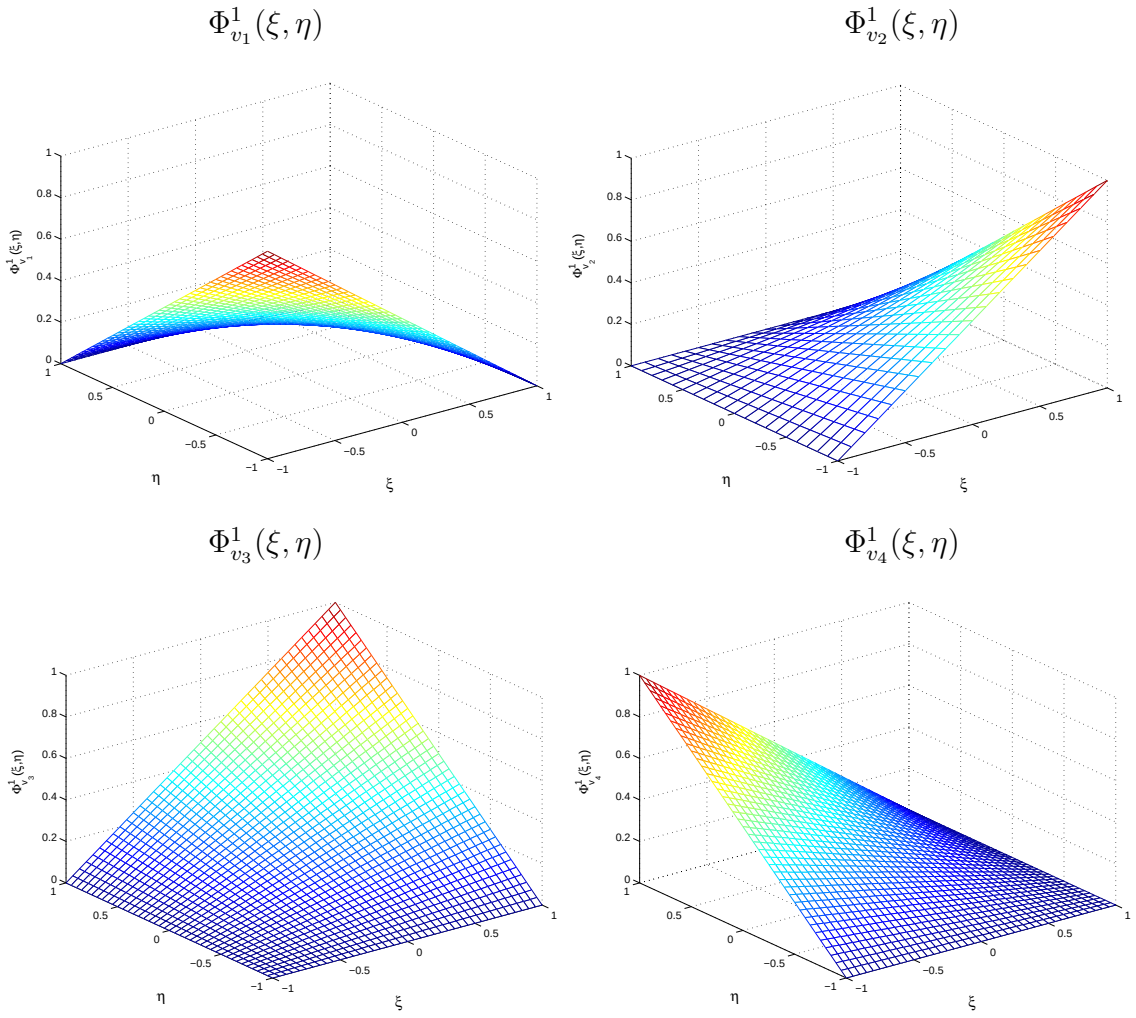


Figure 2.6: Vertex shape functions.

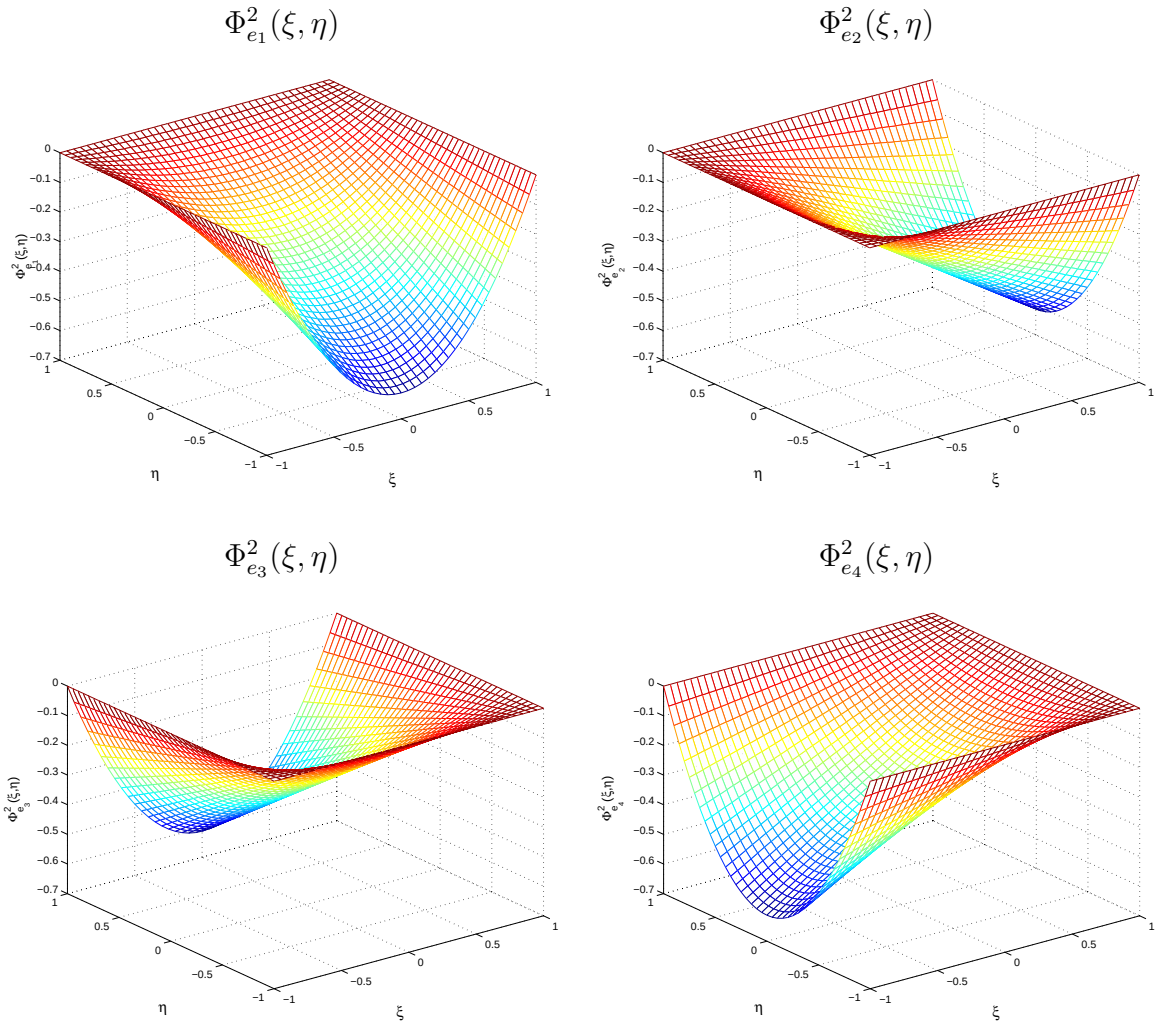


Figure 2.7: Quadratic edge shape functions for edges $1, 2, 3, 4$ (upper left to lower right).

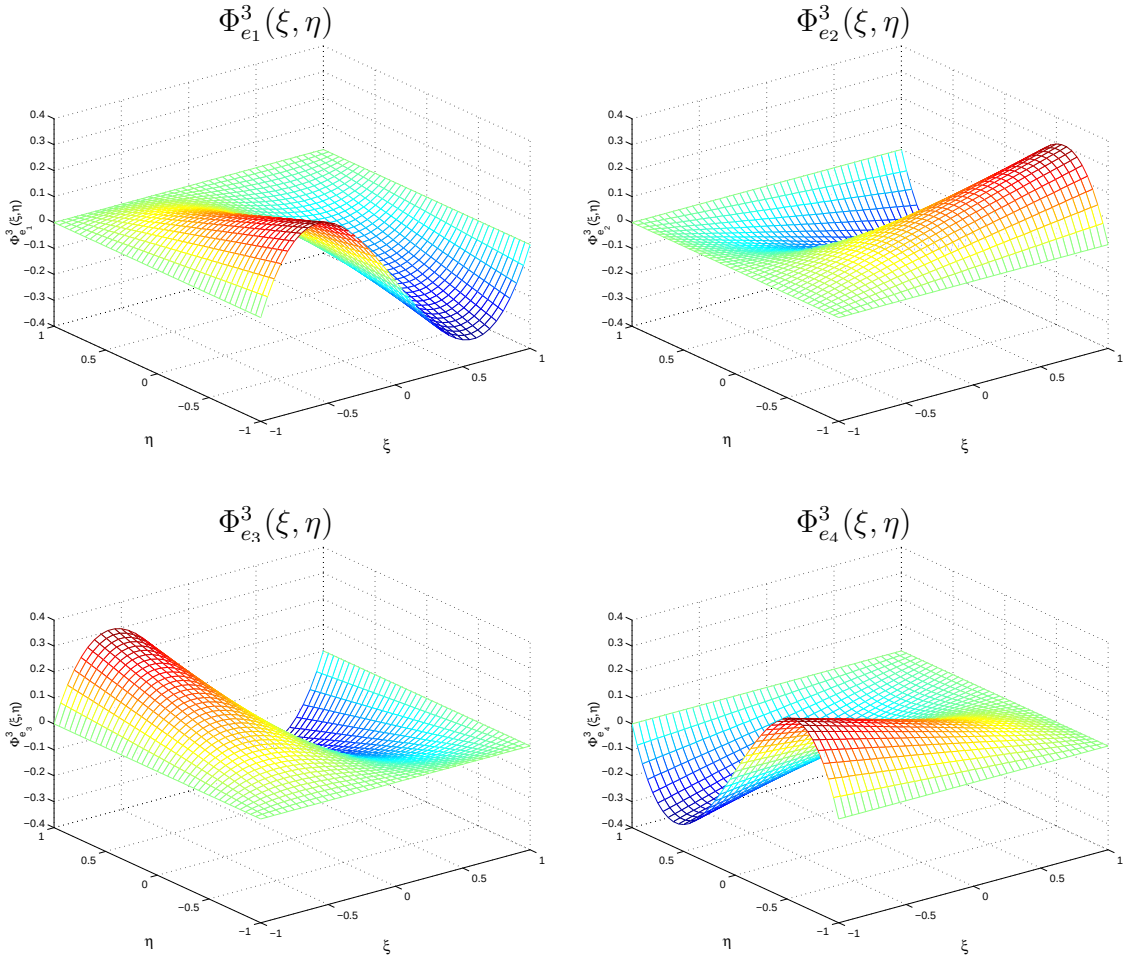


Figure 2.8: Cubic edge shape functions for edges $1, 2, 3, 4$ (upper left to lower right).

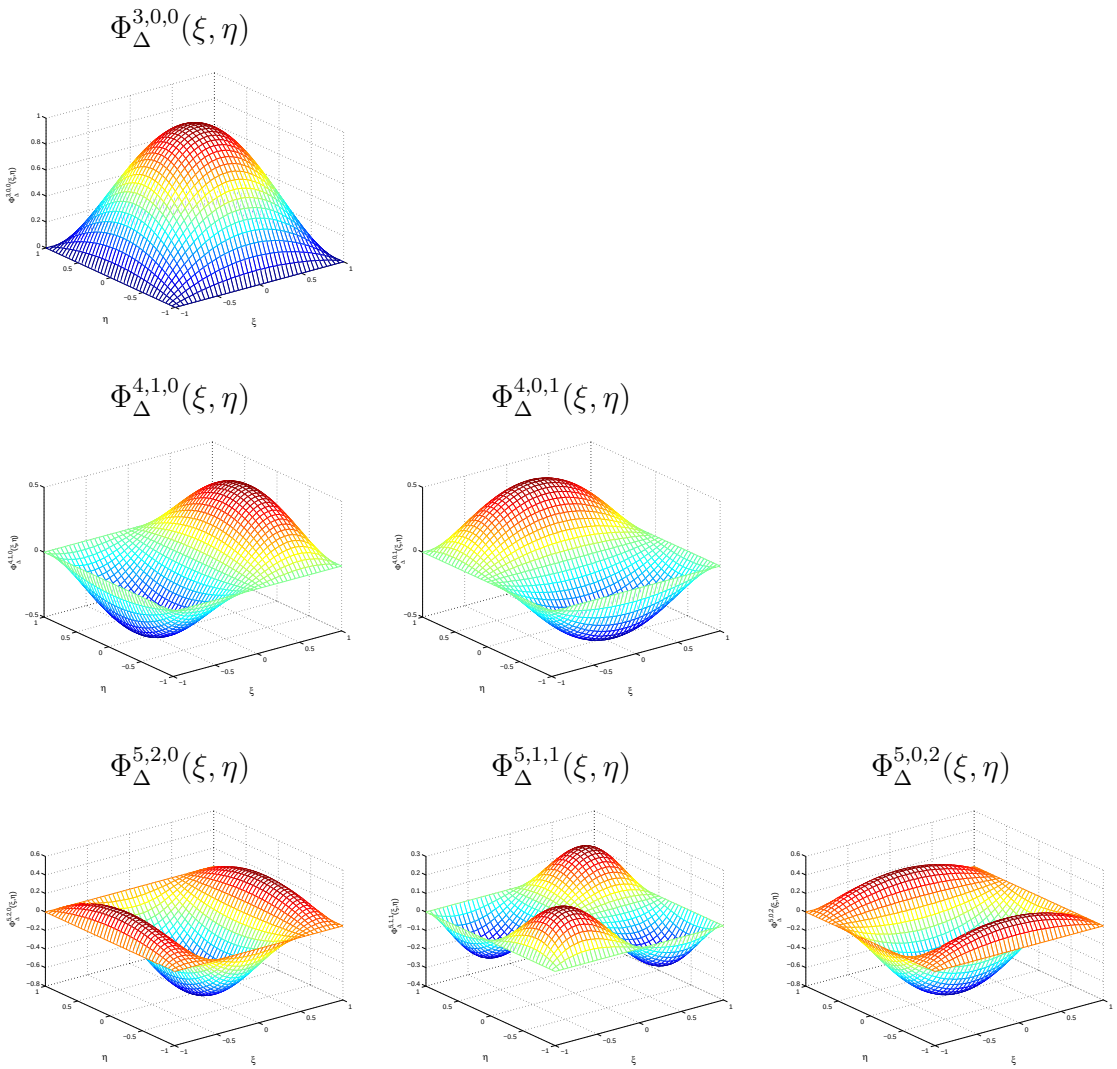


Figure 2.9: The first six interior shape functions.

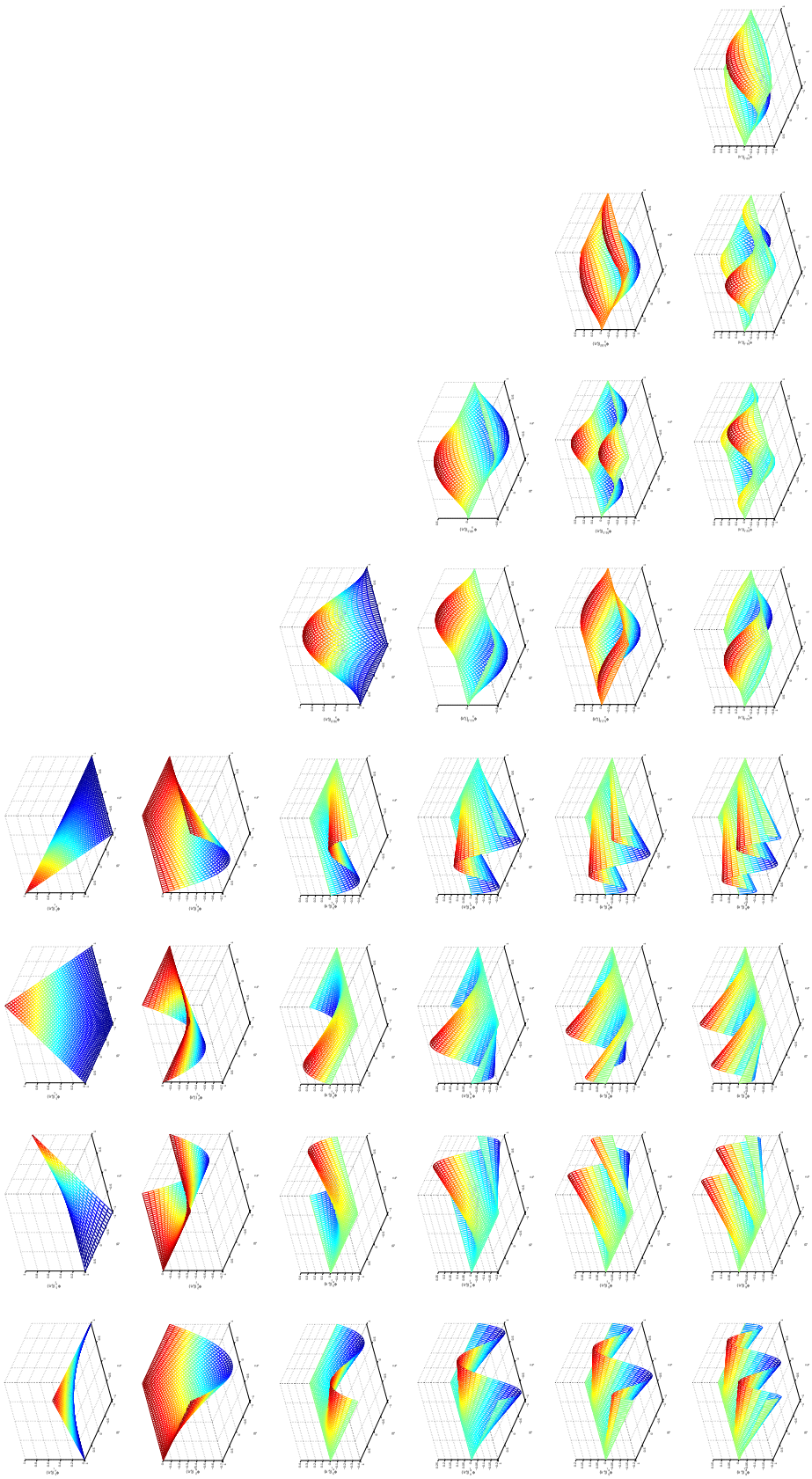


Figure 2.10: The first six degrees of the basis functions in the space $\mathcal{V}_p$.

$n=1$	$n=2$	$n=3$	$n=4$
1	-0.33333 33333 33333	-0.68989 79485 56636	-0.82282 40809 74592
	1	0.28989 79485 56636	-0.18106 62711 18531
		1	0.57531 89235 21694
			1
$n=5$	$n=6$	$n=7$	$n=8$
-0.88579 16077 70965	-0.92038 02858 97062	-0.94136 71456 80430	-0.95504 12271 22575
-0.44631 39727 23752	-0.60397 31642 52784	-0.70384 28006 63031	-0.77064 18936 78191
0.16718 08647 37834	-0.12405 03795 05228	-0.32603 06194 37691	-0.46842 03544 30821
0.72048 02713 12439	0.39092 85467 07272	0.11734 30375 43100	-0.09430 72526 611107
1	0.80292 98284 02347	0.53846 77240 60109	0.29475 05657 73661
	1	0.85389 13426 39482	0.63951 86165 26215
		1	0.88747 48789 26156
			1
$n=9$	$n=10$	$n=11$	$n=12$
-0.96444 01697 05273	-0.97117 51807 02247	-0.97616 47731 35169	-0.97996 34390 76639
-0.81735 27842 00412	-0.85122 52205 81608	-0.87653 58562 45704	-0.89592 90977 45639
-0.57138 30412 08739	-0.64776 66876 74009	-0.70577 71007 13860	-0.75076 15497 11114
-0.25613 56708 33455	-0.38066 48401 44724	-0.47768 06479 83088	-0.55431 87859 12324
0.09037 33696 068533	-0.07605 91978 379781	-0.21072 03062 28426	-0.31998 36841 70670
0.42635 04857 11139	0.23623 44693 90588	0.07347 75314 313213	-0.06372 47738 208319
0.71126 74859 15709	0.52564 60303 70079	0.35188 89233 53330	0.19699 45595 34278
0.91073 20894 20060	0.76384 20424 20003	0.60195 78420 73798	0.44440 65697 81936
1	0.92748 43742 33581	0.80342 19755 80294	0.66164 97992 45637
	1	0.93994 19356 77027	0.83391 67731 05190
		1	0.94945 27592 04959
			1

Table 2.3: Zeros of right Radau polynomial, $R_n^+(\xi)$ for $n = 1$ to 12.

Now that we have constructed the building blocks that make up our approximation $U(x,y)$ on each element, we are ready to derive the weak formulation for the FGM on both of our model problems. For simplicity, we begin with the linear model problem.

2.3 Weak Formulation for the Linear Model Problem

We reproduce the linear model problem defined in Chapter 1 for the reader's convenience. The problem is

$$\boldsymbol{\alpha} \cdot \nabla u + \gamma u = f(x,y), \qquad (x,y) \in \Omega = [0,1] \times [0,1], \tag{2.33a}$$

subject to the boundary conditions at the inflow boundaries

$$u(x,0) = g_1(x), \quad u(0,y) = g_2(y), \tag{2.33b}$$

where γ is a scalar constant and $\boldsymbol{\alpha}$ is a constant vector with $\alpha_i > 0,\ i = 1,2$. We again assume that $f(x,y)$, $g_1(x)$, and $g_2(y)$ are chosen such that (2.33) admits a unique solution $u(x,y) \in C^\infty(\Omega)$.

To determine our approximation $U(x,y)$, we start with element Δ_{ij} whose inflow boundary lies on the inflow boundary of the domain, $\Gamma_{\text{in}} \subset \partial\Omega_{\text{in}}$. We begin by multiplying equation (2.33a) by an arbitrary function $v(x,y)$ and then integrate over Δ_{ij}

$$\iint_{\Delta_{ij}} (\boldsymbol{\alpha} \cdot \nabla u + \gamma u - f)\ v\ dx\ dy = 0. \tag{2.34}$$

Next we apply Green's theorem and obtain

$$\oint_{\Gamma_{ij}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ u\ v\ ds - \iint_{\Delta_{ij}} (\boldsymbol{\alpha} \cdot \nabla v)\ u\ dx\ dy = \iint_{\Delta_{ij}} (f - \gamma u)\ v\ dx\ dy, \tag{2.35}$$

where $\boldsymbol{n}$ is the unit outward normal to Γ_{ij}. We decompose Γ_{ij} into inflow edges and outflow edges so that $\Gamma_{ij} = \Gamma_{\text{in}}^{ij} \cup \Gamma_{\text{out}}^{ij}$. With this, equation (2.35) becomes:

$$\oint_{\Gamma_{\text{in}}^{ij}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ u\ v\ ds + \oint_{\Gamma_{\text{out}}^{ij}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ u\ v\ ds$$

$$- \iint_{\Delta_{ij}} (\boldsymbol{\alpha} \cdot \nabla v)\ u\ dx\ dy = \iint_{\Delta_{ij}} (f - \gamma u)\ v\ dx\ dy. \tag{2.36}$$

Next we replace the true solution $u(x,y)$ with our approximate solution $U(x,y) \in \mathcal{V}_p$ and replace $v(x,y)$ with $V(x,y) \in \mathcal{V}_p^o$.

The space $\mathcal{V}_p^o$ is spanned by the mode shape functions that are associated with the unknown coefficients, which are any modes that are discontinuous on an inflow boundary, all the modes on the outflow boundaries, and all the interior modes. Thus each basis element V_i in $\mathcal{V}_p^o$ will be such that

$$V_i \in \{\Phi_{v_k}^1, \Phi_{e_k}^r, \Phi_{\Delta}^{s,\lambda,\mu}\} \tag{2.37}$$

where k is a local edge number (1 to 4) of the element Δ_{ij}. We note that $\mathcal{V}_p^o \subset \mathcal{V}_p$.

At this point we must take care in how we treat the inflow boundary terms. The true solution $u(x, y)$ when restricted to an edge of an element is approximated by two discrete solutions, one from each of the elements that share that edge. Since in the DGM case, these solutions need not be continuous, we must introduce a flux term to account for the two different approximations. For hyperbolic problems, we know that solution information follows along characteristic flow lines. Since we are determining the solution on the active element Δ_{ij} we must only concern ourselves with the flux on the inflow edges of Δ_{ij}, which we assume with out loss of generality to be edges 1 and 4. Using Green's theorem again eliminates the outflow terms. The discrete FGM formulation consists of finding $U \in \mathcal{V}_p$ such that

$$\oint_{\Gamma_{\text{in}}^{ij}} \boldsymbol{\alpha} \cdot \boldsymbol{n} \, [U - U^-] \, V \, ds - \iint_{\Delta_{ij}} (\boldsymbol{\alpha} \cdot \nabla U + \gamma U - f) \, V \, dx \, dy = 0, \qquad \forall \, V \in \mathcal{V}_p^o, \tag{2.38}$$

where U^- denotes the solution from the inflow boundary and is already known, i.e.,

$$U^-(x, y) = \lim_{s \to 0^+} U((x, y) + s\boldsymbol{n}). \tag{2.39}$$

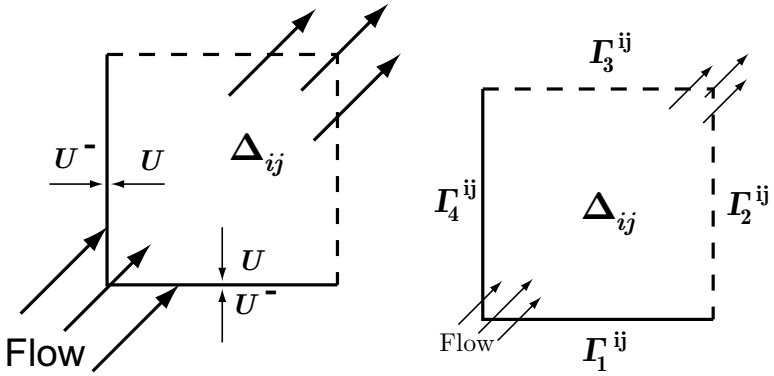


Figure 2.11: Element Δ_{ij} with inflow boundaries shown with solid lines and outflow boundaries with dashed lines.

For the boundary information U^- on Γ_{in}^{ij} we use either the exact boundary conditions or their interpolants

$$U^-(x, y) = \begin{cases} \pi g_1, & \text{if } (x, y) \in \Gamma_1^{ij} \\ \pi g_2, & \text{if } (x, y) \in \Gamma_4^{ij} \end{cases}, \tag{2.40}$$

where Γ_1^{ij} denotes edge one of element Δ_{ij} and likewise Γ_4^{ij} denotes edge four, both of which are on an inflow boundary, see Figure 2.11. The quantity πw is the one dimensional p-degree polynomial that interpolates w at the roots of the $(p+1)$-degree right Radau polynomial.

We now consider the composition of the discrete solution $U(x,y)$. We compose U in terms of the flexible Galerkin finite element shape functions described in §2.2. Therefore, we must consider vertex $\Phi^1_{v_k}$, edge $\Phi^{r_k}_{e_k}$, and interior $\Phi^{s,\lambda,\mu}_{\Delta_{ij}}$ shape functions when composing $U(x,y)$.

$$U(\mathbf{T}(\xi,\eta)) = \sum_{k=1}^{4} C_{v_k}\Phi^1_{v_k}(\xi,\eta) \qquad \text{(vertex modes)} \tag{2.41a}$$

$$+ \sum_{k=1}^{4}\sum_{r=2}^{r_k} C_{e_{k,r}}\Phi^r_{e_k}(\xi,\eta) \qquad \text{(edge modes)} \tag{2.41b}$$

$$+ \sum_{s=3}^{P_{\max}}\sum_{\lambda+\mu=s} C_{\Delta_{s,\lambda,\mu}}\Phi^{s,\lambda,\mu}_{\Delta}(\xi,\eta). \qquad \text{(interior modes)} \tag{2.41c}$$

We employ here the bilinear mapping $\mathbf{T}$, which we define in detail in (3.29), from (ξ,η) onto (x,y). We note that all the physical domains Ω in this work, are such that the bilinear mapping introduces no error with respect to the geometry.

We know that $U(x,y)$ is composed of three basic building blocks: vertex, edge, and interior modes, but in addition to these basic building blocks, we must also split $U(x,y)$ into two parts based on characteristic flow: (1) inflow modes and (2) outflow modes, which are respectively, those modes defined on characteristic inflow boundaries and characteristic outflow boundaries. We do this because information about the solution travels along characteristic lines. Namely, the coefficients of the vertex and edge components of the solution defined on inflow sides that are either continuous or on the boundary, have already been determined. For instance, if a vertex or edge mode on a characteristic inflow side is designated as being continuous between elements, then the coefficients corresponding to those modes are already known which helps reduce the size of the Galerkin system to be solved, this is a computational advantage over the usual discontinuous Galerkin method. Figure 2.12 shows the components of $U(x,y)$ divided into inflow and outflow parts and further divided into continuous and discontinuous parts, the ellipses represent known coefficient values. Therefore, using (2.38) we define a system of equations $G(C)=0$ to be solved,

$$G(C) := \oint_{\Gamma^{ij}_{\text{in}}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\,[U(C)-U^-]\;V\;ds - \iint_{\Delta_{ij}} (\boldsymbol{\alpha}\cdot\nabla U(C)+\gamma U(C)-f)\;V\;dx\;dy, \tag{2.42}$$

where the vector C contains the unknown coefficients of $U(x,y)$ and $V\in\mathcal{V}^o_p$.

An example will be useful here, so consider a cubic, $p=3$, approximation $U(x,y)$ to $u(x,y)$ which is continuous between boundaries and elements up to quadratic terms. Consider the

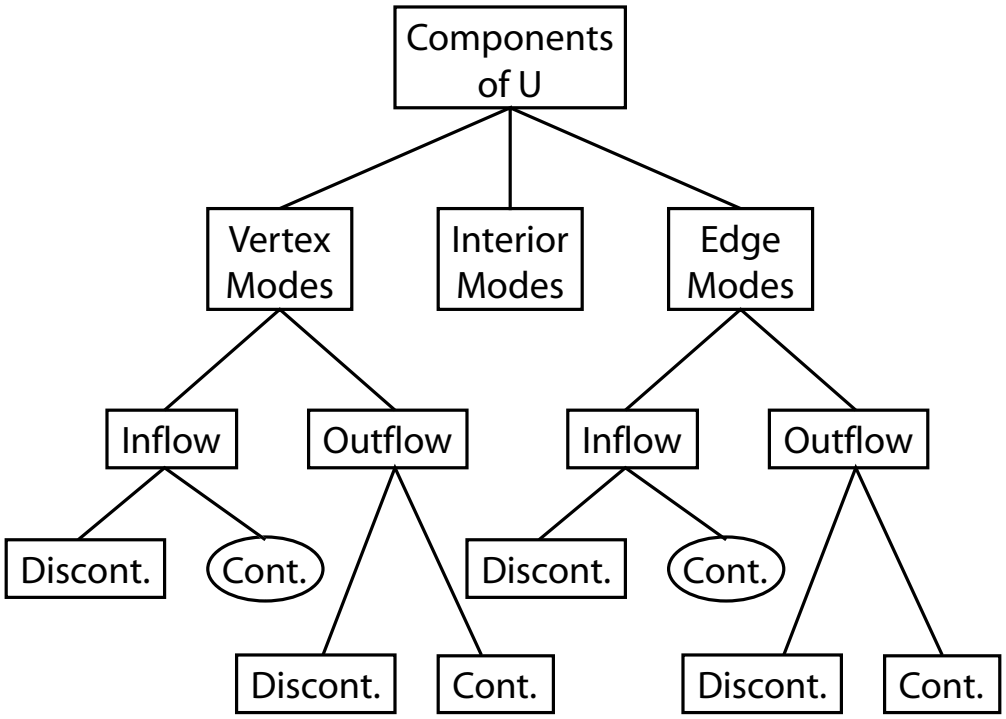


Figure 2.12: Components of $U(x, y)$ with an ellipse indicating known values and rectangles indicating unknown values.

canonical element and let edges 1 and 4 be inflow sides. Then our $U(x, y)$ is

$$\begin{aligned} U(\mathbf{T}(\xi,\eta)) = {} & C_{v_1}\Phi^1_{v_1} + C_{v_2}\Phi^1_{v_2} + C_{v_3}\Phi^1_{v_3} + C_{v_4}\Phi^1_{v_4} \\ & + C_{e_{1,2}}\Phi^2_{e_1} + C_{e_{1,3}}\Phi^3_{e_1} + C_{e_{2,2}}\Phi^2_{e_2} + C_{e_{2,3}}\Phi^3_{e_2} \\ & + C_{e_{3,2}}\Phi^2_{e_3} + C_{e_{3,3}}\Phi^3_{e_3} + C_{e_{4,2}}\Phi^2_{e_4} + C_{e_{4,3}}\Phi^3_{e_4} + C_{\Delta_{3,0,0}}\Phi^{3,0,0}_{\Delta}. \end{aligned} \quad (2.43)$$

Since we have continuity up to quadratic terms, the coefficients

$$\{C_{v_1}, C_{v_2}, C_{v_4}, C_{e_{1,2}}, C_{e_{4,2}}\} \quad (2.44)$$

are already known. Therefore, compared to the DGM we have reduced the number of unknowns from 13 to 8, using the FGM. We also note that there are only 2 more unknowns in the FGM than in the CGM for this example. For this example the space $\mathcal{V}_p^o$ of equation (2.42) is

$$\mathcal{V}_p^o \in \text{span}\{\Phi^1_{v_3}, \Phi^3_{e_1}, \Phi^2_{e_2}, \Phi^3_{e_2}, \Phi^2_{e_3}, \Phi^3_{e_3}, \Phi^3_{e_4}, \Phi^{3,0,0}_{\Delta}\}, \quad (2.45)$$

with each basis element V_i in $\mathcal{V}_p^o$ associated with one of the unknown coefficients

$$C = [C_{v_3}, C_{e_{1,3}}, C_{e_{2,2}}, C_{e_{2,3}}, C_{e_{3,2}}, C_{e_{3,3}}, C_{e_{4,3}}, C_{\Delta_{3,0,0}}]. \quad (2.46)$$

To solve the reduced system, $G(C) = 0$, for the C's in (2.46) we use Newton's method,

$$C_n = C_{n-1} - J(C_{n-1})^{-1} G(C_{n-1}), \tag{2.47}$$

where $J(C) = \mathbf{J}[G(C)]$ is the Jacobian of $G(C)$. Note that since our problem (2.33) is linear, the system of equations to be solved (2.42) is also linear and thus only one step of Newton's method is required to solve for C.

Once we have the solution on our first element, we move to an adjoining element in the direction of the characteristic flow lines, and repeat this process until all elements in the domain have be covered.

2.4 Weak Formulation for the Nonlinear Model Problem

Much of the work we did in the previous section on linear constant coefficient problems, carries over to nonlinear problems.
For the reader's convenience we reproduce the nonlinear model problem:

$$f_x(u) + g_y(u) = h(x, y), \qquad (x, y) \in \Omega = [0, 1] \times [0, 1], \tag{2.48a}$$

subject to the boundary conditions at the inflow boundaries

$$u(x, 0) = \phi_1(x) \qquad \text{and} \qquad u(0, y) = \phi_2(y). \tag{2.48b}$$

We also assume that $f'(u) > 0$ and $g'(u) > 0$, so that edges 1 and 4 of our elements will be the inflow edges.

The Galerkin weak formulation is obtained by multiplying (2.48a) by a test function v and integrating over Ω to obtain

$$\iint_{\Omega} \left(f_x(u) + g_y(u) - h(x, y)\right)\ v(x, y)\ dxdy = 0. \tag{2.49}$$

We partition Ω as described in (2.1) and select for each element Δ_{ij} a discrete approximation $U(x, y)$ for $u(x, y)$ and $V(x, y)$ for $v(x, y)$. Then use Green's theorem as we did before and account for the possibly discontinuous approximations on the edges of the element to get the following discrete weak form

$$\oint_{\Gamma_{\text{in}}^{ij}} \boldsymbol{n} \cdot \left[f(U) - f(U^-), g(U) - g(U^-)\right]\ V\ ds$$

$$- \iint_{\Delta_{ij}} \left(f_x(U) + g_y(U) - h(x, y)\right)\ V(x, y)\ dxdy = 0, \qquad \forall\ V \in \mathcal{V}_p^o, \tag{2.50}$$

where U^- is the solution from the inflow edges and is already known. Currently for the boundary information, U^- is either the exact boundary conditions or their interpolants defined

$$U^-(x,y) = \begin{cases} \pi\phi_1, & \text{if } (x,y) \in \Gamma_1^{ij} \\ \pi\phi_2, & \text{if } (x,y) \in \Gamma_4^{ij} \end{cases}, \tag{2.51}$$

where Γ_1^{ij} denotes edge one of element Δ_{ij} and likewise Γ_4^{ij} denotes edge four, both of which are on our assumed inflow boundary, see Figure 2.11. Again, the quantity πw is the one dimensional p-degree polynomial that interpolates w at the roots of the $(p+1)$-degree right Radau polynomial. However, in light of the proof for the superconvergence of the error in the flux on the outflow boundaries, see Theorem 4.3, an alternative would be to interpolate $f(u)$ and $g(u)$ at the right Radau points, instead of u.
Computationally, to determine inflow sides of element Δ_{ij} we let $\boldsymbol{n}$ be the unit outward normal to Γ^{ij}, then for each inflow edge where U^- exists, we check

$$\Gamma_{\text{in}}^{ij} = \{(x,y) \in \Delta_{ij} \mid \boldsymbol{n} \cdot [f'(U^-), g'(U^-)]|_{(x^*,y^*)} < 0\}, \qquad \text{inflow boundary} \tag{2.52a}$$

$$\Gamma_{\text{out}}^{ij} = \{(x,y) \in \Delta_{ij} \mid \boldsymbol{n} \cdot [f'(U^-), g'(U^-)]|_{(x^*,y^*)} \geq 0\}, \qquad \text{outflow boundary} \tag{2.52b}$$

where (x^*, y^*) is the midpoint of the edge in question. The discrete solution $U(x,y)$ is developed in the same manner as that of (2.41). Therefore, we need to solve a nonlinear system of equations

$$G(C) := \oint_{\Gamma_{\text{in}}^{ij}} \boldsymbol{n} \cdot \left[f(U(C)) - f(U^-), g(U(C)) - g(U^-)\right] \; V \; ds$$

$$- \iint_{\Omega} \left(f_x(U(C)) + g_y(U(C)) - h(x,y)\right) \; V(x,y) \; dxdy = 0, \tag{2.53}$$

for the unknown coefficients C. The vector C contains the unknown coefficients of $U(x,y)$ and $V \in \mathcal{V}_p^o$. In order to solve (2.53) we use Newton's method. For our initial guess to C, we solve the following linearized system of equations

$$\hat{G}(C) := \oint_{\Gamma_{\text{in}}^{ij}} \boldsymbol{n} \cdot [a,b] \; \left(U - U^-\right) \; V \; ds$$

$$- \iint_{\Omega} \left(aU_x(C) + bU_y(C) - h(x,y)\right) \; V(x,y) \; dxdy = 0, \qquad \forall \; V \in \mathcal{V}_p^o, \tag{2.54a}$$

where

$$a = f'(U^*) \quad \text{and} \quad b = g'(U^*), \tag{2.54b}$$

and

$$U^* = \frac{U_1^-(x_1^*, y_1^*) + U_4^-(x_2^*, y_2^*)}{2}, \tag{2.54c}$$

with $U_i^-(x_i^*, y_i^*)$ being U^- evaluated at the midpoint of edge i. Once we solve (2.54) for C we use it to start Newton's method and solve (2.53). Having found our approximate solution on element Δ_{ij} we move to the next element in the direction of the characteristic flow and repeat the process until we go over all the elements.

2.5 Examples and Cost Comparisons

We begin this section by stating the degrees of freedom associated with the FGM method when using the space $\mathcal{V}_p$. For a given element with a p-degree finite element approximation and a uniform level of continuity c enforced between elements and on boundaries, the degrees of freedom, N, for the local system of equations to be solved using the basis $\mathcal{V}_p$ is

$$N(p, c, d) = \left[4p + \frac{(p-1)(p-2)}{2}\right] - (1 - \delta_{0,c})(d\ c + 1), \quad p \geq 1, \quad 0 \leq c \leq p, \tag{2.55a}$$

where d is the number of inflow edges ($d = 1$ or $d = 2$) and

$$\delta_{0,c} = \begin{cases} 1 & \text{if } c = 0 \\ 0 & \text{if } c \neq 0 \end{cases}. \tag{2.55b}$$

From (2.55) we see that for $p \geq 2$ and $c \geq 2$ the number of degrees of freedom decrease by $-d\ c$ as we increase the level of continuity enforced. Thus the size of the system to be solved decreases by $-d\ c$, which is good news since solving an $n \times n$ full matrix system requires approximately $\frac{1}{3}n^3 + \frac{3}{2}n^2$ operations. This number of operations is based on using Gaussian elimination with scaled row pivoting, see Kincaid and Cheney [62] for more details. Furthermore since each step of our Newton's method requires that we solve a new system of equations this savings can add up quickly.

Table 2.4 lists the degrees of freedom for various p-degree finite element approximations with various levels of continuity, c, for an element with two inflow edges.

Here we consider our first example where we use the FGM. Consider a constant coefficient linear problem, as in the model linear problem (2.33) with $\gamma = 0$,

$$u_x + 2u_y = f(x, y), \qquad (x, y) \in \Omega = [0, 1] \times [0, 1], \tag{2.56}$$

with $f(x, y)$ and the initial conditions chosen so that the true solution is

$$u(x, y) = e^{x^2 - y^2}. \tag{2.57}$$

We solve this problem using $p = 1$ to 6 on uniform meshes having $25, 100, 225, 400, 625, 900$ elements and allowing the level of continuity to vary for each p from $c = 0$ to $c = p$.

p \ c	0	1	2	3	4	5	6
0	1	0	x	x	x	x	x
1	4	1	x	x	x	x	x
2	8	5	3	x	x	x	x
3	13	10	8	6	x	x	x
4	19	16	14	12	10	x	x
5	26	23	21	19	17	15	x
6	34	31	29	27	25	23	21

Table 2.4: The degrees of freedom for a single element with two inflow edges, versus the approximation degree p and the continuity level c.

Figures 2.14 and 2.15 show the rates of h-convergence for $||e||_{\mathcal{L}_2(\Omega)}$, denoted by m, with $p = 1$ to 6. Inside each picture is a table that gives the computed rate of convergence m for a particular level of continuity level c. For each p, the level of continuity c enforced is varied from $c = 0$, fully discontinuous, to $c = p$, fully continuous. In all cases, when $c = 0$ we get the optimal rate of convergence of $O(h^{p+1})$ for the DGM, and for $c = p$ we also obtain the optimal rate of $O(h^p)$ for the CGM. In each case, when $p \geq 2$ and $c = 1$, the rate of convergence is almost $O(h^{p+\frac{1}{2}})$. Whenever $p \geq 3$ and $2 \leq c < p$, the rate of convergence is almost $O(h^p)$ which is the same as in the fully continuous case.

Table 2.5 lists the CPU times in seconds required to solve (2.56) for a uniform 225-element mesh with approximation degrees from $p = 1$ to 4 and for each p the continuity level c varies from fully discontinuous to fully continuous. These are the total times required, including mesh generation, solving the linear system of equations, and storing the coefficients. These times were computed using *Matlab* version 6.0.0.88 release 12, on a desktop PC with dual 888 MHz 686 AT processors running Microsoft Windows 2000. What we see is that for each approximation degree p, the amount of change associated with varying the level of continuity enforced is nearly a linear quantity, but if p were large, we would see more of a savings, since the dominate operation then would be solving the system of equations. Also for $c = 0$ and $c = 1$, increasing the approximation degree from $p = 1$ to 2 and also from $p = 2$ to 3 nearly triples the CPU time while increasing from $p = 3$ to $p = 4$ almost doubles the CPU time.

Figure 2.13 shows the surface plot of the error, $e = u - U$, when solving (2.56) using a uniform 25-element mesh with $p = 3$ and allowing the continuity level to vary from $c = 0$ to 3. Notice that the error pattern is very regular in the fully discontinuous case, $c = 0$. These observations lead us to prove that the one-dimensional results of Adjerid *et al.* [5] for the DGM could be extended to higher dimensions. We present the necessary theory and results of these extensions in Chapter 4.

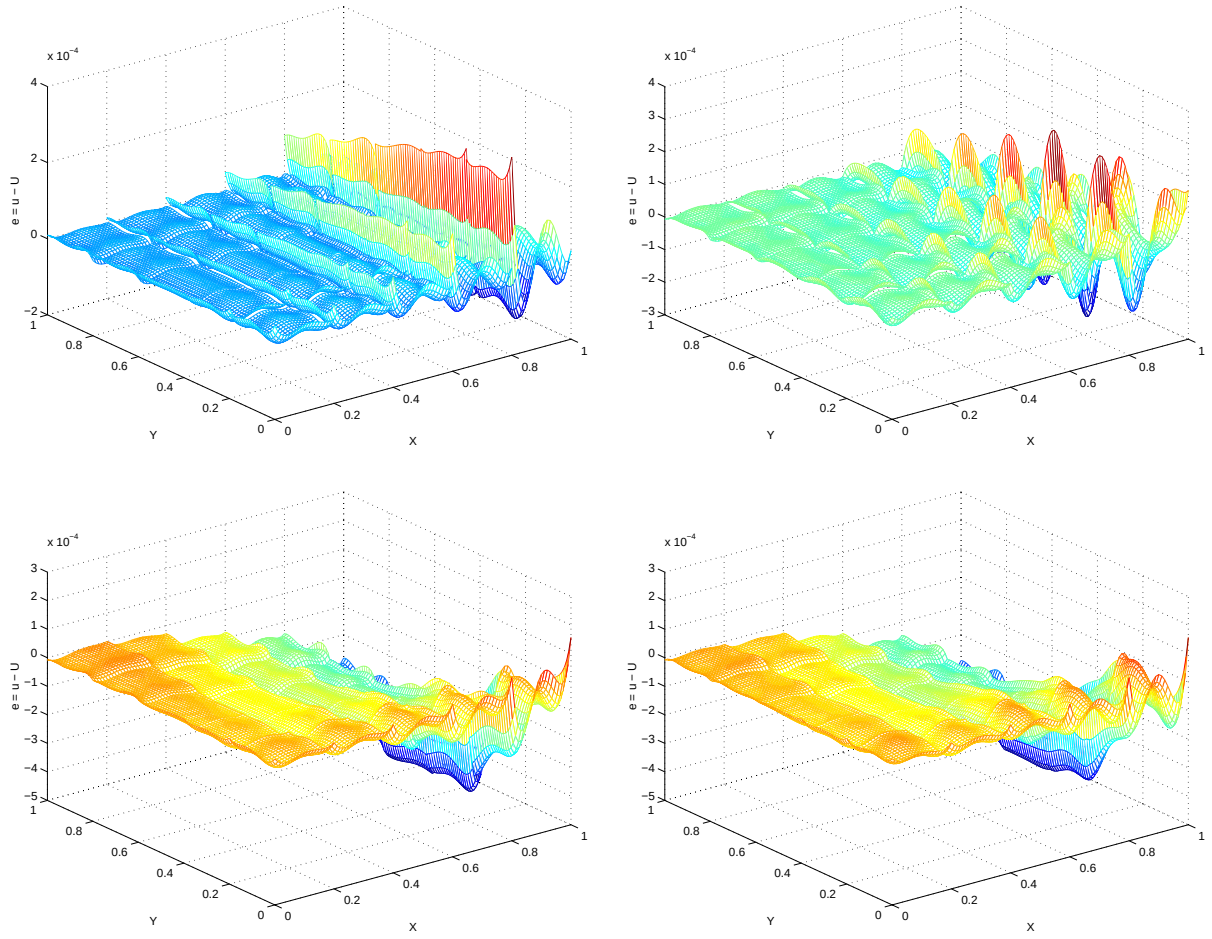


Figure 2.13: Surface plot of the error $e = u - U$, for (2.56) using a uniform 25-element mesh with $p = 3$ and continuity levels $c = 0$ to 3 (upper left to lower right).

p	c = 0	c = 1	c = 2	c = 3	c = 4
1	11.875	8.687			
2	32.875	28.641	23.735		
3	104.594	96.271	89.063	78.000	
4	206.812	194.62	185.265	173.656	158.343

Table 2.5: CPU times in seconds for a uniform 225-element mesh with approximation degrees $p = 1$ to 4 and continuity levels $c = 0$ to p.

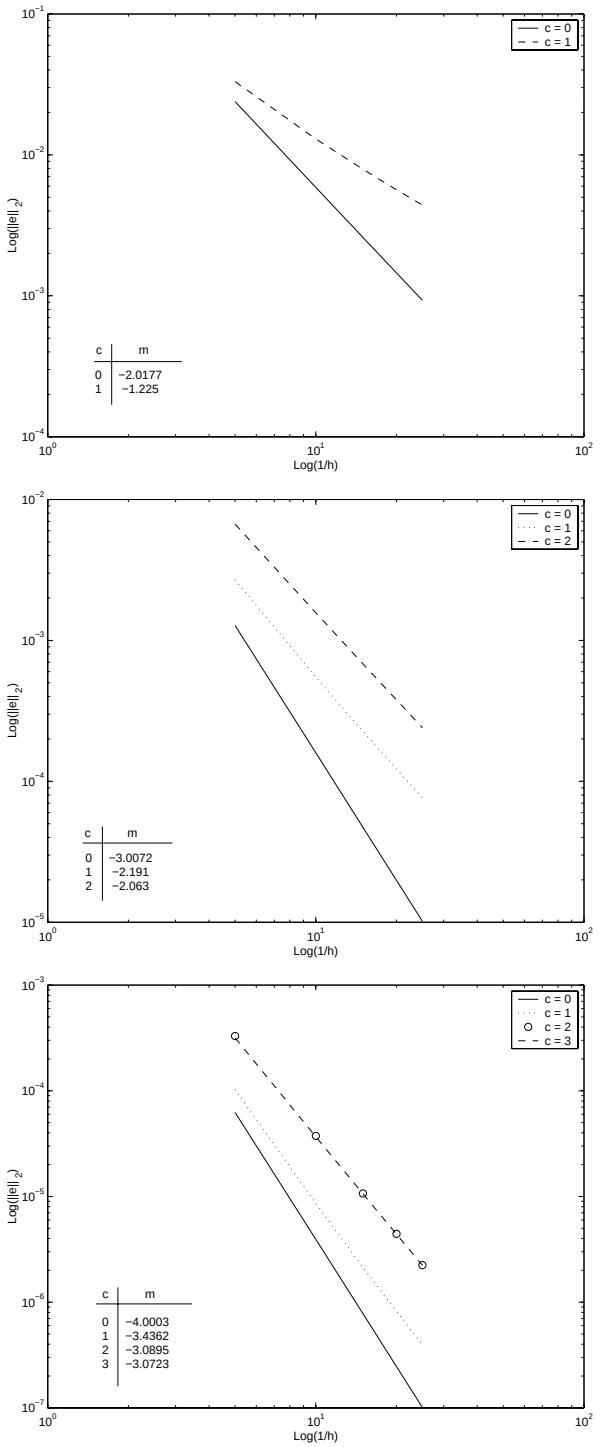


Figure 2.14: Rates of h-convergence for $||e||_{\mathcal{L}_2(\Omega)}$, denoted by m, for $p = 1$ to 3 with $c = 0$ to p.

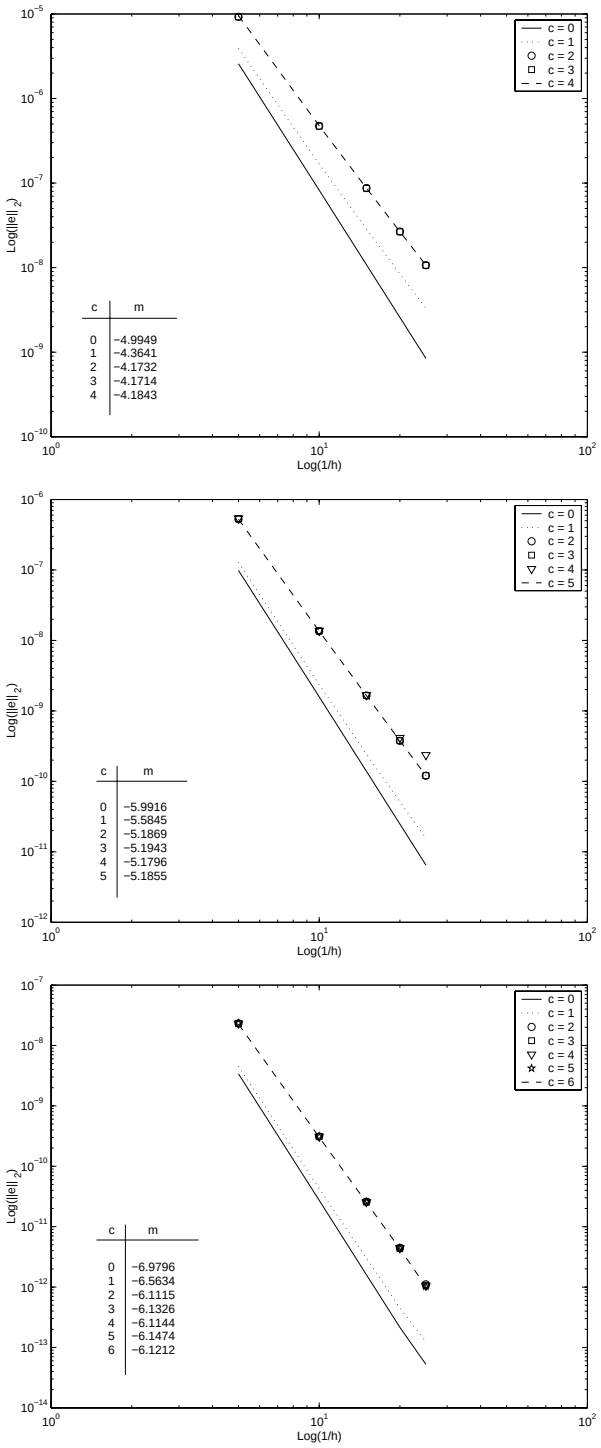


Figure 2.15: Rates of h-convergence for $||e||_{\mathcal{L}_2(\Omega)}$, denoted by m, for $p = 4$ to 6 with $c = 0$ to p.

Chapter 3

Computational Aspects for the FGM

3.1 Introduction

In this chapter we present a few of the computational aspects associated with the FGM as well as special issues associated with some of the examples included in Chapter 5.

We begin in §3.2 by describing the necessary components that make up an element and how this information is stored along with the storage and retrieval of the computed solution coefficients. In particular, in §3.2.1 we define the information needed to completely characterize the FGM element, in §3.2.2 we define how we store the solution coefficients, in §3.2.3 we describe how to retrieve the stored solution coefficients using pointers, and we conclude §3.2 with four examples in §3.2.4 that illustrate various aspects of all the data structures used for the FGM.

In §3.3 we present Gaussian numerical integration techniques that we use in our program for the FGM. In particular in §3.3.1 we show how to numerically approximate an oriented line integral. In §3.3.2 we present product type Gaussian quadratures for double integrals over general quadrilateral domains as well as non-product type symmetrical Gaussian quadratures for triangular domains. We also discuss the triangular coordinate system.

We conclude in §3.3.3 by discussing a routine we use to compute the $\mathcal{L}_2$ norm of a function that is discontinuous across an element.

3.2 Data Structures

3.2.1 Element Characterization

We begin by describing the data structure called **VERTEX**. **VERTEX** is a table collection of the vertex modes for the entire mesh. A single row in **VERTEX** contains the following information: 1) $x(v_i)$ the x-coordinate of the vertex , 2) $y(v_i)$ the y-coordinate of the vertex, 3) $\mathrm{SE}_{v_i}^j, j = 1:4$ the element numbers of the four elements sharing the vertex and 4) ϵ_i the continuity indicator. The numbers of the four elements sharing the vertex are listed in a counterclockwise manner with the vertex as the origin and beginning with the bottom/left most element, see Figure 3.1. When $\mathrm{SE}_{v_i}^j < 0$, this indicates the vertex is on a boundary, in which case, $\mathrm{SE}_{v_i}^j = -j$. The continuity indicator ϵ_i is 0 when the vertex is discontinuous and 1 when the vertex is continuous

$$\epsilon_i = \begin{cases} 0 & \text{if discontinuous} \\ 1 & \text{if continuous} \end{cases}.$$

For simplicity sake, a vertex is considered either continuous or discontinuous on all its shared elements. **VERTEX** is arranged in numerical order, from top to bottom, based on the numbering of the global vertex modes so that indexing a particular vertex is easily achieved, see Table 3.1.

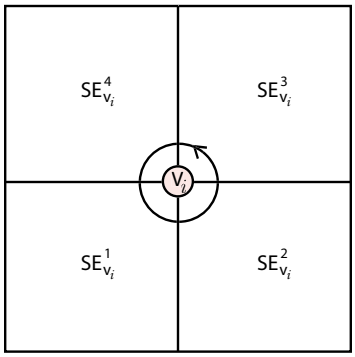


Figure 3.1: A vertex with its shared elements and labeling orientation.

X Coordinate	Y Coordinate	Elements Sharing a Vertex				Continuity Indicator
$x(v_1)$	$y(v_1)$	$\mathrm{SE}_{v_1}^1$	$\mathrm{SE}_{v_1}^2$	$\mathrm{SE}_{v_1}^3$	$\mathrm{SE}_{v_1}^4$	ϵ_1
$x(v_2)$	$y(v_2)$	$\mathrm{SE}_{v_2}^1$	$\mathrm{SE}_{v_2}^2$	$\mathrm{SE}_{v_2}^3$	$\mathrm{SE}_{v_2}^4$	ϵ_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x(v_N)$	$y(v_N)$	$\mathrm{SE}_{v_N}^1$	$\mathrm{SE}_{v_N}^2$	$\mathrm{SE}_{v_N}^3$	$\mathrm{SE}_{v_N}^4$	ϵ_N

Table 3.1: **VERTEX** data structure.

The next data structure is called **EDGE** and is a table collection of the edge modes for the entire mesh. A single row in **EDGE** corresponding to (e_i) contains the following information: 1) v_{LL} is the edge's bottom/left most vertices' global vertex number, 2) v_{UR} is the edge's top/right most vertices' global vertex number and 3) P_{c_i} the maximum level of continuity enforced on that edge. By default the value of P_{c_i} is 0, which indicates it is discontinuous. Since edge shape functions are quadratic and higher in degree, the other possible values of P_{c_i} are $2, 3, 4, \ldots$ which corresponds to the particular level of continuity being enforced. Thus, if local edge 3 (e_3) uses a cubic approximation and we require the solution to be continuous up to linear terms on that edge, then $P_{c_3} = 0$. If we required the solution to be continuous up to quadratic terms, then $P_{c_3} = 2$. A fully continuous cubic solution on edge 3 would be indicated by having $P_{c_3} = 3$. **EDGE** is arranged in numerical order, from top to bottom, based on the numbering of the global edge modes, see Table 3.2.

Bottom/Left Vertex Number	Top/Right Vertex Number	Maximum Continuity Enforced
$v_{\text{LL}}(e_1)$	$v_{\text{UR}}(e_1)$	P_{c_1}
$v_{\text{LL}}(e_2)$	$v_{\text{UR}}(e_2)$	P_{c_2}
$\vdots$	$\vdots$	$\vdots$
$v_{\text{LL}}(e_N)$	$v_{\text{UR}}(e_N)$	P_{c_N}

Table 3.2: **EDGE** data structure.

The next data structure is called **ELEMENT** and is a table collection that describes the components of each individual element. A single row of **ELEMENT**, from left to right, contains the following: 1) the global vertex numbers, 2) the global edge numbers, 3) the neighbor numbers and 4) the maximum approximation degree for each edge of the element. Each of the components, (global vertex numbers, global edge numbers, etc...) are listed in a counterclockwise manner beginning with the bottom/left most individual component, see Figure 3.2. **ELEMENT** is arranged in numerical order, from top to bottom, according to the global element numbers, see Table 3.3.

Global Vertex Numbers				Global Edge Numbers				Neighbor Element Numbers				Maximum Edge Approximation Degrees			
v_1^1	v_2^1	v_3^1	v_4^1	e_1^1	e_2^1	e_3^1	e_4^1	δ_1^1	δ_2^1	δ_3^1	δ_4^1	P_{e1}^1	P_{e2}^1	P_{e3}^1	P_{e4}^1
v_1^2	v_2^2	v_3^2	v_4^2	e_1^2	e_2^2	e_3^2	e_4^2	δ_1^2	δ_2^2	δ_3^2	δ_4^2	P_{e1}^2	P_{e2}^2	P_{e3}^2	P_{e4}^2
$\vdots$				$\vdots$				$\vdots$				$\vdots$			
v_1^N	v_2^N	v_3^N	v_4^N	e_1^N	e_2^N	e_3^N	e_4^N	δ_1^N	δ_2^N	δ_3^N	δ_4^N	P_{e1}^N	P_{e2}^N	P_{e3}^N	P_{e4}^N

Table 3.3: **ELEMENT** data structure.

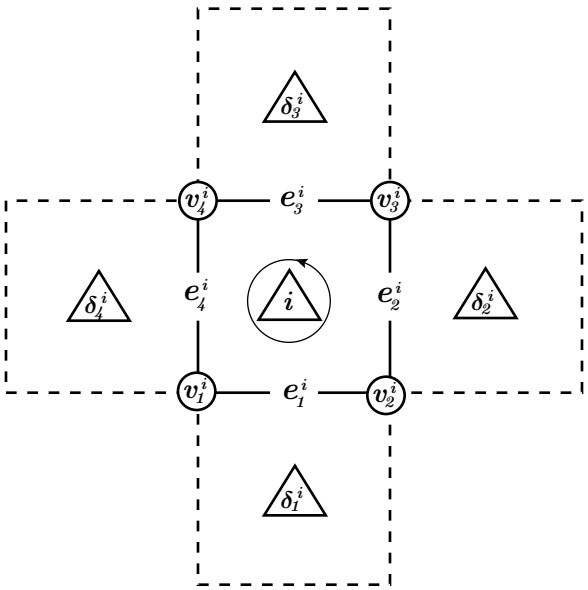


Figure 3.2: Element Δ_i with vertices, edges, and neighbors.

3.2.2 Storage of Solution Coefficients

In this section we discuss how the computed solution coefficients are stored.

All the coefficients are stored in vectors to minimize storage space. There are three data storage structures used: 1) **U-vertex**, 2) **U-edge** and 3) **U-element**. Retrieval and placement of the coefficients requires the use of pointers. A description of the pointers used to access the coefficients for each of the storage data structures is contained in §3.2.3.

U-vertex and **U-edge** are only used for continuous solution coefficients. **U-vertex** is a vector of length equal to the number of vertices in the mesh. A single entry in **U-vertex** contains the coefficient corresponding to the vertex shape function for that vertex if it is designated as having a continuous solution. If a vertex is discontinuous then **U-vertex** has a zero in its place. This wastes some storage space, but is acceptable nonetheless. Table 3.4 illustrates the general design of **U-vertex**.

Vertex 1	Vertex 2	Vertex 3	$\cdots$	Vertex N
C_{v_1}	C_{v_2}	C_{v_3}	$\cdots$	C_{v_N}

Table 3.4: **U-Vertex** data structure.

Since a single edge may have several edge shape functions, each edge may require more than a single entry in a data storage structure. For this reason, a pointer will be used in connection with **U-edge**, see §3.2.3. Only coefficients corresponding to continuous edge components are stored here. Discontinuous component coefficients are stored in **U-element**. **U-edge** can be thought of as being partitioned into blocks with each block containing the coefficients corresponding to a particular edge. This block begins with the coefficient for the

quadratic edge shape function, followed by the coefficient for the cubic edge shape function, and continues until the maximum level of continuity enforced for that edge is obtained. Table 3.5 gives the general framework for **U-edge**. The top row is a header placed here for emphasis only.

Edge 1	Edge 2	$\cdots$	Edge N
$C_{e_{1,2}}, C_{e_{1,3}}, \cdots, C_{e_{1,P_{c_{e_1}}}}$	$C_{e_{2,2}}, \cdots, C_{e_{2,P_{c_{e_2}}}}$	$\cdots$	$C_{e_{N,2}}, \cdots, C_{e_{N,P_{c_{e_N}}}}$

Table 3.5: **U-Edge** data structure.

The table, **U-element**, contains all the coefficients corresponding to discontinuous components, the interior shape function coefficients and piecewise constant solutions. Additionally, we have stored the coefficients for all the vertices in **U-element** regardless of their continuity status. This was done for easy access of that data. **U-element** is partitioned into blocks, much like **U-edge**, with each block containing information about a particular element. A sample block is shown in Table 3.6.

$\cdots$	PW	$V1$	$V2$	$V3$	$V4$	$e1$	$e2$	$e3$	$e4$	Δ_{int}	$\cdots$
$\cdots$	C_{pw}	C_{v1}	C_{v2}	C_{v3}	C_{v4}	$C_{e_{1,P_{c_{e_1}}+1}}, \cdots, C_{e_{1,P_{e_1 max}}}$	$\cdots$	$\cdots$	$\cdots$	$C_{\Delta_1}, \cdots, C_{\Delta_N}$	$\cdots$

Table 3.6: A block from **U-element** data structure for a single element with PW representing a constant solution, (Note: all indices are local).

3.2.3 Solution Retrieval Pointers

U-edge-pt is the pointer corresponding to **U-edge**. It is a vector containing the beginning index numbers where the coefficients corresponding to continuous edge shape functions are stored in **U-edge**. **U-edge-pt** is built one edge at a time in numerical order using a running counter, *pb*. The counter *pb* is initialized to one. As each edge is examined the value of *pb* is entered into the pointer location for that edge and then *pb* is updated if necessary. The value of *pb* is updated based on the following criteria: For a given edge if the approximation degree is high enough to warrant edge mode approximations then the number of continuous modes used is counted and added to *pb*, otherwise *pb* is not updated. A procedure for constructing **U-edge-pt** is given in Figure 3.3. Consecutive values in **U-edge-pt** can be used to indicate the number of coefficients stored in **U-edge** for a particular edge. For example, given edge number k, the number of continuous coefficients stored in **U-edge** for edge k is **U-edge-pt**$(k+1)$ − **U-edge-pt**(k).

U-elem-pt is the pointer that corresponds to **U-element**, where the coefficients for all the discontinuous modes, as well as any constant solution values are stored. **U-elem-pt** is a table, with information about element i located in the i^{th} column of the table, see Table 3.7. **U-elem-pt** is assembled one column at a time which corresponds to one element of the

Variables
k (is the current edge number)
numedges (is the total number of edges)
P_{c_k} (is the maximum level of continuity enforced on edge k)
pb (is the current beginning index number in U-edge)

$pb := 1$
$k := 1$
`while` $k \leq numedges$
    U-edge-pt(k) := pb
    `if` $P_{c_k} > 0$ `then`
        $pb := pb + P_{c_k} - 1$
    `end if`
    $k := k + 1$
`end loop`
U-edge-pt(k) := pb

Figure 3.3: A procedure for defining **U-edge-pt**.

mesh at a time. There are six rows in **U-elem-pt**, see Table 3.7, with the first row, labeled V^i-Begin, corresponding to the beginning position of the vertex coefficients or the constant solution, the second through fifth rows, e_k^i-Begin, $k = 1$ to 4, correspond to the beginning positions of the coefficients for the four edges, and the sixth row, Δ_{int}^i-Begin, corresponds to the beginning of the interior mode coefficients. A counter, pb, is employed to keep a running total of the beginning positions of each of the element's mode's coefficients, namely, vertex, edge, interior and constant solution. The counter is initialized to one. Then for a given element i, the first row of **U-elem-pt** is set to pb and then pb is updated by adding $\delta pb(V^i)$, which is either equal to 1 if the solution is a constant or is equal to 4 which corresponds to the four vertex coefficients, so $pb = pb + \delta pb(V^i)$. The next four rows are determined in the following manner: Given edge k, the value of pb is placed in the corresponding row, then pb is updated by adding the number of discontinuous edge coefficients for edge k, which is given by $pb = pb + \delta pb(e_k^i)$. The last row which corresponds to the interior modes is then set to the value of pb. In the last step pb is updated by adding the number of interior degrees of freedom used on element i, which is given by $pb = \delta pb(\Delta_{int}^i)$. The interior degrees of freedom, idf, for element i, is computed by first finding the maximum of the approximation degrees used over all the edges, call that value $P_{\max}$ and then computing

$$idf = \begin{cases} 0 & \text{if } P_{\max} < 3 \\ \frac{(P_{\max}-1)(P_{\max}-2)}{2} & \text{if } P_{\max} \geq 3 \end{cases}. \tag{3.1}$$

We give a procedure for constructing **U-elem-pt** in Figure 3.4.

Variables
i (is the current element number)
k (is the current local edge number)
$P^i_{c_k}$ (is the maximum level of continuity enforced on edge k)
$P^i_{e_k}$ (is the maximum approximation degree on edge k for element Δ_i)
$P_{\max}$ (is the maximum approximation degree used on element i)
pb (is the current beginning index number in U-elem-pt)
idf (is the interior degrees of freedom for element Δ_i)
$nelm$ (is the total number of elements)

```
i := 1,  pb := 1
while i ≤ nelm
    U-elem-pt(1, i) := pb
    if P^i_{e_1} ≠ 0 then
        pb := pb + 4
        k := 1,  P_max := 0,  idf := 0
        while k ≤ 4
            U-elem-pt(k + 1, i) := pb
            if P^i_{e_k} > 1 then
                temp := P^i_{e_k} − P^i_{c_k}
                if P^i_{c_k} > 0 then
                    δpb := temp
                else
                    δpb := temp − 1
                end if
            else
                δpb := 0
            end if
            pb := pb + δpb
            k := k + 1
            if P^i_{e_k} > P_max then
                P_max := P^i_{e_k}
            end if
        end while
        U-elem-pt(6, i) := pb
        if P_max ≥ 3 then
            idf := (P_max − 1)(P_max − 2)/2
        end if
        pb := pb + idf
        i := i + 1
    else
        pb := pb + 1
        U-elem-pt(2 : 6) := pb
        i := i + 1
    end if
end while
U-elem-pt(1, nelm + 1) := pb
```

Figure 3.4: A procedure for constructing **U-elem-pt**.

Element Δ_i	Δ_1	Δ_2	$\cdots$	Δ_N	End
V^i - Begin	$pb = 1$	$pb + \delta pb(\Delta_{int}^1)$	$\cdots$	$pb + \delta pb(\Delta_{int}^{N-1})$	$pb + \delta pb(\Delta_{int}^N)$
e_1^i - Begin	$pb + \delta pb(V^1)$	$pb + \delta pb(V^2)$	$\cdots$	$pb + \delta pb(V^N)$	0
e_2^i - Begin	$pb + \delta pb(e_1^1)$	$pb + \delta pb(e_1^2)$	$\cdots$	$pb + \delta pb(e_1^N)$	0
e_3^i - Begin	$pb + \delta pb(e_2^1)$	$pb + \delta pb(e_2^2)$	$\cdots$	$pb + \delta pb(e_2^N)$	0
e_4^i - Begin	$pb + \delta pb(e_3^1)$	$pb + \delta pb(e_3^2)$	$\cdots$	$pb + \delta pb(e_3^N)$	0
Δ_{int}^i - Begin	$pb + \delta pb(e_4^1)$	$pb + \delta pb(e_4^2)$	$\cdots$	$pb + \delta pb(e_4^N)$	0

Table 3.7: **U-elem-pt** data structure (Note: pb is a counter and $\delta pb(\cdot)$ is an increment).

3.2.4 Examples of Meshes and Their Data Structures

We give four examples to illustrate a few possible cases of meshes and continuity levels and their respective data structures.

Example 1. We consider a rectangular domain on $[0,1] \times [0,1]$ using a 6-element mesh with 2 elements in the x-direction and 3 elements in the y-direction and a degree $p = 3$ finite element approximation that is fully discontinuous. For $p = 3$ degree approximations there is only one interior degree of freedom, so $idf = 1$ for each element. Figure 3.5 illustrates the global numbering of the elements and their parts. Table 3.8 shows **VERTEX**, notice that the last column containing ϵ_i is all zeros since we are not enforcing any continuity. Table 3.9 shows **EDGE** for this mesh and again notice that the last column is all zeros because we are not enforcing any continuity. Table 3.10 illustrates **ELEMENT** and we note that the last four columns are all three's because we are using a uniform $p = 3$ approximation. Since we have a fully discontinuous scheme the solution coefficients are all stored in **U-element**. We show the corresponding pointer **U-elem-pt** in Table 3.11.

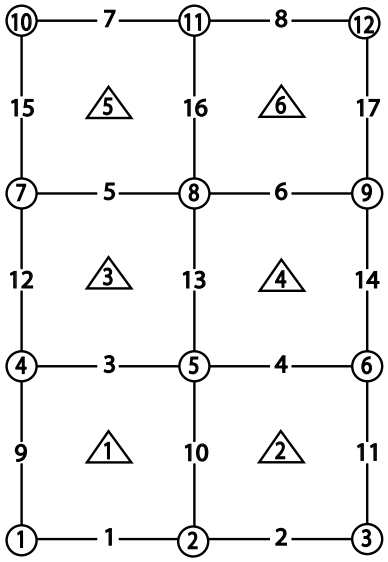


Figure 3.5: Mesh for *Example 1* where △ denotes elements, ◯ vertices, and − edges.

X Coordinate $x(v_i)$	Y Coordinate $y(v_i)$	Elements Sharing a Vertex				Continuity Indicator ϵ_i
		$SE^1_{v_i}$	$SE^2_{v_i}$	$SE^3_{v_i}$	$SE^4_{v_i}$	
0.00	0.00	-1	-2	1	-4	0
0.50	0.00	-1	-2	2	1	0
1.00	0.00	-1	-2	-3	2	0
0.00	0.33333	-1	1	3	-4	0
0.50	0.33333	1	2	4	3	0
1.00	0.33333	2	-2	-3	4	0
0.00	0.66667	-1	3	5	-4	0
0.50	0.66667	3	4	6	5	0
1.00	0.66667	4	-2	-3	6	0
0.00	1.00	-1	5	-3	-4	0
0.50	1.00	5	6	-3	-4	0
1.00	1.00	6	-2	-3	-4	0

Table 3.8: **VERTEX** table for the mesh shown in Figure 3.5.

Bottom/Left Vertex Number (v_{LL})	Top/Right Vertex Number (v_{UR})	Maximum Continuity Enforced P_{c_i}
1	2	0
2	3	0
4	5	0
5	6	0
7	8	0
8	9	0
10	11	0
11	12	0
1	4	0
2	5	0
3	6	0
4	7	0
5	8	0
6	9	0
7	10	0
8	11	0
9	12	0

Table 3.9: **EDGE** table for the mesh shown in Figure 3.5.

Global Vertex Numbers				Global Edge Numbers				Neighbor Element Numbers				Maximum Edge Approximation Degrees			
v_1^i	v_2^i	v_3^i	v_4^i	e_1^i	e_2^i	e_3^i	e_4^i	δ_1^i	δ_2^i	δ_3^i	δ_4^i	P_{e1}^i	P_{e2}^i	P_{e3}^i	P_{e4}^i
1	2	5	4	1	10	3	9	-1	2	3	-4	3	3	3	3
2	3	6	5	2	11	4	10	-1	-2	4	1	3	3	3	3
4	5	8	7	3	13	5	12	1	4	5	-4	3	3	3	3
5	6	9	8	4	14	6	13	2	-2	6	3	3	3	3	3
7	8	11	10	5	16	7	15	3	6	-3	-4	3	3	3	3
8	9	12	11	6	17	8	16	4	-2	-3	5	3	3	3	3

Table 3.10: **ELEMENT** table for the mesh shown in Figure 3.5.

Element Δ_i	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	End
V_1^i - Begin	1	14	27	40	53	66	79
e_1^i - Begin	5	18	31	44	57	70	0
e_2^i - Begin	7	20	33	46	59	72	0
e_3^i - Begin	9	22	35	48	61	74	0
e_4^i - Begin	11	24	37	50	63	76	0
Δ_{int}^i - Begin	13	26	39	52	65	78	0

Table 3.11: **U-elem-pt** table for the mesh shown in Figure 3.5.

Example 2. We consider a rectangular domain on $[0,1] \times [0,1]$ and apply the 6-element mesh shown in Figure 3.6 where we also indicate global numbering of the element's components. We use $p = 5$ degree approximations that are continuous up to cubic terms on each element. For $p = 5$ degree approximations, there are six interior degrees of freedom, i.e., $idf = 6$ on each element. In Table 3.12 we show **VERTEX** and note that the last column is all ones since all the vertex modes are continuous. In Table 3.13 we show **EDGE** and note that all the entries in the last column are 3 which indicates that on every edge we will require the quadratic and cubic edge modes to be continuous across element boundaries. In Table 3.14 we show **ELEMENT** and note that all the entries in the last four columns are all 5 which indicates we are using a uniform $p = 5$ degree approximation on each element. Since we have edge modes that are continuous, we will use the **U-edge-pt** shown in Table 3.15 to store the corresponding coefficients in **U-edge**, which is not included. Finally in Table 3.16 we show **U-elem-pt** where we emphasis that for each edge we are only storing the coefficients for the fourth and fifth degree edge modes in **U-element**.

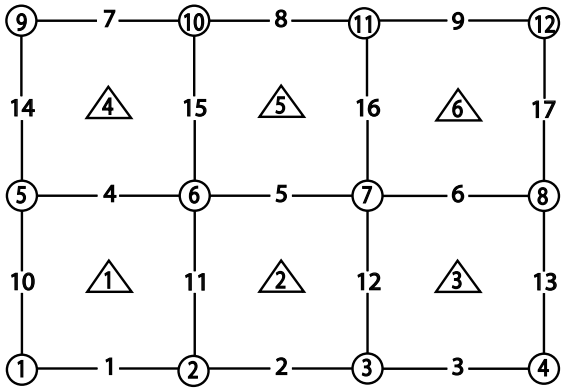


Figure 3.6: Mesh for *Example 2* where △ denotes elements, ◯ vertices, and − edges.

X Coordinate $x(v_i)$	Y Coordinate $y(v_i)$	Elements Sharing a Vertex				Continuity Indicator ϵ_i
		$\mathrm{SE}^1_{v_i}$	$\mathrm{SE}^2_{v_i}$	$\mathrm{SE}^3_{v_i}$	$\mathrm{SE}^4_{v_i}$	
0.00	0.00	-1	-2	1	-4	1
0.333	0.00	-1	-2	2	1	1
0.667	0.00	-1	-2	3	2	1
1.00	0.00	-1	-2	-3	3	1
0.00	0.50	-1	1	4	-4	1
0.333	0.50	1	2	5	4	1
0.667	0.50	2	3	6	5	1
1.00	0.50	3	-2	-3	6	1
0.00	1.00	-1	4	-3	-4	1
0.333	1.00	4	5	-3	-4	1
0.667	1.00	5	6	-3	-4	1
1.00	1.00	6	-2	-3	-4	1

Table 3.12: **VERTEX** table for the mesh shown in Figure 3.6.

Bottom/Left Vertex Number (v_{LL})	Top/Right Vertex Number (v_{UR})	Maximum Continuity Enforced P_{c_i}
1	2	3
2	3	3
3	4	3
5	6	3
6	7	3
7	8	3
9	10	3
10	11	3
11	12	3
1	5	3
2	6	3
3	7	3
4	8	3
5	9	3
6	10	3
7	11	3
8	12	3

Table 3.13: **EDGE** table for the mesh shown in Figure 3.6.

Global Vertex Numbers				Global Edge Numbers				Neighbor Element Numbers				Maximum Edge Approximation Degrees			
v_1^i	v_2^i	v_3^i	v_4^i	e_1^i	e_2^i	e_3^i	e_4^i	δ_1^i	δ_2^i	δ_3^i	δ_4^i	P_{e1}^i	P_{e2}^i	P_{e3}^i	P_{e4}^i
1	2	6	5	1	11	4	10	-1	2	4	-4	5	5	5	5
2	3	6	5	2	11	4	10	-1	-2	4	1	5	5	5	5
3	4	8	7	3	13	6	12	-1	-2	6	2	5	5	5	5
5	6	10	9	4	15	7	14	1	5	-3	-4	5	5	5	5
6	7	11	10	5	16	8	15	2	6	-3	4	5	5	5	5
7	8	12	11	6	17	9	16	3	-2	-3	5	5	5	5	5

Table 3.14: **ELEMENT** table for the mesh shown in Figure 3.6.

Edge 1	Edge 2	Edge 3	$\cdots$	Edge 17	End
1	3	5	$\cdots$	33	35

Table 3.15: **U-edge-pt** table for the mesh shown in Figure 3.6.

Element Δ_i	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	End
V_1^i - Begin	1	19	37	55	73	91	109
e_1^i - Begin	5	23	41	59	77	95	0
e_2^i - Begin	7	25	43	61	79	97	0
e_3^i - Begin	9	27	45	63	81	99	0
e_4^i - Begin	11	29	47	65	83	101	0
Δ_{int}^i - Begin	13	31	49	67	85	103	0

Table 3.16: **U-elem-pt** table for the mesh shown in Figure 3.6.

Example 3. We consider a rectangular domain on $[0,1] \times [0,1]$ and apply the 6-element mesh shown in Figure 3.6. We use mixed approximation degrees and enforce various levels of continuity which are shown in Figure 3.7, where solid lines indicate discontinuous modes, dotted lines indicate continuity enforced up to linear terms, dash-dot lines indicate continuity enforced up to quadratic terms, and dashed lines indicate continuity enforced up to cubic terms. The values P_i indicate the degree of the approximation mode and a value of ϕ for the interior modes means that no interior modes are used. Notice that the last column of **VERTEX** shown in Table 3.17 contains both ones and zeros corresponding to the appropriate continuity level enforced on that vertex. For example vertex number 4 is discontinuous while vertex number 6 is continuous even though edge 15 which joins vertex 6 is discontinuous. In **EDGE**, shown in Table 3.18, if an edge has a level of continuity enforced higher than linear terms, the values in the third column reflect that level of continuity appropriately. This example also illustrates how the individual entries in the last four columns of **ELEMENT**, shown in Table 3.19, reflect the approximation degree used on the edges for a particular element and that they need not be uniform as in the first element which is shown in row one. Tables 3.20 and 3.21 illustrate **U-edge-pt** and **U-elem-pt** respectively.

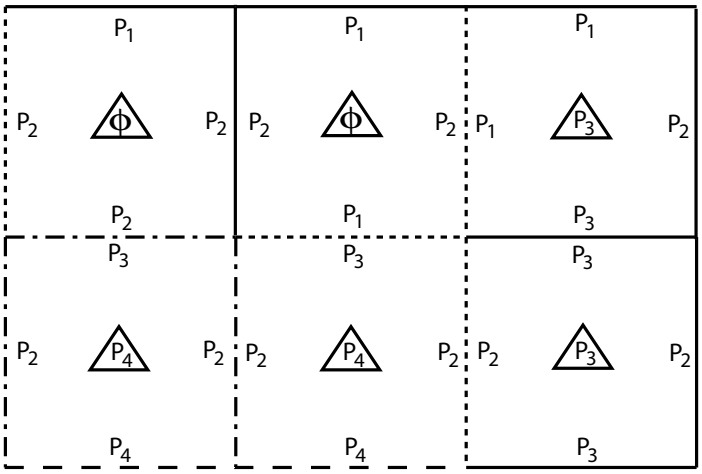


Figure 3.7: Mesh for *Example 3* showing the levels of continuity c and approximation degrees P_i, where solid lines denote $c = 0$, dotted lines denote $c = 1$, dash-dot lines denote $c = 2$, and dashed lines denote $c = 3$.

X Coordinate $x(v_i)$	Y Coordinate $y(v_i)$	Elements Sharing a Vertex				Continuity Indicator ϵ_i
		$\mathrm{SE}^1_{v_i}$	$\mathrm{SE}^2_{v_i}$	$\mathrm{SE}^3_{v_i}$	$\mathrm{SE}^4_{v_i}$	
0	0	-1	-2	1	-4	1
0.3333	0	-1	-2	2	1	1
0.6667	0	-1	-2	3	2	1
1	0	-1	-2	-3	3	0
0	0.5	-1	1	4	-4	1
0.3333	0.5	1	2	5	4	1
0.6667	0.5	2	3	6	5	1
1	0.5	3	-2	-3	6	0
0	1	-1	4	-3	-4	1
0.3333	1	4	5	-3	-4	0
0.6667	1	5	6	-3	-4	1
1	1	6	-2	-3	-4	0

Table 3.17: **VERTEX** table for the mesh shown in Figure 3.7.

Bottom/Left Vertex Number (v_{LL})	Top/Right Vertex Number (v_{UR})	Maximum Continuity Enforced P_{c_i}
1	2	3
2	3	3
3	4	0
5	6	2
6	7	0
7	8	0
9	10	0
10	11	0
11	12	0
1	5	2
2	6	2
3	7	0
4	8	0
5	9	0
6	10	0
7	11	0
8	12	0

Table 3.18: **EDGE** table for the mesh shown in Figure 3.7.

Global Vertex Numbers				Global Edge Numbers				Neighbor Element Numbers				Maximum Edge Approximation Degrees			
v_1^i	v_2^i	v_3^i	v_4^i	e_1^i	e_2^i	e_3^i	e_4^i	δ_1^i	δ_2^i	δ_3^i	δ_4^i	P_{e1}^i	P_{e2}^i	P_{e3}^i	P_{e4}^i
1	2	6	5	1	11	4	10	-1	2	4	-4	4	2	3	2
2	3	7	6	2	12	5	11	-1	3	5	1	4	2	3	2
3	4	8	7	3	13	6	12	-1	-2	6	2	3	2	3	2
5	6	10	9	4	15	7	14	1	5	-3	-4	2	2	1	2
6	7	11	10	5	16	8	15	2	6	-3	4	1	2	1	2
7	8	12	11	6	17	9	16	3	-2	-3	5	3	2	1	1

Table 3.19: **ELEMENT** table for the mesh shown in Figure 3.7.

Edges 1 - 9	1	3	5	5	6	6	6	6	6
Edges 10 - 17	6	7	8	8	8	8	8	8	8

Table 3.20: **U-edge-pt** table for the mesh shown in Figure 3.7.

Element Δ_i	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	End
V_1^i - Begin	1	10	21	32	38	44	52
e_1^i - Begin	5	14	25	36	42	48	0
e_2^i - Begin	6	15	27	36	42	50	0
e_3^i - Begin	6	16	28	37	43	51	0
e_4^i - Begin	7	18	30	37	43	51	0
Δ_{int}^i - Begin	7	18	31	38	44	51	0

Table 3.21: **U-elem-pt** table for the mesh shown in Figure 3.7.

Example 4. We consider the 9-element mesh shown in Figure 3.8 with mixed approximation degrees and no continuity enforced. In Figure 3.8 we specify the global numbering of the elements and their modes while in Figure 3.9 we give the approximation degrees P_i for each element and its modes. This example is provided to show how our data structures handle an element with a constant approximation. Since there is no continuity enforced on any element, the last columns of **VERTEX** and **EDGE** shown in Tables 3.22 and 3.23 respectively, are all zeros. Notice that the last four entries in **ELEMENT**, shown in Table 3.24, are all zero which corresponds to element number nine having a constant approximation, this is also reflected in **U-elem-pt**, shown in Table 3.25, where in column Δ_9 the difference between rows e_1^i and V_1^i is only one, and that all the other values in that column are the same value as row e_1^i, namely 79.

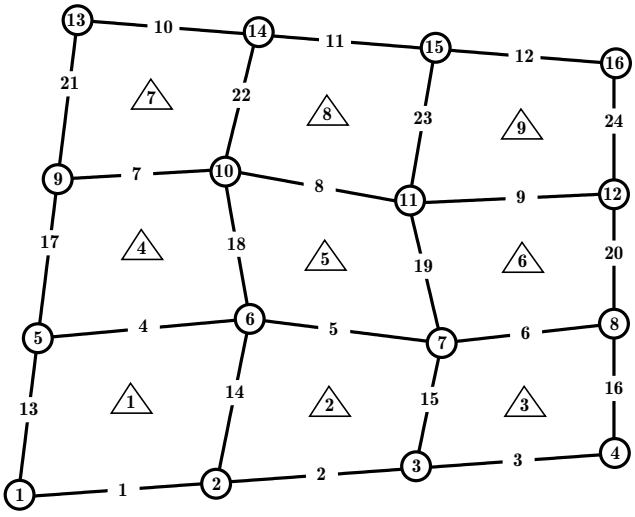


Figure 3.8: The nine-element mesh used in *Example 4.*

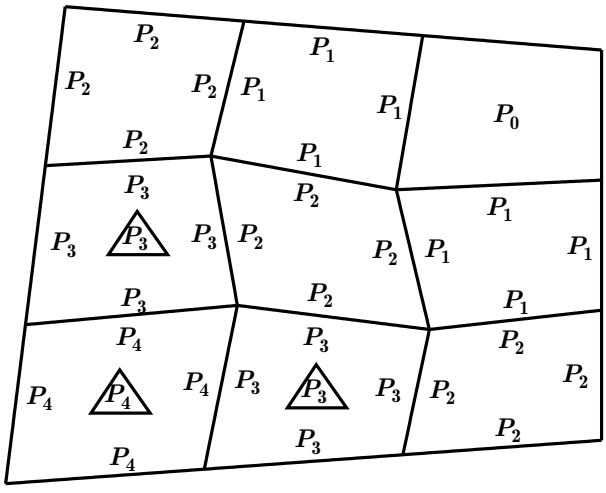


Figure 3.9: The approximation degrees for the mesh shown in Figure 3.8.

X Coordinate $x(v_i)$	Y Coordinate $y(v_i)$	Elements Sharing a Vertex				Continuity Indicator ϵ_i
		$\mathrm{SE}^1_{v_i}$	$\mathrm{SE}^2_{v_i}$	$\mathrm{SE}^3_{v_i}$	$\mathrm{SE}^4_{v_i}$	
0	0	-1	-2	1	-4	0
0.3356	0.0678	-1	-2	2	1	0
0.6689	0.1311	-1	-2	3	2	0
1	0.19	-1	-2	-3	3	0
0.0688	0.4042	-1	1	4	-4	0
0.4473	0.4925	1	2	5	4	0
0.7547	0.3783	2	3	6	5	0
1	0.46	3	-2	-3	6	0
0.1351	0.8036	-1	4	7	-4	0
0.3606	0.8468	4	5	8	7	0
0.6456	0.6889	5	6	9	8	0
1	0.73	6	-2	-3	9	0
0.199	1.198	-1	7	-3	-4	0
0.464	1.128	7	8	-3	-4	0
0.731	1.062	8	9	-3	-4	0
1	1	9	-2	-3	-4	0

Table 3.22: **VERTEX** table for the mesh shown in Figures 3.8 and 3.9.

Bottom/Left Vertex Number (v_{LL})	Top/Right Vertex Number (v_{UR})	Maximum Continuity Enforced P_{c_i}
1	2	0
2	3	0
3	4	0
5	6	0
6	7	0
7	8	0
9	10	0
10	11	0
11	12	0
13	14	0
14	15	0
15	16	0
1	5	0
2	6	0
3	7	0
4	8	0
5	9	0
6	10	0
7	11	0
8	12	0
9	13	0
10	14	0
11	15	0
12	16	0

Table 3.23: **EDGE** table for the mesh shown in Figures 3.8 and 3.9.

Global Vertex Numbers				Global Edge Numbers				Neighbor Element Numbers				Maximum Edge Approximation Degrees			
v_1^i	v_2^i	v_3^i	v_4^i	e_1^i	e_2^i	e_3^i	e_4^i	δ_1^i	δ_2^i	δ_3^i	δ_4^i	P_{e1}^i	P_{e2}^i	P_{e3}^i	P_{e4}^i
1	2	6	5	1	14	4	13	-1	2	4	-4	4	4	4	4
2	3	7	6	2	15	5	14	-1	3	5	1	3	3	3	3
3	4	8	7	3	16	6	15	-1	-2	6	2	2	2	2	2
5	6	10	9	4	18	7	17	1	5	7	-4	3	3	3	3
6	7	11	10	5	19	8	18	2	6	8	4	2	2	2	2
7	8	12	11	6	20	9	19	3	-2	9	5	1	1	1	1
9	10	14	13	7	22	10	21	4	8	-3	-4	2	2	2	2
10	11	15	14	8	23	11	22	5	9	-3	7	1	1	1	1
11	12	16	15	9	24	12	23	6	-2	-3	8	0	0	0	0

Table 3.24: **ELEMENT** table for the mesh shown in Figures 3.8 and 3.9.

Element Δ_i	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	End
V_1^i - Begin	1	20	33	41	54	62	66	74	78	79
e_1^i - Begin	5	24	37	45	58	66	70	78	79	0
e_2^i - Begin	8	26	38	47	59	66	71	78	79	0
e_3^i - Begin	11	28	39	49	60	66	72	78	79	0
e_4^i - Begin	14	30	40	51	61	66	73	78	79	0
Δ_{int}^i - Begin	17	32	41	53	62	66	74	78	79	0

Table 3.25: **U-elem-pt** table for the mesh shown in Figures 3.8 and 3.9.

3.3 Numerical Integration

For the work presented in this dissertation, we need to compute oriented line integrals arising from the use of Green's theorem and double integrals over general quadrilateral and triangular domains. Here we present a summary of the Gaussian quadratures used in this work beginning with the development of general 1-D Gaussian quadratures and their errors, followed by a product type Gaussian quadrature over the square $[-1,1]\times[-1,1]$ which we extend to general quadrilaterals using a bilinear transformation. We conclude by introducing triangular coordinates and using them in conjunction with a non-product type Gaussian quadrature that is symmetrical for double integrals over triangular domains.

3.3.1 One-Dimensional Quadrature

Given an integrable function $f(\xi)$ on $[-1,1]$, define

$$I(f)=\int_{-1}^{1} f(\xi)w(\xi)d\xi, \tag{3.2}$$

where $w(\xi)$ is a fixed nonnegative weight function. We seek an approximation to (3.2) of the form

$$I(f)\approx\sum_{i=1}^{n} W_i f(\xi_i). \tag{3.3}$$

To do this we will first rely on interpolating polynomials of $f(\xi)$. So let $g(\xi)$ be the unique polynomial of degree $n-1$ that interpolates $f(\xi)$ at the points $\{\xi_i\}_{i=1}^{n}$. Then from standard interpolation theory we know that

$$g(\xi)=\sum_{i=1}^{n} f(\xi_i)L_i(\xi), \tag{3.4}$$

where

$$L_i(\xi)=\prod_{\substack{i=1\\ i\neq j}}^{n}\frac{\xi-\xi_i}{\xi_j-\xi_i}. \tag{3.5}$$

Then we have $I(f)\approx I(g)$ where

$$I(g) = \int_{-1}^{1} g(\xi)w(\xi)d\xi=\int_{-1}^{1}(\sum_{i=1}^{n} f(\xi_i)L_i(\xi))w(\xi)d\xi \tag{3.6a}$$

$$= \sum_{i=1}^{n} f(\xi_i)\int_{-1}^{1} L_i(\xi)w(\xi)d\xi=\sum_{i=1}^{n} W_i f(\xi_i), \tag{3.6b}$$

with

$$W_i = \int_{-1}^{1} L_i(\xi) w(\xi) d\xi, \qquad 1 \leq i \leq n \ , \tag{3.6c}$$

being the weights of the quadrature which do not depend on the function f. The points $\{\xi_i\}_{i=1}^n$ are called the nodes of the quadrature. If $f(\xi) \in \mathcal{P}_{n-1}$ then $f(\xi) = g(\xi)$ and $I(f) = I(g)$, thus the quadrature gives the exact value for all polynomials up to degree $n-1$. Quadratures of this type are known as interpolatory.

We can do better, since there are n weights and n nodes that have no a priori restrictions on them, we can form a nonlinear system of $2n$ equations in $2n$ unknowns using

$$\int_{-1}^{1} \xi^k w(\xi) d\xi = \sum_{i=1}^{n} W_i \xi_i^k, \qquad 0 \leq k \leq 2n-1, \tag{3.7}$$

which can be uniquely solved to give a quadrature that is exact for all polynomials of degree $2n-1$, a quadrature of this type is called Gaussian. Notice that Gaussian quadratures require the same number of nodes and weights as do the interpolatory quadratures, but they have considerably higher precision. For this reason, we use Gaussian quadratures in our finite element scheme.
We state the existence and precision of Gaussian quadratures in Theorem (3.3), but first we give a few preliminary theorems.

Theorem 3.1. *There exists polynomials $p_j \in \bar{\Pi}_j,\ j = 0, 1, 2, \ldots,$ such that*

$$\langle p_i, p_k \rangle_{2,w} = 0 \quad \textit{for } i \neq k. \tag{3.8}$$

Proof. See Stoer and Bulirsch [81], section 3.6, page 151. □

Theorem 3.2. *The roots $\xi_i,\ i = 1, \ldots, n,$ of p_n from Theorem (3.1), are real and simple. They all lie in the open interval $(-1, 1)$.*

Proof. See Stoer and Bulirsch [81], section 3.6, page 152. □

Theorem 3.3. *The formula*

$$\int_{-1}^{1} p(\xi) w(\xi) d\xi = \sum_{j=1}^{n} W_j p(\xi_j), \tag{3.9}$$

holds for all $p(\xi) \in \mathcal{P}_{2n-1}$ if and only if $\{\xi_j\}_{j=1}^n$ are the zeros of $p_n(\xi)$ from Theorem (3.1) and the $\{W_j\}_{j=1}^n$ are given in (3.6c).

Proof. See Johnson and Riess [61], section 6.5, page 326. □

Lemma 3.1. *In a Gaussian quadrature formula, the weights W_j are all positive and their sum is $\int_{-1}^{1} w(\xi) d\xi$.*

Proof. See Kincaid and Cheney [62], section 7.3, page 533. □

Theorem 3.4. *If $f(\xi) \in C^0[-1,1]$, then the approximation*

$$\int_{-1}^{1} f(\xi)w(\xi)d\xi \approx \sum_{i=1}^{n} W_i f(\xi_i), \quad n \geq 0, \tag{3.10}$$

converges to the integral as $n \to \infty$.

Proof. See Kincaid and Cheney [62], section 7.3, page 533. □

Theorem 3.5. *For functions $f(\xi) \in C^{2n}[-1,1]$, the error E in the Gaussian quadrature,*

$$\int_{-1}^{1} f(\xi)w(\xi)d\xi = \sum_{i=1}^{n} W_i f(\xi_i) + E, \tag{3.11}$$

is such that there exists $\bar{\xi} \in (-1,1)$ such that

$$E = \frac{f^{2n}(\bar{\xi})}{(2n)!} \int_{-1}^{1} w(x) \prod_{j=1}^{n} (\xi - \xi_j)^2 d\xi. \tag{3.12}$$

Proof. See Kincaid and Cheney [62], section 7.3, page 533. □

For the special, yet significant, case in which $w(\xi) = 1$, the polynomials defined in Theorem (3.1) are Legendre polynomials and the nodes $\{\xi_j\}_{j=1}^{n}$ are just the zeros of the Legendre polynomial of degree $n-1$, hence the quadrature is often called Gauss-Legendre and the associated weights W_i are called Christoffel weights. We note that the zeros of the Legendre polynomials are symmetric about the origin and for the respective quadrature nodes $-\xi$ and ξ, the associated weights are equal. This fact helps greatly in the storage of the weights and the nodes.

Furthermore, if we let m be the order of the Gaussian quadrature used, that is to say, m is the number of nodes, then to integrate a degree n polynomial exactly we should choose m such that

$$m = \texttt{ceil}(\frac{n+1}{2}), \tag{3.13}$$

where $\texttt{ceil}(\,\cdot\,)$ is the greatest integer function defined in (1.22).

Table 3.26 list the zeros of the Legendre polynomials, $P_m(\xi)$, for degrees $m = 1$ to 12 and the corresponding Christoffel weights, W_i. These values were determined using a modified **Matlab** code called **qrule.m** from the NIT toolbox which contained a reference to (Davis and Rabinowitz (1975) 'Methods of Numerical Integration', page 365, Academic Press). These values were checked against the values listed by Stroud [83].

m	$\pm\xi_i$	W_i
1	0.00000 00000 00000	2.00000 00000 00000
2	0.57735 02691 89626	1.00000 00000 00000
3	0.00000 00000 00000	0.88888 88888 88889
	0.77459 66692 41484	0.55555 55555 55555
4	0.33998 10435 84856	0.65214 51548 62546
	0.86113 63115 94053	0.34785 48451 37454
5	0.00000 00000 00000	0.56888 88888 88889
	0.53846 93101 05683	0.47862 86704 99366
	0.90617 98459 38664	0.23692 68850 56189
6	0.23861 91860 83197	0.46791 39345 72691
	0.66120 93864 66264	0.36076 15730 48139
	0.93246 95142 03152	0.17132 44923 79170
7	0.00000 00000 00000	0.41795 91836 73469
	0.40584 51513 77397	0.38183 00505 05119
	0.74153 11855 99394	0.27970 53914 89277
	0.94910 79123 42759	0.12948 49661 68869
8	0.18343 46424 95650	0.36268 37833 78362
	0.52553 24099 16329	0.31370 66458 77887
	0.79666 64774 13627	0.22238 10344 53374
	0.96028 98564 97536	0.10122 85362 90376
9	0.00000 00000 00000	0.33023 93550 01260
	0.32425 34234 03809	0.31234 70770 40003
	0.61337 14327 00590	0.26061 06964 02935
	0.83603 11073 26636	0.18064 81606 94858
	0.96816 02395 07626	0.08127 43883 61575
10	0.14887 43389 81631	0.29552 42247 14753
	0.43339 53941 29247	0.26926 67193 09996
	0.67940 95682 99024	0.21908 63625 15982
	0.86506 33666 88985	0.14945 13491 50581
	0.97390 65285 17172	0.06667 13443 08688
11	0.00000 00000 00000	0.27292 50867 77901
	0.26954 31559 52345	0.26280 45445 10247
	0.51909 61292 06812	0.23319 37645 91990
	0.73015 20055 74049	0.18629 02109 27734
	0.88706 25997 68095	0.12558 03694 64905
	0.97822 86581 46057	0.05566 85671 16174
12	0.12523 34085 11469	0.24914 70458 13403
	0.36783 14989 98180	0.23349 25365 38355
	0.58731 79542 86617	0.20316 74267 23066
	0.76990 26741 94305	0.16007 83285 43346
	0.90411 72563 70475	0.10693 93259 95318
	0.98156 06342 46719	0.04717 53363 86512

Table 3.26: Zeros $\xi_i,\ i = 1, \ldots, m$ of Legendre polynomial P_m and corresponding Christoffel weights W_i.

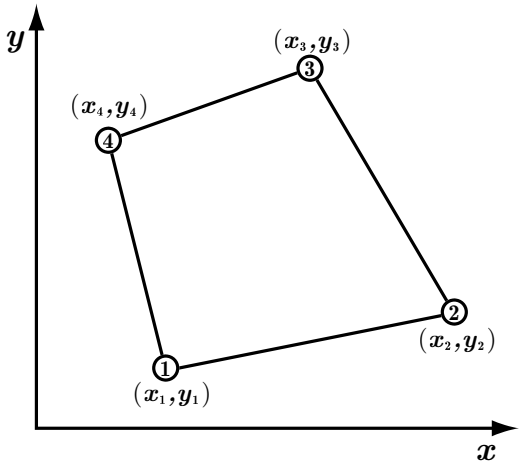


Figure 3.10: An arbitrary quadrilateral physical element.

We are interested in evaluating an oriented line integral that arises from the use of Green's theorem. Given an element Δ_{ij}, see Figure 3.10, we need to evaluate an integral of the form:

$$I(f) = \oint_{\partial\Delta_{ij}} f(x,y)d\sigma. \tag{3.14}$$

Let $\{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)\}$ be the (x,y) coordinates of Δ_{ij} with corresponding edges, e_1, e_2, e_3, e_4, where e_1 has vertices (x_1,y_1) and (x_2,y_2), so that $\partial\Delta_{ij} = \bigcup_{i=1}^{4} e_i$. Then Equation (3.14) becomes

$$I(f) = \oint_{\bigcup_{i=1}^{4} e_i} f(x,y)d\sigma. \tag{3.15}$$

Now for each edge we introduce a smooth parametrization of e_i of the form

$$r_i(t) = \frac{1-t}{2}\begin{bmatrix} x_i \\ y_i \end{bmatrix} + \frac{1+t}{2}\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix}, \qquad -1 \le t \le 1, \tag{3.16}$$

where $\begin{bmatrix} x_5 \\ y_5 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$. We define $|dr_i(t)| = \|\frac{dr_i(t)}{dt}\|_2$, where $\|\cdot\|_2$ is the usual Euclidean norm. Since our parametrizations $r_i(t)$ are all linear, $|dr_i(t)|$ will be a constant on each edge and we will drop the dependence on t. We can now transform $d\sigma$ over each edge e_i, to get $d\sigma = |dr_i|dt$ and Equation (3.15) becomes

$$I(f) = \sum_{i=1}^{4} \int_{-1}^{1} f(\ r_i(t)\ )\ |dr_i|\ dt. \tag{3.17}$$

We can now apply a Gauss-Legendre quadrature, i.e. $w(\xi) = 1$, on each of the edges to get

$$I(f) \approx \sum_{i=1}^{4} |dr_i| \sum_{j=1}^{N_i} W_j \; f(\; r_i(\xi_j) \;), \tag{3.18}$$

where N_i is the order of the quadrature used for edge, e_i.

3.3.2 Two-Dimensional Quadrature

Quadrilateral domains

Having developed one dimensional Gaussian quadratures we move on two dimensional quadratures and begin by considering integration over the canonical square element shown in Figure 3.11. It is customary to use tensor-products of the one-dimensional quadrature formulas when integrating over squares. To see how this works consider the following iterated integral

$$Q(f) = \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta) \; d\xi d\eta. \tag{3.19}$$

Now we will use the one dimensional Gauss-Legendre formulas from above to approximate $Q(f)$. First define a new function in η

$$g(\eta) = \int_{-1}^{1} f(\xi, \eta) d\xi, \tag{3.20}$$

so that

$$Q(f) \;\; = \;\; \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta) \; d\xi d\eta \tag{3.21a}$$

$$= \;\; \int_{-1}^{1} g(\eta) d\eta. \tag{3.21b}$$

We perform our first approximation to $Q(f)$ by using a Gauss-Legendre quadrature on (3.21b) with nodes $\{\eta_i\}_{i=1}^{n}$ and weights $\{W_i\}_{i=1}^{n}$ to get

$$Q(f) = \int_{-1}^{1} g(\eta) d\eta \approx \sum_{i=1}^{n} W_i \; g(\eta_i). \tag{3.22}$$

Next we approximate $g(\eta_i)$ with $G(\eta_i)$ which is another Gauss-Legendre quadrature with nodes $\{\xi_j\}_{j=1}^{m}$ and weights $\{V_j\}_{j=1}^{m}$ to get

$$g(\eta_i) = \int_{-1}^{1} f(\xi, \eta_i) d\xi \approx G(\eta_i) = \sum_{j=1}^{m} V_j \; f(\xi_j, \eta_i). \tag{3.23}$$

We make our final approximation to $Q(f)$ by substituting $G(\eta_i)$ into (3.22) for $g(\eta_i)$ to get

$$Q(f) \approx \sum_{i=1}^{n} W_i \; g(\eta_i) \approx \sum_{i=1}^{n} W_i \; G(\eta_i), \tag{3.24}$$

which gives our final form

$$Q(f) \approx \sum_{i=1}^{n} \sum_{j=1}^{m} W_i \; V_j \; f(\xi_j, \eta_i). \tag{3.25}$$

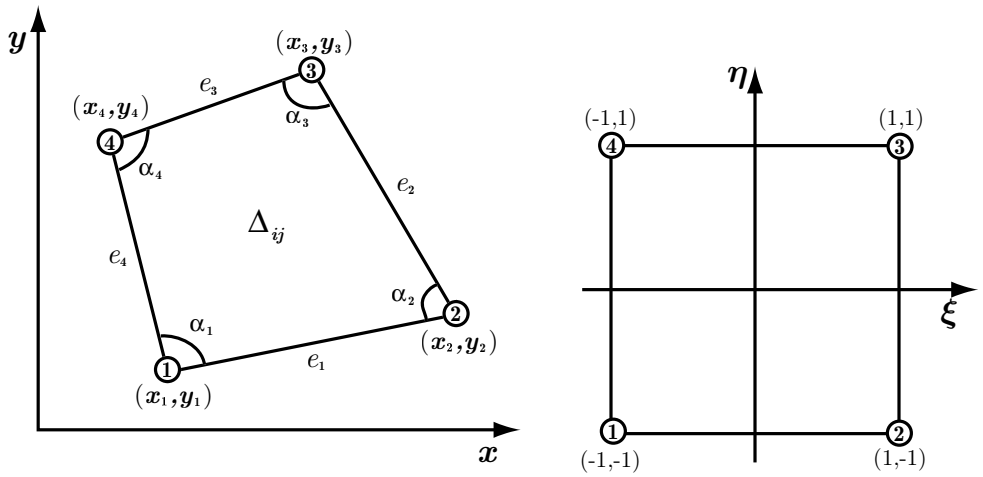


Figure 3.11: Bilinear mapping of a quadrilateral to the canonical element $[-1,1]^2$.

Product type quadratures for multiple integrals are easy to use and to code, since all the one-dimensional nodes and weights are usually already in place. Their errors behave in a similar fashion to the one-dimensional formulas. The major draw back to product type quadratures is their cost. In two dimensions the cost for using an order n quadrature in ξ and an order m quadrature in η would be nm function evaluations. Clearly in higher dimensions, d, if we use uniform orders n, the cost would be n^d. There are special cases such as integrating over, rectangles, triangles, or circles, where non-product type quadratures have been developed and require much fewer function evaluations to obtain similar precision. Such quadratures would have the usual form

$$\int_{\Omega} w(\boldsymbol{x}) f(\boldsymbol{x}) d\Omega \approx \sum_{i=1}^{n} W_i f(\boldsymbol{x}_i), \tag{3.26}$$

where Ω is the region of integration and $\boldsymbol{x}$ is a vector of length d. We will use such a formula for integration over a triangular region and give a cost comparison with product type quadratures. A possible improvement for the code used in this work, would be to employ one of these non-product type quadratures for integration over our quadrilateral elements.

Now that we have established how to integrate over the square region $[-1,1]\times[-1,1]$ we turn our attention to integrating over general quadrilaterals. Again consider a general quadrilateral element Δ_{ij} like the one shown in Figure 3.11 and the problem

$$Q(f)=\iint_{\Delta_{ij}} f(x,y)dxdy. \tag{3.27}$$

To use the Gaussian quadratures we derived for the square region, we will make a change of variable using a transformation from the square to a general quadrilateral. From advanced calculus we have the following theorem which establishes under what conditions we can maintain equality of our integrals when using a change of variable.

Theorem 3.6. *Suppose that* $\mathbf{T}$ *is a* C^1 *invertible transformation that maps a rectangular region* R *in the* $\xi\eta$*-plane onto a region* S *in the* xy*-plane. Suppose that* f *is continuous on* S *except on a finite number of smooth curves. Then*

$$\iint_S f(x,y)\ dxdy=\iint_R f(\mathbf{T}(\xi,\eta))\ \det\left(\mathbf{J}[\mathbf{T}(\xi,\eta)]\right)\ d\xi d\eta, \tag{3.28}$$

where $\mathbf{J}[\mathbf{T}]$ *is the Jacobian of the transformation* $\mathbf{T}$ *and* $\det\left(\mathbf{J}[\mathbf{T}(\xi,\eta)]\right)$ *is its determinant.*

For our purpose we use the bilinear mapping $\mathbf{T}$ from (ξ,η) onto (x,y) defined as

$$\mathbf{T}(\xi,\eta)=\begin{bmatrix}x(\xi,\eta)\\ y(\xi,\eta)\end{bmatrix}=\sum_{k=1}^{4}\begin{bmatrix}x_k\\ y_k\end{bmatrix}\Phi^1_{v_k}(\xi,\eta), \tag{3.29}$$

where $\Phi^1_{v_k}(\xi,\eta)$ is the usual k^{th} vertex shape function and $[x_k,y_k]^\top$ are the coordinates of the physical domain element, listed in a counter clockwise manner, see Figure 3.11. The Jacobian of $\mathbf{T}$ is given by

$$\mathbf{J}[\mathbf{T}(\xi,\eta)]=\frac{1}{2}\mathbf{H}\ \mathbf{N}, \tag{3.30a}$$

where

$$\mathbf{H}=\begin{bmatrix}x_2-x_1 & x_3-x_4 & x_4-x_1 & x_3-x_2\\ y_2-y_1 & y_3-y_4 & y_4-y_1 & y_3-y_2\end{bmatrix}, \tag{3.30b}$$

$$\mathbf{N}=\begin{bmatrix}\bar{N}_1(\eta) & 0\\ \bar{N}_2(\eta) & 0\\ 0 & \bar{N}_1(\xi)\\ 0 & \bar{N}_2(\xi)\end{bmatrix}, \tag{3.30c}$$

and $\bar{N}_1(t)=\frac{1-t}{2}$, $\bar{N}_2(t)=\frac{1+t}{2}$. The determinant, $\det\left(\mathbf{J}[\mathbf{T}(\xi,\eta)]\right)$, is easily obtained from (3.30a) which is just a 2×2 matrix.

Lemma 3.2. *The bilinear mapping* $\mathbf{T}$ *from (3.29) is* C^1 *invertible on* $R=[-1,1]\times[-1,1]$ *to* $S=\Delta_{ij}$ *provided the physical element* Δ_{ij} *is convex.*

Proof. See Flaherty, [50], Chapter 5. □

Having established that our bilinear mapping $\mathbf{T}$ is C^1 invertible allows us to use Theorem (3.6) for (3.27). Furthermore, we are guaranteed that $\mathbf{T}$ is one-to-one.

To approximate (3.27) we make a change of variable using (3.29) and apply our product type Gaussian quadrature where (ξ_i, η_j) and W_i, V_j are respectively the nodes and weights.

$$Q(f) = \iint_{\Delta_{ij}} f(x,y)dxdy \tag{3.31a}$$

$$= \int_{-1}^{1}\int_{-1}^{1} f(x(\xi,\eta), y(\xi,\eta))\ \det\left(\mathbf{J}[\mathbf{T}(\xi,\eta)]\right)\ d\xi d\eta \tag{3.31b}$$

$$\approx \sum_{i=1}^{n}\sum_{j=1}^{m} W_i\ V_j\ f(x(\xi_i,\eta_j), y(\xi_i,\eta_j))\ \det\left(\mathbf{J}[\mathbf{T}(\xi_i,\eta_j)]\right). \tag{3.31c}$$

Note that the integral in (3.31b) involves a product of f and a linear term in ξ and η, therefore one may wish to increase the degree of the quadrature used to ensure sufficient precision. If the region R is a rectangle then $\det\left(\mathbf{J}[\mathbf{T}(\xi,\eta)]\right)$ is a constant.

Triangular domains

Although we are using quadrilateral elements for our method, we still need to evaluate an integral over a triangular domain in cases where we have a discontinuous true solution that splits the quadrilateral element into parts which might be triangular, see the discussion at the end of this chapter.

We are interested in computing a double integral over a triangular domain Ω_Δ

$$Q(f) = \iint_{\Omega_\Delta} f(x,y)\ dxdy. \tag{3.32}$$

We could employ product type Gaussian quadratures for this problem just as we did in the case of a quadrilateral domain. We will instead use non-product type Gaussian quadratures which are symmetrical and more cost efficient. Hammer *et al.* [54] developed quadrature formulas over simplexes and cones in 1956. Dunavant [43] in 1985 presented symmetrical Gaussian quadratures for the triangle up to order 20. However, some of the rules in Dunavant's work, orders $p = 11, 15, 16, 18, 20$, have nodes that lie outside the domain, for example see Figure 3.12 with $p = 11$. Without explanation of how to obtain the nodes and weights we will use the rules presented in Dunavant's work to perform our integration over the triangle. Table 3.27 gives a comparison of the number of evaluation points needed for a product type Gaussian quadrature versus the symmetrical Gaussian quadrature over a triangle. Notice

that for quadrature orders $m < 7$, that the product and non-product quadratures require nearly the same number of function evaluations. For higher order quadratures, say $m = 12$, the non-product formulas require far less evaluation points. In fact a 17^{th} order symmetrical quadrature requires less function evaluations than a 12^{th} order product type quadrature. We can avoid the rules with nodes outside the domain by increasing the order. Doing so we would obviously have more evaluation points, but we would also have an increased accuracy and the number of evaluation points in all cases would still be less than the original number in the product type quadrature.

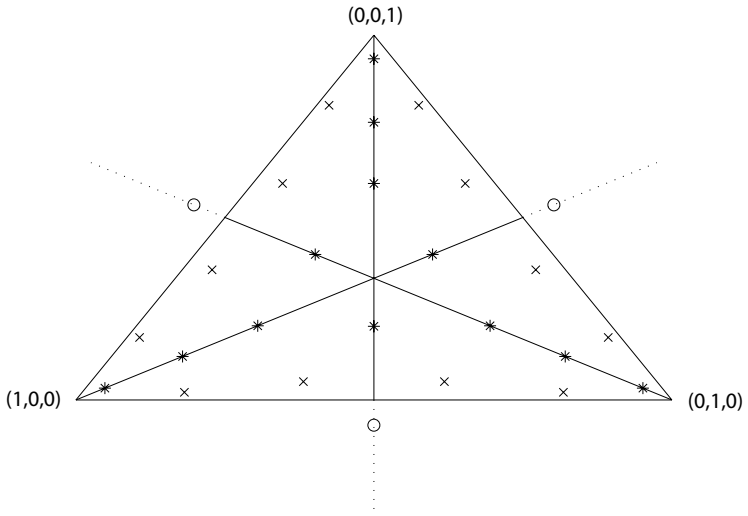


Figure 3.12: Natural triangle with the Gaussian nodes for $m = 11$. Points marked with an * are associated with a median line, points marked with an x are arbitrary interior points, and o indicates points outside the triangle but still associated with a median line.

When performing two-dimensional numerical integration over triangles, the formulas are easily stated using triangular coordinates, also known as barycentric coordinates and denoted by $(\zeta_1, \zeta_2, \zeta_3)$. We diverge here to define and explain triangular coordinates.

Consider the triangle Δ_{123} shown on the right in Figure 3.13, that has vertices numbered from one to three with Cartesian coordinates $(x_i, y_i),\ i = 1, 2, 3$ and area A_{123}. Consider a point P in the triangle with Cartesian coordinate (x, y). Connecting P to all three vertices with line segments forms three separate triangles with areas $\{A_{P12}, A_{P13}, A_{P23}\}$. Then the point P has triangular coordinates $(\zeta_1, \zeta_2, \zeta_3)$ given by

$$\zeta_1 = \frac{A_{P23}}{A_{123}}, \qquad \zeta_2 = \frac{A_{P13}}{A_{123}}, \qquad \zeta_3 = \frac{A_{P12}}{A_{123}}. \tag{3.33}$$

From this we can see that the vertices of Δ_{123}, would have the triangular coordinates listed in Table 3.28.

Order m	Product N_p	Symmetrical N_s	Difference $N_p - N_s$
1	1	1	0
2	4	3	1
3	4	4	0
4	9	6	3
5	9	7	2
6	16	12	4
7	16	13	3
8	25	16	9
9	25	19	6
10	36	25	11
11	36	27	9
12	49	33	16
13	49	37	12
14	64	42	22
15	64	48	16
16	81	52	29
17	81	61	20
18	100	70	30
19	100	73	27
20	121	79	42

Table 3.27: The number of evaluation points for product vs. symmetrical Gaussian quadratures over triangles for a given order m.

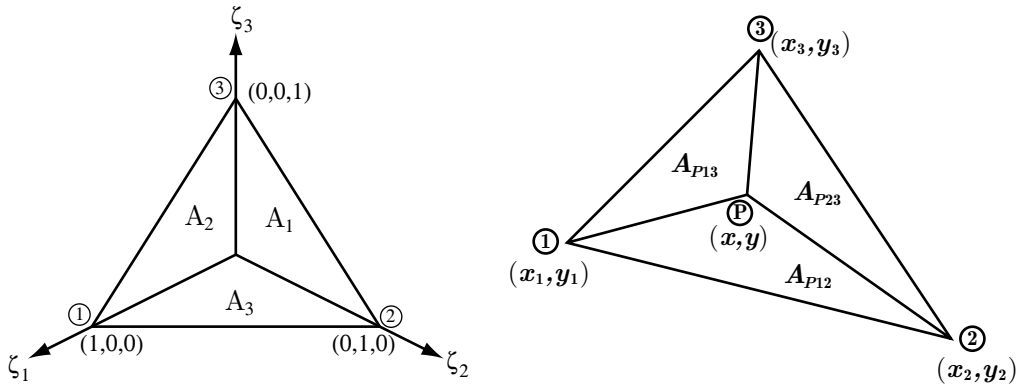


Figure 3.13: Canonical triangle with barycentric coordinates (left) and triangle Δ_{123} with an arbitrary point P (right).

Vertex Number	Cartesian Coordinate (x, y)	Triangular Coordinate $(\zeta_1, \zeta_2, \zeta_3)$
1	(x_1, y_1)	(1,0,0)
2	(x_2, y_2)	(0,1,0)
3	(x_3, y_3)	(0,0,1)

Table 3.28: Triangular coordinates of the vertices.

Finding the area of a triangle is easily done using a cross product of two edges, for example the area of triangle Δ_{123} is

$$A_{123} = \frac{1}{2}\ (e_{13} \times e_{12}) = \frac{1}{2}((x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)), \tag{3.34}$$

where e_{12} is the edge of the triangle formed by connecting vertices 1 and 2. We recall that it only takes two quantities to locate a point in the plane, thus the triangular coordinate system in redundant. This redundancy is expressed by

$$\zeta_3 = 1 - \zeta_1 - \zeta_2. \tag{3.35}$$

For a triangle with vertices $(x_i, y_i),\ i = 1, 2, 3$, we define a transformation $\mathcal{B}$ from the triangular coordinate system to the Cartesian Coordinate system

$$\mathcal{B}(\zeta_1, \zeta_2, \zeta_3) = \begin{bmatrix} x(\zeta_1, \zeta_2, \zeta_3) \\ y(\zeta_1, \zeta_2, \zeta_3) \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}. \tag{3.36}$$

Using (3.33), we can define a transformation, $\hat{\mathcal{B}}$, from the Cartesian coordinates (x, y) of the triangle Δ_{123} into the corresponding triangular coordinates $(\zeta_1, \zeta_2, \zeta_3)$ to be

$$\hat{\mathcal{B}}(x, y) = \begin{bmatrix} \zeta_1(x, y) \\ \zeta_2(x, y) \\ \zeta_3(x, y) \end{bmatrix} = \frac{1}{2A_{123}} \begin{bmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2 \\ x_1y_3 - x_3y_1 & y_1 - y_3 & x_3 - x_1 \\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}, \tag{3.37}$$

where A_{123} is given in (3.34). We are ready to use these in our Gaussian quadrature formula

$$Q(f) = \iint_{\Omega_\Delta} f(x, y)\ dxdy \approx A_\Omega \sum_{i=1}^{n} W_i\ f(\zeta_1^i, \zeta_2^i, \zeta_3^i), \tag{3.38}$$

where A_Ω is the area of the triangular region Ω_Δ and $(\zeta_1^i, \zeta_2^i, \zeta_3^i)$ are the i^{th} triangular coordinates of the quadrature nodes. Tables 3.31 and 3.32 give the weights, W_i, and the triangular coordinates, $(\zeta_1^i, \zeta_2^i, \zeta_3^i)$ of the symmetrical Gaussian quadrature over triangles for orders $p = 1$ to 11. These values can be found in Dunavant [43] along with the values for orders up to 20. The value M in the table is the number of unique permutations associated with an evaluation point with corresponding weight W_i. The value n is the sum of all the M's and

is the total number of function evaluations required by the quadrature. We note that the factor $M = 1$ is always associated with the triangle's centroid $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, the factor $M = 3$ is associated with a point on a median line, and $M = 6$ is associated with an arbitrary point in the interior. To illustrate the symmetrical Gaussian quadrature over triangular domains, consider a triangle Ω_Δ with area A_Ω, using a third order quadrature with a total of $n = 4$ function evaluation points then

$$Q(f) = \iint_{\Omega_\Delta} f(x,y)\, dxdy \approx$$

$$A_\Omega \left(\frac{-9}{16} f(\mathcal{B}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})) + \frac{25}{48} \left[f(\mathcal{B}(\frac{3}{5}, \frac{1}{5}, \frac{1}{5})), f(\mathcal{B}(\frac{1}{5}, \frac{3}{5}, \frac{1}{5})), f(\mathcal{B}(\frac{1}{5}, \frac{1}{5}, \frac{3}{5})) \right] \right). \quad (3.39)$$

The storage of the weights and nodes and the execution of this symmetrical Gaussian quadrature deserve some mention. The nodes and weights are stored in separate tables, labeled *tgdat*, for each order p inside of a single **Matlab** routine. This allows only the nodes and weights currently needed to be in memory. Each table has five columns containing information for $[M,\ W_i,\ \zeta_1^i,\ \zeta_2^i,\ \zeta_3^i]$ and one row for each unique weight, W_i, see Table 3.29 for an example with order $p = 3$ and fractions equivalent to the data in Table 3.31. We use a pointer stored in a 6×3 table called *perm*, see Table 3.30, that contains the index values into *tgdat* for all the needed permutations of the quadrature nodes. To compute $\sum_{i=1}^{n} W_i\ f(\zeta_1^i, \zeta_2^i, \zeta_3^i)$ we perform two looping sums, one over the number of unique weights, the other over the number of permutations associated with that weight. In Figure 3.14 we provide a sample procedure using *tgdat* and *perm* for computing a symmetrical Gaussian quadrature over a triangular region with area equal to 1.

M	W_i	ζ_1^i	ζ_2^i	ζ_3^i
1	-9/16	1/3	1/3	1/3
3	25/48	3/5	1/5	1/5

Table 3.29: Example storage table, *tgdat* for the nodes and weights of a third order symmetrical Gaussian quadrature over a triangle.

3.3.3 A Special Case

In this work we have two examples that have a discontinuous true solution, one that has a contact discontinuity where the initial boundary conditions were discontinuous and the other Burger's equation where a shock forms away from the initial boundary, see for example Figure 1.1(B). In each case, the slope of the shock is positive and moves in a straight line. These lines intersect our rectangular elements in such a way that in order to accurately compute

3	4	5
4	5	3
5	3	4
3	5	4
4	3	5
5	4	3

Table 3.30: Table *perm*.

Variables
tgdat (is the matrix with the factors, weights and nodes)
perm (is the index value permutation matrix)
$Nrows$ (is the number of rows of *tgdat*)
M (is the factor from *tgdat*)
W (is the Gaussian weight from *tgdat*)
i (is the row loop counter for *tgdat*)
k (is the row loop counter for *perm*)
f (is the function $f[\zeta_1, \zeta_2, \zeta_3]$)
S (is the sum)

$k := 1$
$i := 1$
$S := 0$
`While` $i \leq Nrows$
    $M := tgdat(i, 1)$
    $W := tgdata(i, 2)$
    `While` $k \leq M$
        $S := S + f[tgdat(i, perm(k, 1)), tgdat(i, perm(k, 2)), tgdat(i, perm(k, 3))]$
        $k := k + 1$
    `end loop`
    $S := W * S$
    $i := i + 1$
`end loop`

Figure 3.14: A procedure for computing a symmetrical Gaussian quadrature over a triangle with area equal to 1.

p	n	M	W_i	ζ_1	ζ_2, ζ_3
1	1	1	1.00000 00000 00000	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
2	3	3	0.33333 33333 33333	0.66666 66666 66667	0.16666 66666 66667
					0.16666 66666 66667
3	4	1	-0.56250 00000 00000	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
		3	0.52083 33333 33333	0.60000 00000 00000	0.20000 00000 00000
					0.20000 00000 00000
4	6	3	0.10995 17436 55322	0.81684 75729 80459	0.09157 62135 09771
					0.09157 62135 09771
		3	0.22338 15896 78011	0.10810 30181 68070	0.44594 84909 15965
					0.44594 84909 15965
5	7	1	0.22500 00000 00000	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
		3	0.12593 91805 44827	0.79742 69853 53087	0.10128 65073 23456
					0.10128 65073 23456
		3	0.13239 41527 88506	0.05971 58717 89770	0.47014 20641 05115
					0.47014 20641 05115
6	12	3	0.05084 49063 70207	0.87382 19710 16996	0.06308 90144 91502
					0.06308 90144 91502
		3	0.11678 62757 26379	0.50142 65096 58179	0.24928 67451 70910
					0.24928 67451 70910
		6	0.08285 10756 18374	0.63650 24991 21399	0.31035 24510 33785
					0.05314 50498 44816
7	13	1	-0.14957 00444 67670	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
		3	0.17561 62574 33204	0.47930 80678 41923	0.26034 59660 79038
					0.26034 59660 79038
		3	0.05334 72356 08839	0.86973 97941 95568	0.06513 01029 02216
					0.06513 01029 02216
		6	0.07711 37608 90257	0.63844 41885 69809	0.31286 54960 04875
					0.04869 03154 25316
8	16	1	0.14431 56076 77787	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
		3	0.09509 16342 67285	0.08141 48234 14554	0.45929 25882 92723
					0.45929 25882 92723
		3	0.10321 73705 34718	0.65886 13844 96480	0.17056 93077 51760
					0.17056 93077 51760
		3	0.03245 84976 23198	0.89890 55433 65938	0.05054 72283 17031
					0.05054 72283 17031
		6	0.02723 03141 74435	0.00839 47774 09958	0.26311 28296 34638
					0.72849 23929 55404

Table 3.31: Nodes, $(\zeta_1^i, \zeta_2^i, \zeta_3^i)$ and weights, W_i, for symmetrical Gaussian quadratures on a triangle for $p = 1$ to 8.

p	n	M	W_i	ζ_1	ζ_2, ζ_3
9	19	1	0.09713 57962 82799	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
		3	0.03133 47002 27139	0.02063 49616 02525	0.48968 25191 98738
					0.48968 25191 98738
		3	0.07782 75410 04774	0.12582 08170 14127	0.43708 95914 92937
					0.43708 95914 92937
		3	0.07964 77389 27210	0.62359 29287 61935	0.18820 34346 19033
					0.18820 35356 19033
		3	0.02557 76756 58698	0.91054 09732 11095	0.04472 95133 94453
					0.04472 95133 94453
		6	0.04328 35393 77289	0.03683 84120 54736	0.22196 29891 60766
					0.74119 85987 84498
10	25	1	0.09081 79903 82754	0.33333 33333 33333	0.33333 33333 33333
					0.33333 33333 33333
		3	0.03672 59577 56467	0.02884 47332 32685	0.48557 76333 83657
					0.48557 76333 83657
		3	0.04532 10594 35528	0.78103 68490 29926	0.10948 15754 85037
					0.10948 15754 85037
		6	0.07275 79168 45420	0.14170 72194 14880	0.30793 98387 64121
					0.55035 29418 20999
		6	0.02832 72425 31057	0.02500 35347 62686	0.24667 25606 39903
					0.72832 39045 97411
		6	0.00942 16669 63733	0.00954 08154 00299	0.06680 32510 12200
					0.92365 59335 87500
11	27	3	0.00092 70063 28961	-0.06922 20965 41517	0.53461 10482 70758
					0.53461 10482 70758
		3	0.07714 95349 14813	0.20206 13940 68290	0.39896 93029 65855
					0.39896 93029 65855
		3	0.05932 29773 80774	0.59338 01991 37435	0.20330 99004 31282
					0.20330 99004 31282
		3	0.03618 45405 03418	0.76129 81754 34837	0.11935 09122 82581
					0.11935 09122 82581
		3	0.01365 97310 02678	0.93527 01037 77448	0.03236 49481 11276
					0.03236 49481 11276
		6	0.05233 71119 62204	0.05017 81383 10495	0.35662 06482 61293
					0.59320 12134 28213
		6	0.02070 76596 39141	0.02102 20165 36166	0.17148 89803 04042
					0.80748 90031 59792

Table 3.32: Nodes, $(\zeta_1^i, \zeta_2^i, \zeta_3^i)$ and weights, W_i, for symmetrical Gaussian quadratures on a triangle for $p = 9$ to 11.

the $\mathcal{L}_2$ norm of the error on that element we have to partition the domain of the element according to the line of discontinuity, into triangular and/or quadrilateral domains and then perform the necessary integration over each subdomain where the solution is continuous. When the line of discontinuity intersects our element we have nine possible domain splittings which are shown in Figure 3.15. Each splitting is identified with a unique four element array named ***SPLIT*** whose entries are either 1 or 0. The first element in the array ***SPLIT*** corresponds to local edge number 1 of the element, the second element to the second edge and so forth, see Figure 2.3 for an example of an element with its edges numbered. From the characteristic theory we developed earlier in §1.4, it is possible to determine the equation of the line of discontinuity and the location, in the domain of Ω, where the discontinuity forms for the first time. Once we know this information we locate the first element that the discontinuity intersects and then perform a search on all four edges of the element to determine if the discontinuity intersects that edge. If the discontinuity intersects that edge, then the element of ***SPLIT*** that corresponds to that edge number is given the value of 1, otherwise ***SPLIT*** has a value of 0. Additionally, we temporarily store the coordinates of the intersection points. Once the type of splitting is identified an integrating routine is called that uses ***SPLIT*** to choose the correct scheme for integrating over each subdomain defined by ***SPLIT*** and the stored intersection points. Since we know what direction the characteristic lines slope, we can also use ***SPLIT*** to point to the next element that is intersected by the discontinuity line, access that element and then start the process over again.

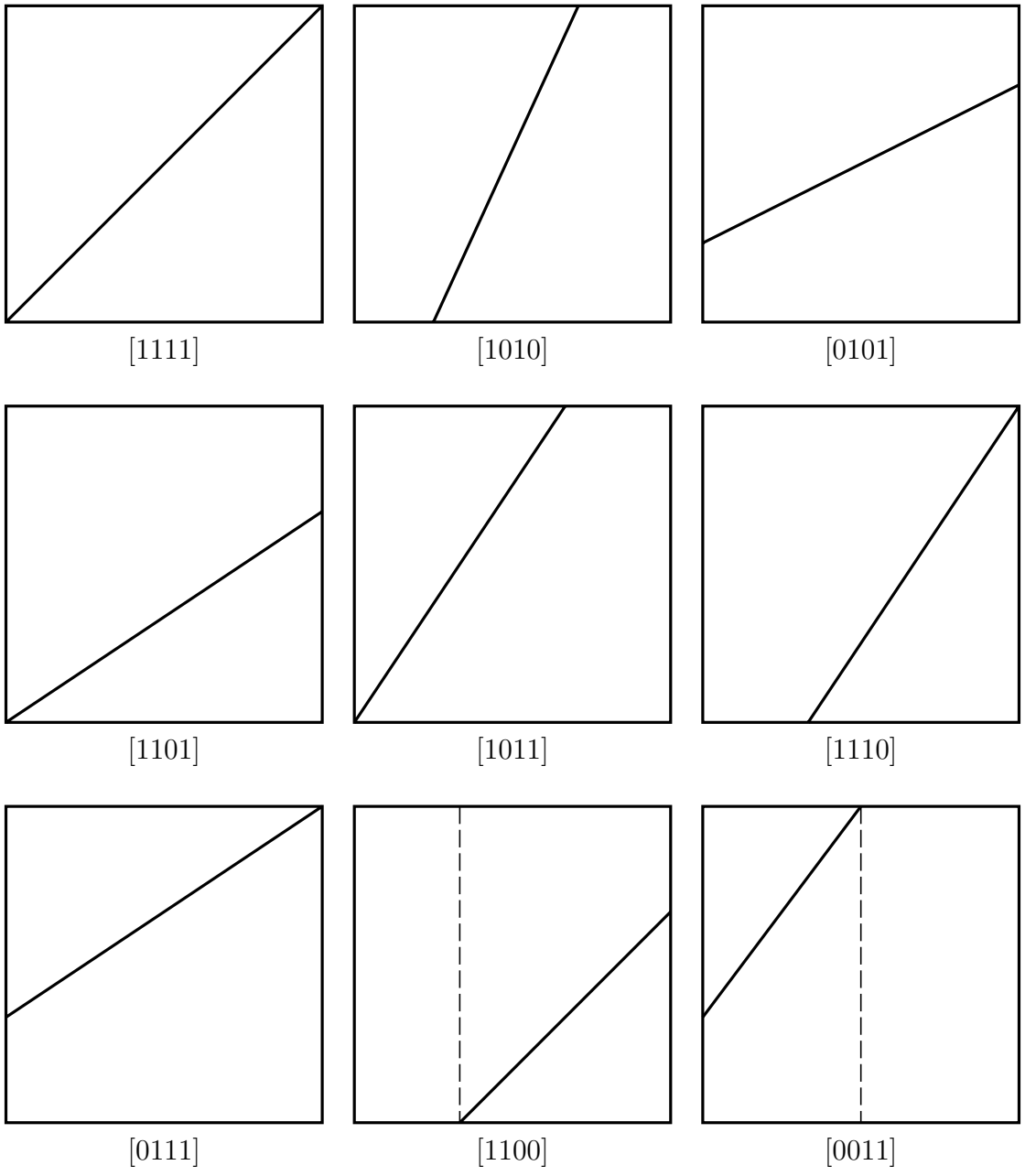


Figure 3.15: Possible divisions of a square when split by a straight line with positive slope.

Chapter 4

A Posteriori Error Estimate for Hyperbolic Problems

4.1 Introduction

In this chapter we extend the one-dimensional DGM *a posteriori* and superconvergence results of Adjerid *et al.* [5] to multi-dimensional hyperbolic problems on rectangular meshes. We show how to select the finite element space such that the leading term in the true local error is spanned by two $(p+1)-$degree Radau polynomials in the x and y directions, respectively. The p-degree discontinuous finite element solution exhibits an $O(h^{p+2})$ superconvergence at the Radau points obtained as a tensor product of the roots of Radau polynomial of degree $p+1$. For conservation laws we show that, locally, the solution flux is $O(h^{2p+2})$ superconvergent on average on the outflow element boundary. We further show that the solution flux converges at an $O(h^{2p+1})$ rate on average at the outflow boundary of the domain. For a hyperbolic problem with a nonlinear reaction term, we show that locally, the solution flux is $O(h^{\min(p+4,2p+2)})$ superconvergent on average on the outflow element boundary. We use these superconvergence results to construct asymptotically correct *a posteriori error* estimates.

These results readily extend to three-dimensional nonlinear hyperbolic problems of the form

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x + \mathbf{g}(\mathbf{u})_y + \mathbf{h}(\mathbf{u})_z + \mathbf{q}(\mathbf{u}) = 0. \tag{4.1}$$

4.2 Error Analysis

We begin our error analysis by considering the linear first-order model problem (1.6) with $\gamma = 0$,

$$\boldsymbol{\alpha} \cdot \nabla u = f(x,y), \qquad (x,y) \in \Omega = [0,h] \times [0,h], \tag{4.2a}$$

subject to the boundary conditions at the inflow boundaries

$$u(x,0) = g_1(x) \quad \text{and} \quad u(0,y) = g_2(y). \tag{4.2b}$$

We use the FGM weak formulation for this problem that we derived in (2.38) and designate all the approximate solutions $U(x,y)$ to be discontinuous, which gives the DGM. We restate the weak form here. The discontinuous Galerkin method [59] consists of determining $U(x,y) \in \mathcal{V}_p$ on Δ such that

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\,[U^- - U]V d\sigma + \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla U V dxdy \;=\; \iint_{\Delta} f(x,y)V dxdy, \quad \forall\, V \in \mathcal{V}_p. \tag{4.3}$$

Once we determine the solution on the first element Δ we proceed to the elements whose inflow boundaries are either on the inflow boundary of Ω or an outflow boundary of Δ and continue this process until the solution is determined in the whole domain. On an element whose inflow boundary is not on the boundary of Ω, U^- is defined as before to be

$$U^-(x,y) = \lim_{s\to 0^+} U((x,y) + s\boldsymbol{n}), \quad (x,y) \in \Gamma_{in}. \tag{4.4}$$

We will show that the discontinuous Galerkin local finite element error can be split into an $O(h^{p+1})$ component which is a linear combination of two right Radau polynomials of degree $p+1$ in the x and y directions, respectively, and a higher-order component. Prior to this, we need to establish the DGM orthogonality condition and prove two preliminary lemmas.

We begin by multiplying (4.2) by $V \in \mathcal{V}_p$, integrating over the element Δ and using Green's formula to obtain

$$\oint_{\Gamma} \boldsymbol{\alpha}\cdot\boldsymbol{n} V u d\sigma \;-\; \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla V u dxdy \;=\; \iint_{\Delta} f(x,y)V dxdy. \tag{4.5}$$

Applying Green's formula to (4.3) leads to

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, UV d\sigma \;+\; \oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, U^- V d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla V U dxdy \;=$$

$$\iint_{\Delta} f(x,y)V dxdy. \tag{4.6}$$

Subtracting (4.6) from (4.5) leads to the DGM orthogonality condition

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\,\epsilon^- V d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\,\epsilon V d\sigma - \int_0^h \int_0^h \boldsymbol{\alpha}\cdot\nabla V \epsilon\, dx\, dy = 0, \qquad \forall V \in \mathcal{V}_p, \tag{4.7}$$

where

$$\epsilon \;=\; u - U \tag{4.8}$$

is the local finite element discretization error. Using the mapping of $[0,h]^2$ onto the canonical element $[-1,1]^2$ defined by $x = h(1+\xi)/2$ and $y = h(1+\eta)/2$ and $\hat{u}(\xi,\eta) = u(x(\xi), y(\eta))$ we obtain the DGM orthogonality condition (4.7) on the canonical element

$$\int_{\hat{\Gamma}_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\hat{\epsilon}^{-}\hat{V}d\hat{\sigma} + \int_{\hat{\Gamma}_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\hat{\epsilon}\hat{V}d\hat{\sigma} - \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{\alpha}\cdot\nabla\hat{V}\hat{\epsilon}\,d\xi\;d\eta = 0, \qquad \forall\hat{V}\in\hat{\mathcal{V}}_p. \tag{4.9}$$

In the remainder of this paper we will omit the $\hat{}$ unless we feel it is needed for clarity.

Lemma 4.1. *If $Q_k \in \mathcal{V}_k$ satisfies*

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k V d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla V Q_k d\xi\;d\eta = 0, \qquad \forall V\in\mathcal{V}_p,\; k\le p, \tag{4.10}$$

then

$$Q_k = 0, \qquad k\le p. \tag{4.11}$$

Proof. Using Green's formula we write (4.10) as

$$-\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k V d\sigma + \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla Q_k V d\xi\;d\eta = 0, \qquad \forall V\in\mathcal{V}_p. \tag{4.12}$$

Adding (4.10) to (4.12) with $V = Q_k$ we obtain

$$-\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k^2 d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k^2 d\sigma = \oint_{\Gamma} |\boldsymbol{\alpha}\cdot\boldsymbol{n}|Q_k^2 d\sigma = 0, \qquad k\le p. \tag{4.13}$$

This leads to $Q_k = 0$ on Γ. Combining this with (4.12) for $V = \boldsymbol{\alpha}\cdot\nabla Q_k$ yields $\boldsymbol{\alpha}\cdot\nabla Q_k = 0$ on Δ which completes the proof. □

In the following lemma we will write the interpolation error as a power series in h.

Lemma 4.2. *Let $w\in C^{\infty}(0,h)$ and πw be a p-degree polynomial that interpolates w at Radau points on [0,h]. Then the interpolation error*

$$w(x(\xi)) - \pi w(x(\xi)) = \sum_{k=p+1}^{\infty} Q_k^{-}(\xi)h^k, \tag{4.14}$$

where

$$Q_{p+1}^{-}(\xi) = \frac{w^{(p+1)}(0)}{2^{p+1}(p+1)!}(\xi-\xi_0)(\xi-\xi_1)\ldots(\xi-\xi_p) = c_{p+1}R_{p+1}(\xi), \tag{4.15}$$

and

$$Q_k^{-}(\xi) = R_{p+1}(\xi)r_{k-p-1}(\xi),\;\; k > p+1, \tag{4.16}$$

with $r_k(\xi)$ being a polynomial of degree k.

Proof. By the standard interpolation error there exists $s(x)$ such that

$$w(x) - \pi w(x) = \frac{w^{(p+1)}(s(x,h))}{(p+1)!}(x-x_0)(x-x_1)\dots(x-x_p), \qquad x \in [0,h], \tag{4.17}$$

where $x_i = h(1+\xi_i)/2$ and $\xi_i, i = 0, 1, \dots, p$, are the roots of right Radau polynomial R_{p+1} in [-1,1]. On $[-1,1]$ the interpolation error can be written as

$$w(\xi) - \pi w(\xi) = \frac{h^{p+1} w^{(p+1)}(s(x(\xi),h))}{2^{p+1}(p+1)!}(\xi-\xi_0)(\xi-\xi_1)\dots(\xi-\xi_p), \ \xi \in [-1,1]. \tag{4.18}$$

The Maclaurin series of $w^{(p+1)}(s(x(\xi),h))$ with respect to h yields

$$w^{(p+1)}(s(x(\xi),h)) = w^{(p+1)}(0) + \sum_{k=1}^{\infty} h^k q_k(\xi) \tag{4.19a}$$

where

$$q_k(\xi) = \frac{1}{k!}\frac{d^k w^{(p+1)}(s(x(\xi),h))}{dh^k}|_{h=0}, \qquad k > 0, \tag{4.19b}$$

is a polynomial of degree k.

Combining (4.18) and (4.19) completes the proof. □

Now we are ready to state the main result of this section.

Theorem 4.1. *Let u and U be the solution of (4.2) and (4.3), respectively. Then the local finite element error can be written as*

$$\epsilon(\xi,\eta) = \sum_{k=p+1}^{\infty} h^k Q_k(\xi,\eta), \tag{4.20}$$

where

$$Q_{p+1} = \beta_1 R_{p+1}(\xi) + \beta_2 R_{p+1}(\eta). \tag{4.21}$$

Furthermore, at the outflow boundary of the physical element Δ

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\ \epsilon\ d\sigma = O(h^{2p+2}). \tag{4.22}$$

Proof. Writing the Maclaurin series of the local error ϵ with respect to h yields

$$\epsilon = \sum_{k=0}^{\infty} Q_k(\xi,\eta) h^k. \tag{4.23}$$

Substituting the series (4.23) in the DGM orthogonality condition (4.9), using (4.14) and collecting terms having the same powers of h we write

$$\sum_{k=0}^{p} h^k \left(\oint_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ Q_k V d\sigma - \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla V Q_k\ d\xi\ d\eta \right) +$$

$$\sum_{k=p+1}^{\infty} h^k \left(\oint_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ Q_k^- V\ d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ Q_k V\ d\sigma - \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla V Q_k\ d\xi\ d\eta \right) = 0,$$

$$\forall V \in \mathcal{V}_p. \tag{4.24}$$

Thus, for $0 \leq k \leq p$ we have

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ Q_k V\ d\sigma - \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla V Q_k\ d\xi\ d\eta = 0, \qquad \forall V \in \mathcal{V}_p. \tag{4.25}$$

By lemma 4.1, (4.25) leads to $Q_k = 0,\ k\ =\ 0,\ 1,\ \ldots,\ p$, and establishes (4.20).

On the other-hand, using Lemma 4.2, we write

$$Q_{p+1}^-(\xi,\eta) = \begin{cases} \frac{1}{2^{p+1}(p+1)!} \frac{\partial^{p+1} u}{\partial x^{p+1}}(0,0)(\xi-\xi_0)(\xi-\xi_1)\ldots(\xi-\xi_p), & \text{on } \Gamma_1 \\ \\ \frac{1}{2^{p+1}(p+1)!} \frac{\partial^{p+1} u}{\partial y^{p+1}}(0,0)(\eta-\xi_0)(\eta-\xi_1)\ldots(\eta-\xi_p), & \text{on } \Gamma_4 \end{cases}. \tag{4.26}$$

A direct computation reveals that Q_{p+1} can be split as

$$Q_{p+1} = \frac{1}{(p+1)!} \frac{d^{p+1}(u-U)}{dh^{p+1}}(\xi,\eta)|_{h=0} = \check{Q}_{p+1} + \tilde{Q}_p, \tag{4.27a}$$

where $\tilde{Q}_p(\xi,\eta) \in \mathcal{V}_p$ and

$$\check{Q}_{p+1} = \frac{1}{2^{p+1}(p+1)!} \frac{\partial^{p+1} u}{\partial x^{p+1}}(0,0)(\xi-\xi_0)(\xi-\xi_1)\ldots(\xi-\xi_p)$$

$$+ \frac{1}{2^{p+1}(p+1)!} \frac{\partial^{p+1} u}{\partial y^{p+1}}(0,0)(\eta-\xi_0)(\eta-\xi_1)\ldots(\eta-\xi_p). \tag{4.27b}$$

Substituting (4.27) in the $O(h^{p+1})$ term of the series (4.24) leads to

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ Q_{p+1}^- V\ d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ \check{Q}_{p+1} V\ d\sigma - \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla V \check{Q}_{p+1}\ d\xi\ d\eta$$

$$+ \oint_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n}\ \tilde{Q}_p V\ d\sigma - \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla V \tilde{Q}_p\ d\xi\ d\eta\ = 0, \quad \forall\ V \in \mathcal{V}_p. \tag{4.28}$$

Using (4.26) and (4.27b) we can show that

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\; Q_{p+1}^{-}V\; d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\; \check{Q}_{p+1}V\; d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla V\check{Q}_{p+1}\; d\xi\; d\eta \;=\; 0,$$

$$\forall\; V\in\mathcal{V}_p. \tag{4.29}$$

Combining (4.28) and (4.29) with Lemma 4.1 leads to $\tilde{Q}_p \;=\; 0$. Using (4.27) we establish (4.21).

Using (4.26) with Lemma 4.2 we can show that

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\; Q_{k}^{-}V\; d\sigma \;=\; 0,\;\; \forall\; V\in\mathcal{V}_{2p-k},\;\; k \;=\; p+1,\;\cdots,\;2p. \tag{4.30}$$

Using (4.30), the $O(h^k),\; p+1\le k\le 2p$, term of (4.24) yields

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_kV d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla VQ_k\; d\xi\; d\eta = 0,\;\; \forall\; V\in\mathcal{V}_{2p-k}. \tag{4.31}$$

Testing against $V=1$ we obtain

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\; Q_k\; d\sigma = 0,\;\; k \;=\; p+1,\;\cdots,\;2p, \tag{4.32}$$

which establishes (4.22). □

In the previous theorem we have extended the results of Adjerid *et al.* [5] to multi-dimensional problems on square meshes. In particular, we have proved that the $p-$degree DG solution is $O(h^{p+2})$ superconvergent at Radau points $(\xi_i,\xi_j),\; i,j=0,...,p$ and that the DG solution has an $O(h^{2p+2})$ superconvergence on average at the outflow boundary of Δ.

We note that for an arbitrary nonzero constant vector $\boldsymbol{\alpha}$, (4.20), (4.21) and (4.22) hold where the significant part of the error is spanned by the $p+1$-degree Radau polynomials

$$R_{p+1}(\xi) = L_{p+1}(\xi) - sign(\alpha_1)L_p(\xi),\;\; R_{p+1}(\eta) = L_{p+1}(\eta) - sign(\alpha_2)L_p(\eta). \tag{4.33}$$

4.2.1 Global Error Analysis

In the following theorem we will show a superconvergence result for the global discretization error if $p\ge 1$.

Theorem 4.2. *Under the conditions of Theorem 4.1 the global finite element error e for* $p\ge 1$ *satisfies the superconvergence result*

$$\oint_{\Omega_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\; e(x,y)\; d\sigma = O(h^{2p+1}). \tag{4.34}$$

Proof. Adding and subtracting $u(x,y)$ to both terms on the left side of (4.3) we obtain

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, V(U^- - u + u - U)\, d\sigma + \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla(U-u+u)\, V\, dxdy \;=\; \iint_{\Delta} f(x,y)V\, dxdy,$$

$$\forall\, V \in \mathcal{V}_p, \tag{4.35}$$

where Δ is an arbitrary element.

This can be written as

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, V(e - e^-)\, d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla e\, V\, dxdy \;=\; 0$$

$$\forall\, V \in \mathcal{V}_p. \tag{4.36}$$

Integrating by parts leads to

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, Ve^-\, d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, Ve\, d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla V\, e\, dxdy \;=\; 0$$

$$\forall\, V \in \mathcal{V}_p. \tag{4.37}$$

Testing against $V = 1$ yields

$$\oint_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, e^-\, d\sigma + \oint_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, e\, d\sigma = 0. \tag{4.38}$$

Summing over all elements we obtain

$$\oint_{\Omega_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, e^-\, d\sigma + \oint_{\Omega_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}\, e\, d\sigma = 0. \tag{4.39}$$

Combining (4.39) and Lemma 4.2 we establish (4.34). □

4.2.2 Nonlinear Conservation Laws

We will describe similar results for nonlinear conservation laws of the form

$$\nabla\cdot\boldsymbol{F}(\boldsymbol{u}) = H(x,y), \quad (x,y)\in\Omega = [0,1]^2 \tag{4.40}$$

with boundary conditions on the inflow boundaries and $\boldsymbol{F}(u) : \mathbb{R} \to \mathbb{R}^2$.

The DGM weak formulation consists of determining $U(x,y) \in \mathcal{V}_p$ on Δ such that

$$\oint_{\Gamma_{in}} \boldsymbol{n}\cdot[\boldsymbol{F}(U^-) - \boldsymbol{F}(U)]V d\sigma + \iint_{\Delta} \nabla\cdot\boldsymbol{F}(U)\, V dxdy = \iint_{\Delta} \boldsymbol{H}V dxdy,$$

$$\forall\ V \in \mathcal{V}_p. \tag{4.41}$$

We will establish a superconvergence result for the flux on the outflow boundary of the first element Γ_{out} and on the entire outflow boundary $\partial\Omega_{\text{out}}$, but first for completeness we provide a vector equivalent version of Lemma (4.2).

Lemma 4.3. *Given a vector function $\boldsymbol{W}(x) = (w_1, w_2, \ldots, w_m)^\top(x) \in C^\infty(0,h)$, let $\boldsymbol{\pi W}$ be a vector of $p-$degree polynomials such that πw_j interpolates w_j at Radau points on $[0,h]$. Then the interpolation error*

$$\boldsymbol{W}(x(\xi)) - \boldsymbol{\pi W}(x(\xi)) = \sum_{k=p+1}^{\infty} \boldsymbol{\Psi}_k^-(\xi) h^k, \tag{4.42}$$

where

$$\boldsymbol{\Psi}_{p+1}^-(\xi) = \frac{\boldsymbol{W}^{p+1}(0)}{2^{p+1}(p+1)!}(\xi-\xi_0)(\xi-\xi_1)\cdots(\xi-\xi_p) = \boldsymbol{c}_{p+1} R_{p+1}, \tag{4.43}$$

with $\boldsymbol{c}_{p+1}$ a constant vector and

$$\boldsymbol{\Psi}_k^-(\xi) = R_{p+1}^+(\xi)\ \boldsymbol{\tau}^{k-p-1}(\xi), \quad k > p+1, \tag{4.44}$$

with $\boldsymbol{\tau}^k(\xi)$ a vector polynomial of degree k.

Proof. The proof follows the same line of reasoning as in the proof of Lemma (4.2). □

Theorem 4.3. *Let u and U be the solution of (4.40) and (4.41) respectively. If u and F are smooth, then on the first element Δ the local error in the flux $\boldsymbol{\Upsilon}_\Delta(x,y) = \boldsymbol{F(u)} - \boldsymbol{F(U)}$, is such that*

$$\oint_{\Gamma_{out}} \boldsymbol{n} \cdot \boldsymbol{\Upsilon}_\Delta\ d\sigma = O(h^{2p+2}). \tag{4.45}$$

The global error in the flux $\boldsymbol{\Upsilon}(x,y)$, is such that

$$\oint_{\partial\Omega_{out}} \boldsymbol{n} \cdot \boldsymbol{\Upsilon}\ d\sigma = O(h^{2p+1}). \tag{4.46}$$

Proof. The DGM orthogonality condition for (4.40) on an element Δ mapped to $[-1,1] \times [-1,1]$ can be written as

$$\oint_{\Gamma_{in}} \boldsymbol{n} \cdot \boldsymbol{\Upsilon}_\Delta^- V d\sigma + \oint_{\Gamma_{out}} \boldsymbol{n} \cdot \boldsymbol{\Upsilon}_\Delta V d\sigma - \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{\Upsilon}_\Delta \cdot \nabla V\ d\xi d\eta = 0, \quad \forall\ V \in \mathcal{V}_p. \tag{4.47}$$

Writing the Maclaurin series of the local error in the flux $\boldsymbol{\Upsilon}_\Delta$ with respect to h yields

$$\boldsymbol{\Upsilon}_\Delta = \sum_{k=0}^{\infty} \boldsymbol{\Psi}_k(\xi,\eta) h^k, \tag{4.48}$$

where each element of $\mathbf{\Psi}_k$ is in $\mathcal{V}_k$.

Substituting (4.48) into the DGM orthogonality condition (4.47), using Lemma (4.3) on the inflow boundary, and collecting terms having the same powers of h we get

$$\sum_{k=0}^{p} h^k \left(\oint_{\Gamma_{out}} \boldsymbol{n} \cdot \mathbf{\Psi}_k V d\sigma - \int_{-1}^{1}\int_{-1}^{1} \mathbf{\Psi}_k \cdot \nabla V \; d\xi d\eta \right)$$

$$+ \sum_{k=p+1}^{\infty} h^k \left(\oint_{\Gamma_{in}} \boldsymbol{n} \cdot \mathbf{\Psi}_k^- V d\sigma + \oint_{\Gamma_{out}} \boldsymbol{n} \cdot \mathbf{\Psi}_k V d\sigma - \int_{-1}^{1}\int_{-1}^{1} \mathbf{\Psi}_k \cdot \nabla V \; d\xi d\eta \right) = 0,$$

$$\forall \; V \in \mathcal{V}_p \tag{4.49}$$

Letting $V = 1$ yields

$$\sum_{k=0}^{p} h^k \left(\oint_{\Gamma_{out}} \boldsymbol{n} \cdot \mathbf{\Psi}_k d\sigma \right) + \sum_{k=p+1}^{\infty} h^k \left(\oint_{\Gamma_{in}} \boldsymbol{n} \cdot \mathbf{\Psi}_k^- d\sigma + \oint_{\Gamma_{out}} \boldsymbol{n} \cdot \mathbf{\Psi}_k d\sigma \right) = 0. \tag{4.50}$$

Clearly for $k < p+1$

$$\oint_{\Gamma_{out}} \boldsymbol{n} \cdot \mathbf{\Psi}_k d\sigma = 0 \tag{4.51}$$

Recalling the form of $\mathbf{\Psi}^-$ and using the orthogonality of Legendre polynomials, we see that

$$\oint_{\Gamma_{in}} \boldsymbol{n} \cdot \mathbf{\Psi}_k^- d\sigma = 0, \quad k < 2p+1. \tag{4.52}$$

Using (4.51) and (4.52) in (4.50) we get

$$\oint_{\Gamma_{out}} \boldsymbol{n} \cdot \mathbf{\Psi}_k d\sigma = 0, \quad k < 2p+1. \tag{4.53}$$

which establishes (4.45).

Following the same line of reasoning as in the proof of Theorem (4.2), we sum over all elements to obtain

$$\oint_{\Gamma_{\text{in}}} \boldsymbol{n} \cdot \mathbf{\Upsilon}^- \; d\sigma + \oint_{\Gamma_{\text{out}}} \boldsymbol{n} \cdot \mathbf{\Upsilon} \; d\sigma = 0. \tag{4.54}$$

Combining (4.54) and Lemma (4.3) we establish (4.46). □

We have seen computational evidence suggesting that the leading term in the true local error for these nonlinear conservation laws is spanned by two $(p+1)$-degree Radau polynomials in the x and the y directions respectively. A proof for this in under investigation.

4.2.3 Hyperbolic Problems with a Nonlinear Reaction Term

We will describe similar results for problems of the form

$$\boldsymbol{\alpha} \cdot \nabla u \ + \ \phi(u) = f(x,y), \qquad (x,y) \in \Omega, \tag{4.55}$$

with boundary conditions at the inflow boundary.

The DGM weak formulation consists of determining $U(x,y) \in \mathcal{V}_p$ on Δ such that

$$\int_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n} V(U^- - U) d\sigma + \iint_{\Delta} [\boldsymbol{\alpha} \cdot \nabla U V + \phi(U) V] dx dy \ = \ \iint_{\Delta} f(x,y) V dx dy,$$

$$\forall \ V \in \mathcal{V}_p. \tag{4.56}$$

In the following theorem we state superconvergence results and local error estimates for hyperbolic problems with a nonlinear reaction term.

Theorem 4.4. *Let u and U be the solution of (4.55) and (4.56), respectively. If u is smooth, then the local error estimates (4.20) and (4.21) hold, and*

$$\oint_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n} \ \epsilon \ d\sigma = O(h^{\min(p+4, 2p+2)}). \tag{4.57}$$

Proof. The DGM orthogonality for (4.55) on an element Δ can be written as

$$\int_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n} \epsilon^- V d\sigma + \int_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n} \ \epsilon V d\sigma - \int_0^h \int_0^h [\boldsymbol{\alpha} \cdot \nabla V \epsilon + (\phi(u) - \phi(U)) V] dx \ dy = 0,$$

$$\forall V \in \mathcal{V}_p. \tag{4.58}$$

On $[-1,1] \times [-1,1]$ the orthogonality condition (4.58) becomes

$$\int_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n} \epsilon^- V d\sigma + \int_{\Gamma_{out}} \boldsymbol{\alpha} \cdot \boldsymbol{n} \ \epsilon V d\sigma - \int_{-1}^1 \int_{-1}^1 [\boldsymbol{\alpha} \cdot \nabla V \epsilon + \frac{h}{2} (\phi(u) - \phi(U)) V] d\xi \ d\eta = 0,$$

$$\forall V \in \mathcal{V}_p. \tag{4.59}$$

Assuming ϕ analytic and ϕ'' bounded and using Taylor series of ϕ about u we have

$$\phi(u) - \phi(U) = a(u)\epsilon - \frac{\epsilon^2}{2} \phi''(\bar{u}), \quad a(u) = \phi'(u). \tag{4.60a}$$

The Maclaurin series of $a(u)$ with respect to h yields

$$a(u) = 2 \sum_{k=0}^{\infty} h^k \bar{Q}_k(\xi, \eta), \quad \bar{Q}_k(\xi, \eta) = \frac{1}{2} \frac{\phi^{(k+1)}(u(x(\xi), y(\eta)))}{k!} \frac{d^k u}{dh^k}(x(\xi), y(\eta))|_{h=0} \in \mathcal{P}_k. \tag{4.60b}$$

We note that in order to establish (4.60b) we used the chain rule with $x(\xi) = h(1+\xi)/2$ and $y(\eta) = h(1+\eta)/2$.

Substituting (4.23) and (4.60) in (4.59) and collecting terms having same powers of h lead to

$$\int_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_0 V d\sigma - \iint_{\Delta} \boldsymbol{\alpha}\cdot\nabla V Q_0 d\xi\ d\eta +$$

$$\sum_{k=1}^{p} h^k \left(\int_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k V d\sigma - \iint_{\Delta} [\boldsymbol{\alpha}\cdot\nabla V Q_k + Z_{k-1}V] d\xi\ d\eta \right) +$$

$$\sum_{k=p+1}^{\infty} h^k \Big(\int_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k^- V d\sigma + \int_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k V d\sigma -$$

$$\iint_{\Delta} [\boldsymbol{\alpha}\cdot\nabla V Q_k + Z_{k-1}V] d\xi\ d\eta \Big) = 0, \qquad \forall V \in \mathcal{V}_p, \tag{4.61a}$$

where

$$Z_k = \sum_{l=0}^{k} \bar{Q}_l Q_{k-l}. \tag{4.61b}$$

Applying Lemma 4.2, the $O(1)$ term yields $Q_0 = 0$. By induction we prove that $Q_k = 0,\ k = 0, 1, \ldots, p$. We note that the term in (4.60a) involving ϵ^2 is higher order and does not contribute to our leading terms.

Following the same line of reasoning used to prove (4.21) we establish the same result for nonlinear problems.

We prove the strong superconvergence (4.57) for nonlinear problems by first using (4.21) to obtain

$$\int_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_{p+1} d\sigma = 0. \tag{4.62}$$

The $O(h^k)$ term in (4.61a) with $k > p+1$ with $V = 1$ leads to

$$\int_{\Gamma_{in}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k^- d\sigma + \int_{\Gamma_{out}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_k d\sigma - \iint_{\Delta} Z_{k-1} d\xi\ d\eta = 0. \tag{4.63}$$

Using

$$Z_m = \begin{cases} 0,\ if\ m \le p \\ \\ \sum_{l=0}^{m-p-1} \bar{Q}_l Q_{m-l},\ otherwise \end{cases}, \tag{4.64}$$

with $k = p+2$ in (4.63) leads to

$$\oint_{\Gamma_{\text{out}}} \boldsymbol{\alpha}\cdot\boldsymbol{n}Q_{p+2} d\sigma \ = \ 0. \tag{4.65}$$

When $k = p + 3$ in (4.63) we have

$$\oint_{\Gamma_{\text{out}}} \boldsymbol{\alpha} \cdot \boldsymbol{n} Q_{p+3} d\sigma \; = \iint_{\Delta} \bar{Q}_0 Q_{p+2} \; d\xi \; d\eta, \tag{4.66}$$

which is not necessarily zero. This leads to (4.57). □

4.3 *A Posteriori* Error Estimation

The results of Theorem 4.1 and (4.2) suggest that the global finite element error on each element Δ can be approximated as

$$e(x, y) \; = \; u - U \; \approx E(x, y) = \; b_1 R_{p+1}(x) + b_2 R_{p+1}(y). \tag{4.67}$$

Substituting u in (4.3) by $e + U$ leads to

$$\iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla (U + e) W dxdy = \iint_{\Delta} f W dxdy. \tag{4.68}$$

Approximating e by E yields the following discrete problem for the error

$$\iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla (U + E) R_{p+1}(x) dxdy = \iint_{\Delta} f R_{p+1}(x) dxdy, \tag{4.69a}$$

$$\iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla (U + E) R_{p+1}(y) dxdy = \iint_{\Delta} f R_{p+1}(y) dxdy. \tag{4.69b}$$

We note that the strong superconvergence (4.22) of p-degree, $p > 0$, finite element solution flux at the outflow boundary of each element suggests that we should neglect the jump terms in the error problem without compromising the accuracy of our error estimate. Since for $p = 0$ this strong superconvergence at the outflow boundary is lost and (4.69) fails to have a unique solution, we solve the following problem for the error estimate

$$\int_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n} (U^- + E^- - U - E) R_1(x) d\sigma \; + \; \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla (U + E) R_1(x) dxdy$$

$$= \iint_{\Delta} f R_1(x) dxdy, \tag{4.70a}$$

$$\int_{\Gamma_{in}} \boldsymbol{\alpha} \cdot \boldsymbol{n} (U^- + E^- - U - E) R_1(y) d\sigma \; + \; \iint_{\Delta} \boldsymbol{\alpha} \cdot \nabla (U + E) R_1(y) dxdy$$

$$= \iint_{\Delta} f R_1(y) dxdy. \tag{4.70b}$$

For nonlinear problems of the form

$$f_x(u) + g_y(u) = h(x, y) \tag{4.71}$$

we find E by solving the linearized problem

$$\iint_{\Delta} [f'(U), g'(U)] \cdot \nabla(U + E) R_{p+1}(x) dxdy = \iint_{\Delta} h R_{p+1}(x) dxdy, \tag{4.72a}$$

$$\iint_{\Delta} [f'(U), g'(U)] \cdot \nabla(U + E) R_{p+1}(y) dxdy = \iint_{\Delta} h R_{p+1}(y) dxdy. \tag{4.72b}$$

We also use the solution of the linearized problem (4.72) as an initial guess for Newton iteration when solving the nonlinear finite element problem for E

$$\iint_{\Delta} [f'(U + E), g'(U + E)] \cdot \nabla(U + E) R_{p+1}(x) dxdy = \iint_{\Delta} h R_{p+1}(x) dxdy, \tag{4.73a}$$

$$\iint_{\Delta} [f'(U + E), g'(U + E)] \cdot \nabla(U + E) R_{p+1}(y) dxdy = \iint_{\Delta} h R_{p+1}(y) dxdy. \tag{4.73b}$$

An accepted efficiency measure of *a posteriori* error estimates is the effectivity index. In this dissertation we use the local effectivity indices in the $\mathcal{L}^2$ norm

$$\theta_i \;=\; \frac{||E||_{\mathcal{L}^2(\Delta_i)}}{||e||_{\mathcal{L}^2(\Delta_i)}}, \qquad i = 1,\ 2,\ \ldots,\ N \tag{4.74}$$

and the global effectivity index

$$\theta \;=\; \frac{||E||_{\mathcal{L}^2(\Omega)}}{||e||_{\mathcal{L}^2(\Omega)}}. \tag{4.75}$$

Ideally, effectivity indices should approach unity under mesh refinement.

We perform several tests to study the effect of boundary conditions on the quality of our *a posteriori* error estimates, show the superconvergence points of the discontinuous Galerkin solution, and show the superconvergence of the flux in the next chapter.

Chapter 5

Examples for the DGM *A Posteriori* Error Estimate

5.1 Linear Examples

Example 1. We consider a linear hyperbolic problem

$$u_x + 2u_y = f(x,y), \qquad (x,y) \in [0,1]^2 \tag{5.1a}$$

subject to the boundary conditions

$$u(x,0) = g_1(x), \quad u(0,y) = g_2(y). \tag{5.1b}$$

We select $f(x,y)$, g_1 and g_2 such that the exact solution is

$$u(x,y) = e^{x+y}. \tag{5.1c}$$

We perform several tests on this example to study the effect of boundary conditions on the quality of our a posteriori error estimates and show the superconvergence points of the discontinuous Galerkin solution. We also show the superconvergence in the flux of the error on the outflow boundaries.

We start by solving (5.1) on a uniform mesh having 64 elements with $p = 1, 2, 3$ with U^- being the true boundary conditions. We repeat the previous experiment with U^- being the interpolant of true boundary conditions at Radau points as defined in (2.40) and show the true error in each case in Figure 5.1. We present the local effectivity indices in Figure 5.3. The effectivity indices corresponding to the exact boundary conditions are farther from unity than those corresponding to interpolated boundary conditions especially on elements near the inflow boundaries. In both experiments the effectivity indices converge to one under p-refinement.

We solve (5.1) using U^- defined in (2.40) on a 16-element uniform mesh with p ranging from 1 to 6 and show the zero-level curves of the true error on each element in Figure 5.2. As predicted by the theory of §4.2, these results indicate that the $p-$degree discontinuous Galerkin solution is superconvergent at Radau points, shown by 'x', on each element for $p \geq 1$.

Since there is no superconvergence for global errors when $p = 0$, the error estimation procedures (4.69), (4.72) and (4.73) do not apply. This is illustrated by solving (5.1) and (4.70) using approximate boundary conditions on a 100-element mesh with $p = 0$ and presenting the local effectivity indices in Figure 5.4. In this case we observe that the local effectivity indices get larger than unity away from the inflow boundary elements. Even though there is no superconvergence, we observe that the rate of convergence for $||e||_{\mathcal{L}_2(\Omega)}$ with the DG method for $p = 0$ is still order $O(h^{p+1})$.

As a final test, we solve (5.1) on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements with $p = 1, 2, 3, 4$, using the true boundary conditions. We present the $\mathcal{L}_2(\Omega)$ norm of the true errors in Table 5.3 and the global effectivity indices in Table 5.1. We repeat the previous experiment using approximate boundary conditions (2.40) with all other parameters left unchanged, except we add the case $p = 0$, and present the errors and global effectivity indices in Tables 5.4 and 5.2, respectively. For the sake of illustration we include Figure 5.5 which demonstrates the convergence to one of the global effectivity indices for both the true and approximated boundary conditions. As noted earlier since there is no superconvergence for global errors when $p = 0$ we do not include results for $p = 0$ in Table 5.2. The computational results indicate that the error estimates obtained using the procedure (4.69) converge to the true error under both $h-$ and $p-$ refinements. This is the first a posteriori finite element error estimate that exhibits convergence under $h-$ and $p-$ refinement for multi-dimensional problems. Since the effect of the true versus approximate boundary conditions on the effectivity indices is negligible we shall use the approximate boundary conditions in the remainder of this section.

Finally, we present numerical evidence of the superconvergence results about the flux of the error on the outflow edges of the first element in Table 5.5, the last element in Table 5.6, and the maximum over all elements in Table 5.7. Figure 5.7 gives the same results in graphical format. In each plot there is a table of values showing the degree p and the computed rate of convergence m. As we can see the predicted rate of convergence $O(h^{2p+2})$ for the first element is obtained. In Table 5.8 and Figure 5.8 we present the results for the superconvergence of the flux of the error on the entire outflow boundary and observe that numerically we obtain nearly $O(h^{2p+1})$. In Tables 5.9, 5.10 and 5.11 and in Figure 5.9 we present the results for the superconvergence of the error on the first element, the last element and the maximum over all elements respectively. Again in each plot we have a table of p values and their corresponding computed rate of convergence m. We observe that for the first element, we obtain the predicted $O(h^{p+4})$. Furthermore, the results also indicate that the theory possibly extends to all elements, not just the first.

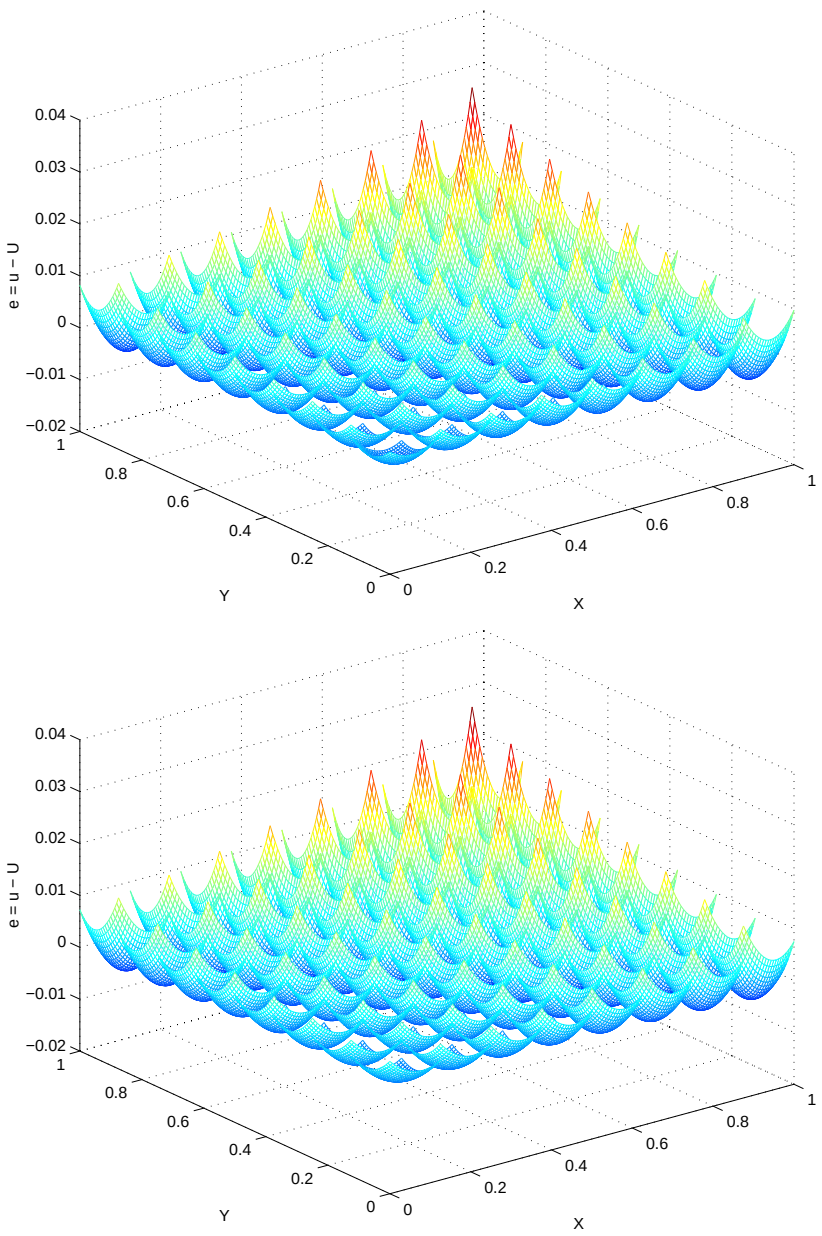


Figure 5.1: Surface plot of the error for *Example 1* on a 64 element uniform mesh with $p = 1$ and using true (top) and approximate (bottom) boundary conditions.

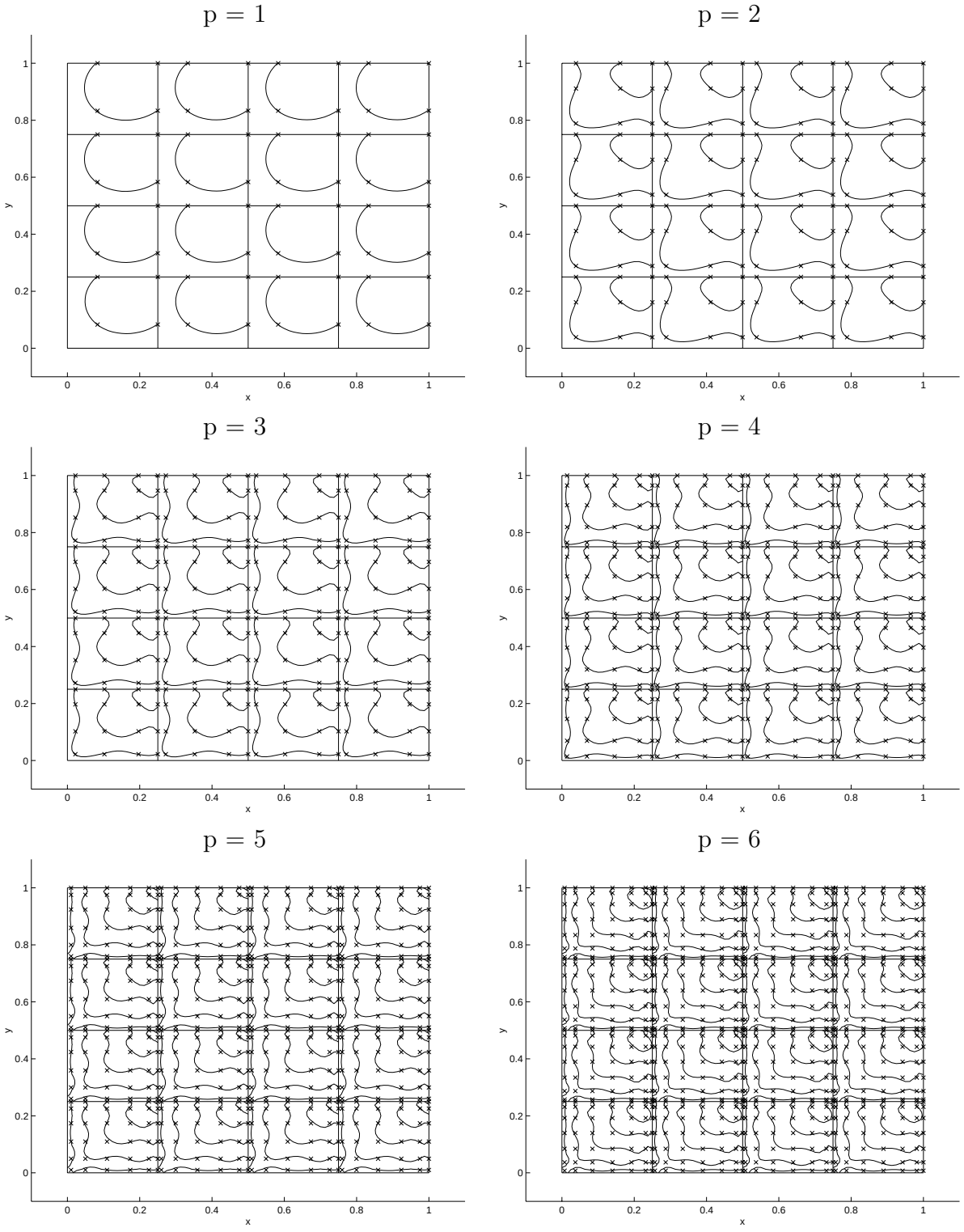


Figure 5.2: Zero-level curves of the contour plot of the DG discretization error for *Example 1* on a 16-element uniform mesh and $p = 1$ to 6 (upper left to lower right) with approximate boundary conditions. Radau points are shown with an '`x`'.

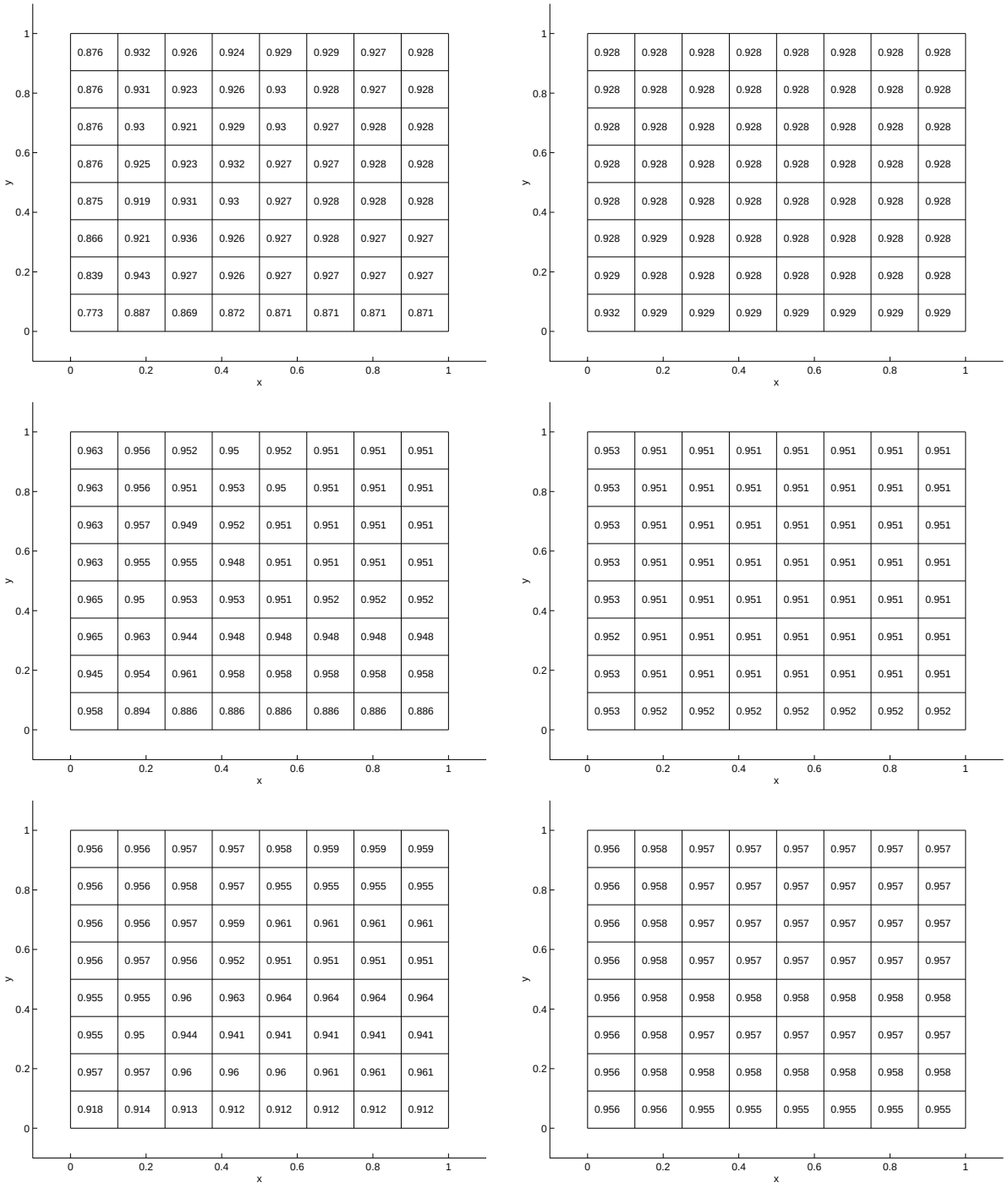


Figure 5.3: Local effectivity index values for *Example 1* on a 64 element uniform mesh for $p = 1$ to 3 from (top to bottom) with true (left) and approximate (right) boundary conditions.

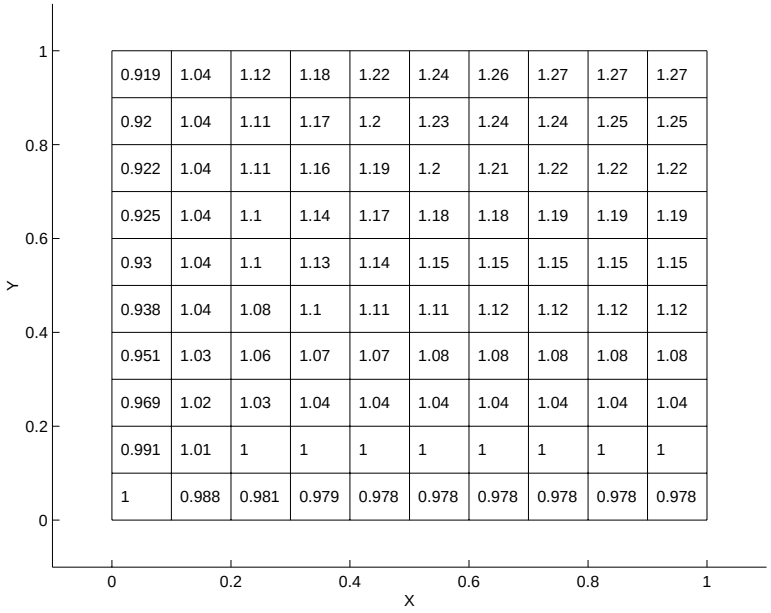


Figure 5.4: Local effectivity indices for *Example 1* on a 100-element uniform mesh for $p = 0$ and using approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	0.8760	0.9188	0.9240	0.9236
100	0.9388	0.9597	0.9641	0.9659
225	0.9594	0.9732	0.9764	0.9781
400	0.9696	0.9800	0.9825	0.9839
625	0.9757	0.9840	0.9861	0.9872
900	0.9798	0.9867	0.9885	0.9894

Table 5.1: Effectivity indices for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with true boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	0.8845	0.9211	0.9285	0.9257
100	0.9423	0.9610	0.9664	0.9676
225	0.9616	0.9741	0.9782	0.9795
400	0.9712	0.9807	0.9838	0.9851
625	0.9770	0.9846	0.9872	0.9883
900	0.9808	0.9871	0.9894	0.9903

Table 5.2: Effectivity indices for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4	p = 5
25	1.8684e-2	3.5066e-4	4.8726e-6	5.3577e-08	4.9797e-10
100	4.7156e-3	4.3948e-5	3.0338e-7	1.6526e-09	7.4672e-12
225	2.1027e-3	1.3035e-5	5.9883e-8	2.1708e-10	6.4986e-13
400	1.1847e-3	5.5019e-6	1.8943e-8	5.1475e-11	1.1707e-13
625	7.5894e-4	2.8179e-6	7.7583e-9	1.6860e-11	3.9903e-14
900	5.2738e-4	1.6311e-6	3.7413e-9	6.7745e-12	2.1435e-14

Table 5.3: $||e||_{\mathcal{L}_2(\Omega)}$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 5 with true boundary conditions.

N	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
25	5.5075e-1	1.8793e-2	3.5112e-4	4.8676e-6	5.3588e-08	4.9826e-10
100	2.7099e-1	4.7299e-3	4.3971e-5	3.0306e-7	1.6514e-09	7.4686e-12
225	1.7965e-1	2.1069e-3	1.3039e-5	5.9836e-8	2.1693e-10	6.4990e-13
400	1.3436e-1	1.1865e-3	5.5031e-6	1.8931e-8	5.1436e-11	1.1706e-13
625	1.0731e-1	7.5987e-4	2.8184e-6	7.7542e-9	1.6849e-11	3.9878e-14
900	8.9323e-2	5.2793e-4	1.6313e-6	3.7396e-9	6.7701e-12	2.1041e-14

Table 5.4: $||e||_{\mathcal{L}_\epsilon(\Omega)}$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 0$ to 5 with approximate boundary conditions.

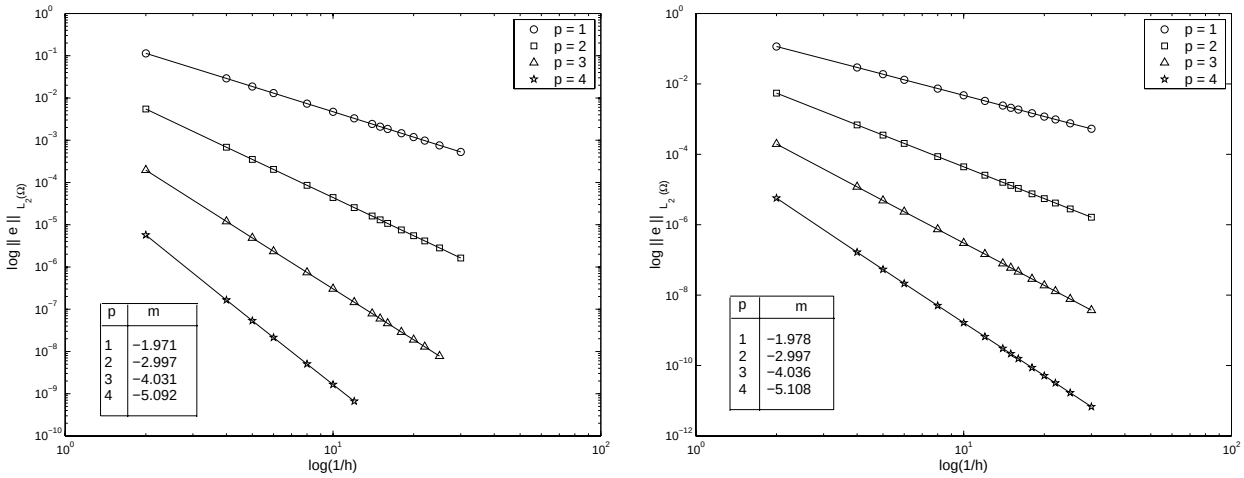


Figure 5.5: Rates of convergence under h-refinement for $||e||_{\mathcal{L}_2(\Omega)}$ for *Example 1* with $p = 1$ to 4 using true (left) and approximate (right) boundary conditions.

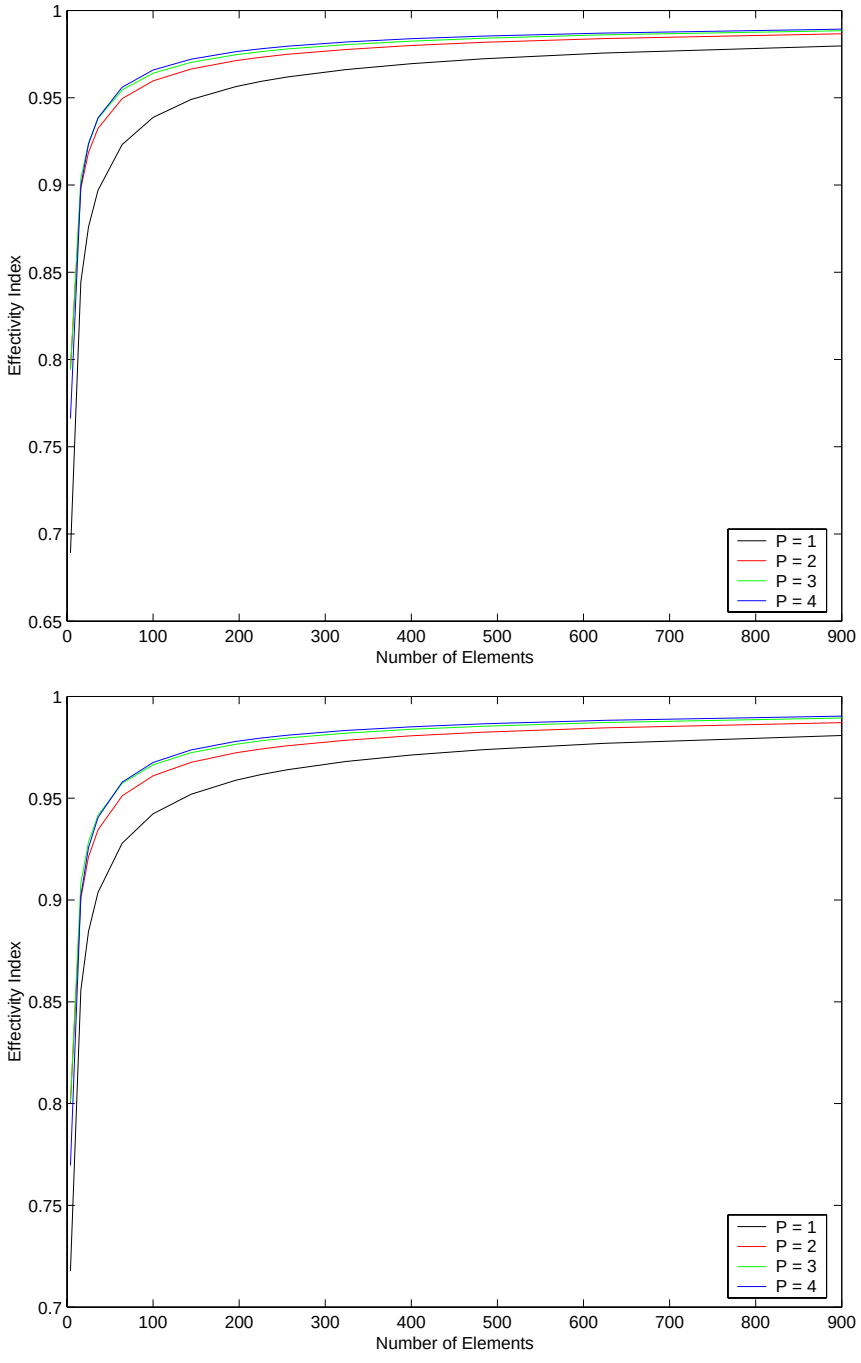


Figure 5.6: Effectivity index under h-refinement for *Example 1* with $p = 1$ to 4 and using true (top) and approximate (bottom) boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	2.4737e-005	2.9566e-009	1.7216e-013	3.8405e-016
100	1.4652e-006	4.3867e-011	5.8573e-016	1.4663e-016
225	2.8430e-007	3.7852e-012	1.1588e-016	5.0192e-017
400	8.9154e-008	6.6786e-013	1.9296e-017	1.0420e-016
625	3.6323e-008	1.7414e-013	1.1306e-017	4.5455e-017
900	1.7455e-008	5.8116e-014	5.9773e-017	2.8427e-017

Table 5.5: $|\int_{\Gamma^1_{\text{out}}} \alpha \cdot \nu \; e \; d\sigma|$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	2.0839e-005	1.2615e-008	9.6208e-011	1.1847e-011
100	1.1992e-006	2.4277e-010	3.2310e-014	3.9044e-014
225	2.3234e-007	2.1034e-011	4.6721e-016	6.5694e-015
400	7.3044e-008	3.7419e-012	5.2954e-016	3.5803e-015
625	2.9793e-008	9.8218e-013	9.9504e-016	3.2600e-015
900	1.4328e-008	3.2985e-013	1.2414e-015	3.5633e-015

Table 5.6: $|\int_{\Gamma^N_{\text{out}}} \alpha \cdot \nu \; e \; d\sigma|$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	1.7452e-004	2.9964e-007	9.3902e-009	1.1456e-010
100	1.2828e-005	1.1703e-008	1.6468e-010	9.9296e-013
225	2.6808e-006	1.6726e-009	1.5029e-011	6.0498e-014
400	8.7264e-007	4.1361e-010	2.7275e-012	8.4319e-015
625	3.6360e-007	1.3893e-010	7.2355e-013	3.6549e-015
900	1.7736e-007	5.6767e-011	2.4420e-013	3.5633e-015

Table 5.7: $\max_{\Delta} |\int_{\Gamma_{\text{out}}} \alpha \cdot \nu \; e \; d\sigma|$ for *Example 1* on uniform meshes having $25, 100, 225, 400,$ 625 and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

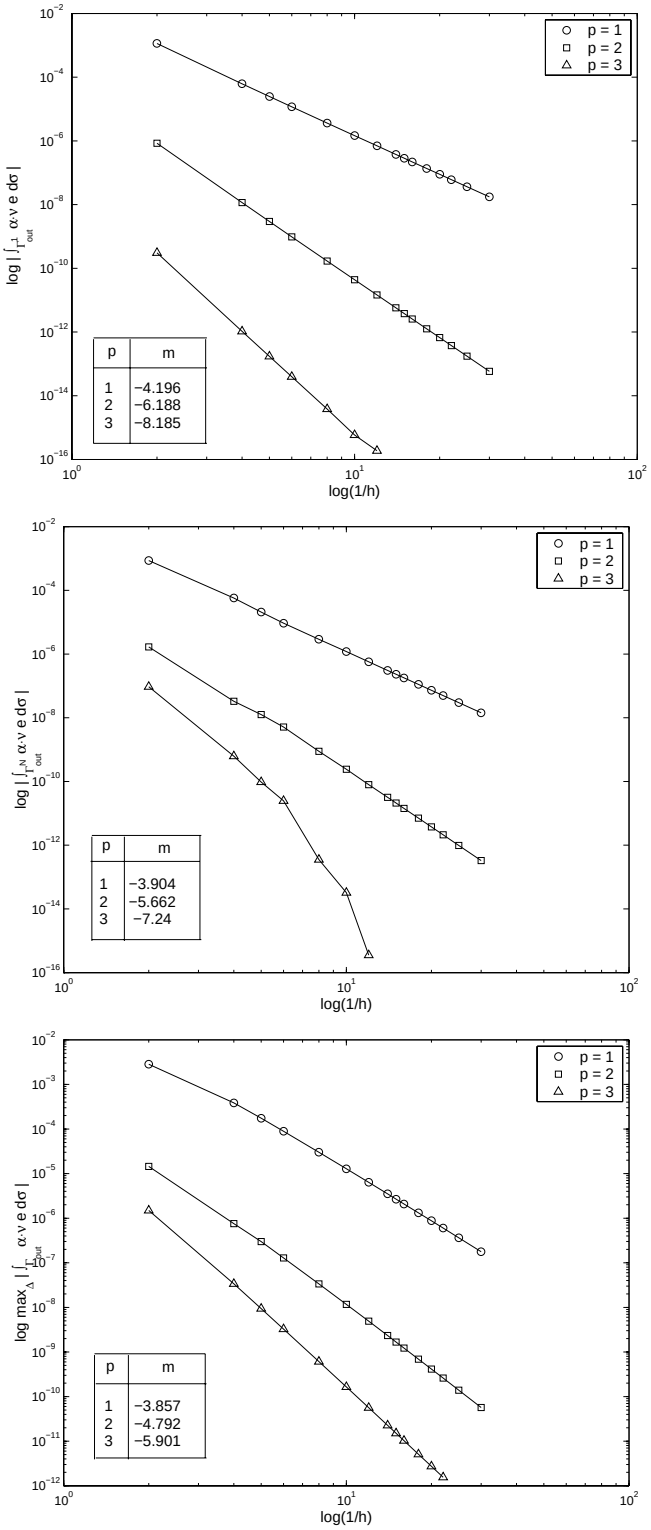


Figure 5.7: Rates of convergence under h-refinement for $|\int_{\Gamma_{\text{out}}} \alpha \cdot \nu \; e \; d\sigma|$ for *Example 1* using approximate boundary conditions with $p = 1$ to 3 for element 1, element N and the maximum over all elements (top to bottom).

N	p = 1	p = 2	p = 3	p = 4
25	1.9198e-004	2.2946e-008	1.3410e-012	2.2583e-014
100	2.3938e-005	7.1669e-010	1.1551e-014	3.0032e-014
225	7.0860e-006	9.4352e-011	5.5258e-015	4.0448e-014
400	2.9879e-006	2.2381e-011	1.6801e-015	4.5484e-014
625	1.5293e-006	7.3307e-012	1.5109e-014	4.9683e-014
900	8.8484e-007	2.9602e-012	6.8863e-015	6.2033e-014

Table 5.8: $|\int_{\partial\Omega_{\text{out}}} \alpha \cdot \nu \; e \; d\sigma|$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 using approximate boundary conditions.

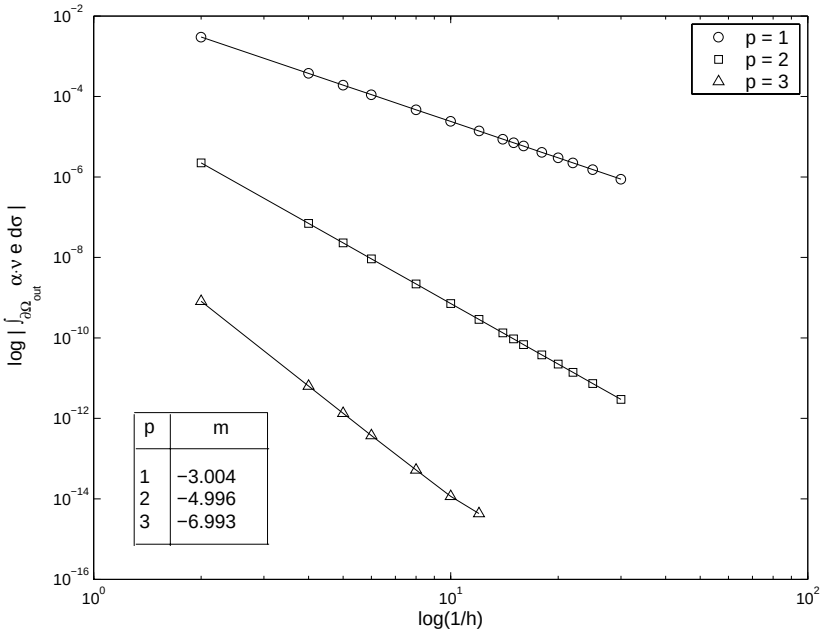


Figure 5.8: Rates of convergence under h-refinement for $|\int_{\partial\Omega_{\text{out}}} \alpha \cdot \nu \; e \; d\sigma|$ for *Example 1* with $p = 1$ to 3 using approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	5.3859e-006	7.0047e-009	5.5501e-011	9.4595e-013
100	1.5547e-007	9.5587e-011	3.8873e-013	3.3299e-015
225	1.9943e-008	8.0153e-012	2.1942e-014	1.2963e-016
400	4.6713e-009	1.3940e-012	2.8770e-015	1.5282e-017
625	1.5188e-009	3.6037e-013	5.9692e-016	3.6125e-018
900	6.0718e-010	1.1957e-013	1.6613e-016	1.5384e-018

Table 5.9: $|\iint_{\Delta_1} e\ dxdy|$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	2.4728e-005	1.5730e-008	9.6274e-012	1.3498e-013
100	8.7460e-007	1.4225e-010	1.6510e-014	1.9302e-016
225	1.2000e-007	8.6812e-012	4.6135e-016	1.1222e-016
400	2.9060e-008	1.1847e-012	3.2268e-017	6.6842e-017
625	9.6389e-009	2.5175e-013	8.1817e-018	4.1173e-017
900	3.9052e-009	7.0887e-014	9.1780e-018	3.7609e-017

Table 5.10: $|\iint_{\Delta_N} e\ dxdy|$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	2.4728e-005	2.2007e-008	5.5501e-011	9.4595e-013
100	8.7460e-007	2.6544e-010	3.8873e-013	3.3299e-015
225	1.2000e-007	2.0578e-011	2.1942e-014	1.2963e-016
400	2.9060e-008	3.5361e-012	2.8770e-015	7.1193e-017
625	9.6389e-009	9.0759e-013	5.9692e-016	4.9662e-017
900	3.9052e-009	2.9969e-013	1.6613e-016	3.9770e-017

Table 5.11: $\max_{\Delta} |\iint_{\Delta} e\ dxdy|$ for *Example 1* on uniform meshes having $25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4 with approximate boundary conditions.

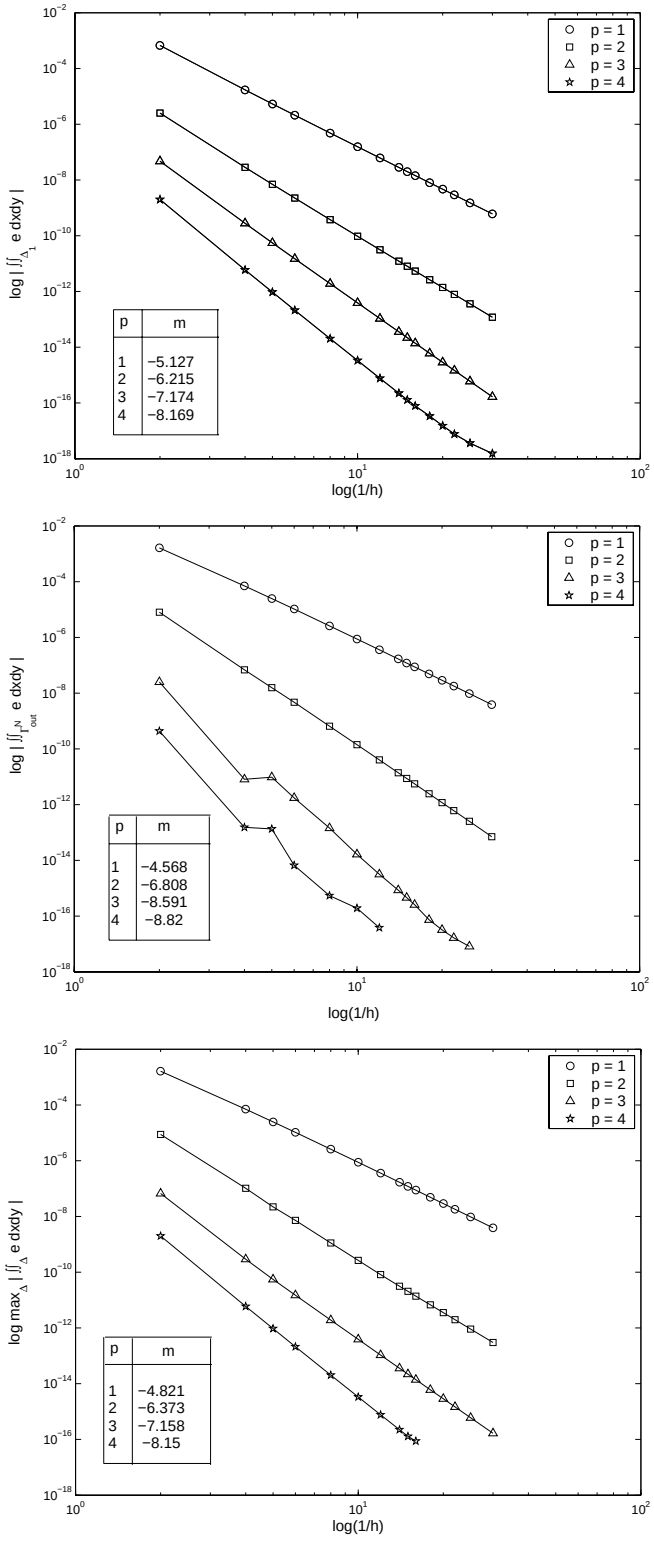


Figure 5.9: Rates of convergence under h-refinement for $|\iint_{\Delta} e\ dxdy|$ for *Example 1* with $p = 1$ to 4 using approximate boundary conditions for element 1, element N and the maximum over all elements (top to bottom).

Example 2. We solve (5.1) on the quadrilateral domain $\Omega = ABCD$ where $A = (0, 0)$, $B = (1, 0.1)$, $C = (1, 1)$ and $D = (1, 1.1)$, on a 64-element mesh for $p = 1$ and 2 and show surface plots of the true error in Figure 5.10. Notice the usual regularity of the pattern in the error for the DGM. We solve the problem again on a 16-element mesh for $p = 1$ to 4 and show the zero-levels of the contour plot of the error on each element in Figure 5.11. The computational results suggest that the DGM solution is superconvergent at the Radau points on more general meshes. We solve the same problem on a 25-element mesh for $p = 1$ to 4 using approximate boundary conditions and show the local effectivity indices in Figure 5.12 which indicates that under p-refinement the local effectivity indices are converging to unity.

To show the convergence of our *a posteriori* error estimates under mesh refinement, we solve the previous problem on meshes having $N = 25, 100, 225, 400, 625, 900$ elements and $p = 1$ to 6. We present the $\mathcal{L}_2(\Omega)$ norm of the true errors in Table 5.12 which indicate that the DGM approximation converges to the true solution on the order of $O(h^{p+1})$. The global effectivity indices are presented in Table 5.13 which indicate that the error estimates converge to the true errors under both $h-$ and $p-$ refinements for quadrilateral meshes. We note that for $p = 5$ and 6 the errors are on the order of machine roundoff and consequently we do not present the effectivity indices for these p.

N	p = 1	p = 2	p = 3	p = 4	p = 5	p = 6
25	1.8489e-2	3.4419e-4	4.8370e-6	5.4669e-08	5.1744e-10	4.2609e-12
100	4.6141e-3	4.2897e-5	3.0143e-7	1.7030e-09	8.0615e-12	4.4075e-14
225	2.0491e-3	1.2694e-5	5.9466e-8	2.2397e-10	7.0651e-13	2.0559e-14
400	1.1521e-3	5.3518e-6	1.8802e-8	5.3112e-11	1.2743e-13	3.2269e-14
625	7.3714e-4	2.7389e-6	7.6981e-9	1.7396e-11	4.6222e-14	3.3476e-14
900	5.1181e-4	1.5846e-6	3.7113e-9	6.9895e-12	3.3136e-14	3.6991e-14

Table 5.12: $\mathcal{L}_2$ norm of the error under h-refinement for *Example 2* with $p = 1$ to 6 using approximate boundary conditions.

N	p = 1	p = 2	p = 3	p = 4
25	0.9416	0.9770	0.9901	0.9940
100	0.9707	0.9885	0.9951	0.9974
225	0.9804	0.9924	0.9968	0.9983
400	0.9853	0.9943	0.9976	0.9988
625	0.9882	0.9954	0.9981	0.9990
900	0.9902	0.9962	0.9984	0.9992

Table 5.13: Global effectivity indices for *Example 2* on uniform meshes having $N = 25, 100, 225, 400, 625$ and 900 elements and $p = 1$ to 4.

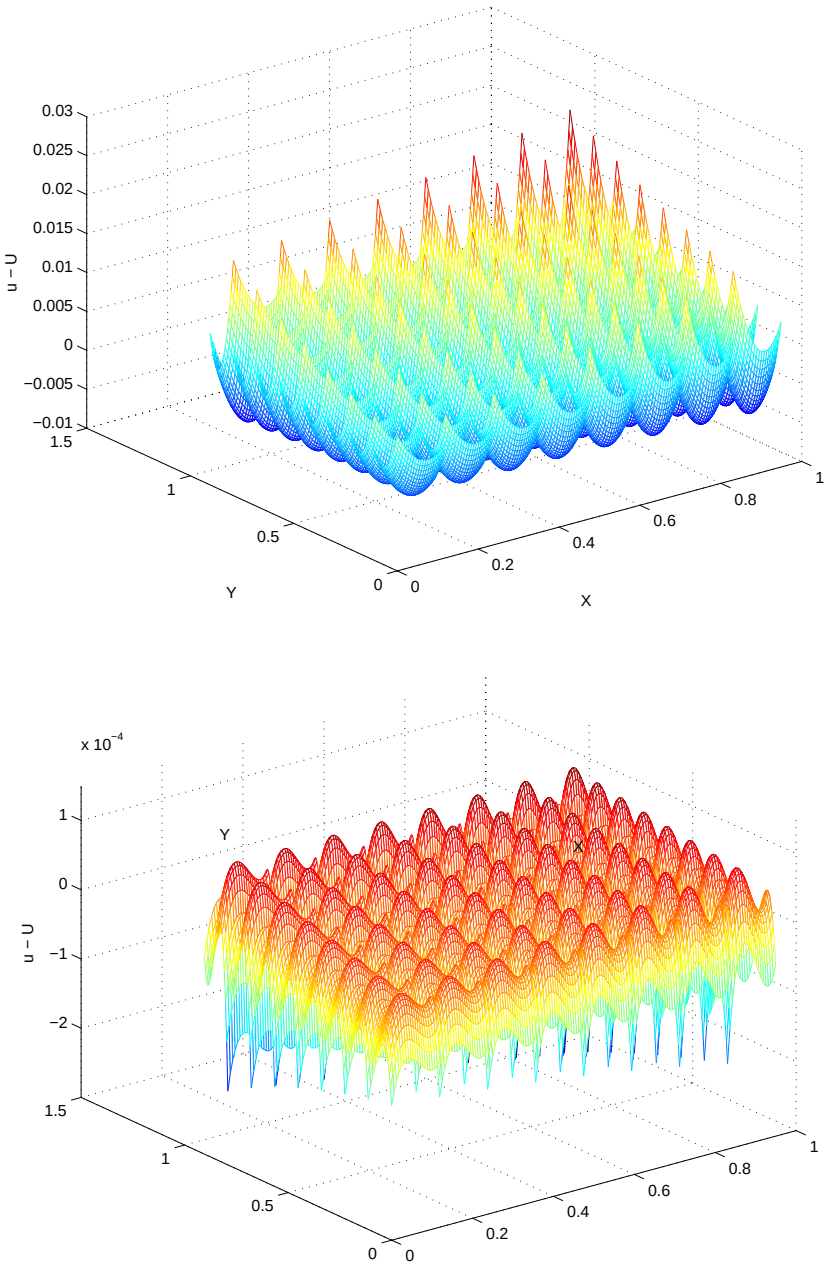


Figure 5.10: Surface plots of the error for *Example 2* with $p = 1$ and 2 (top to bottom).

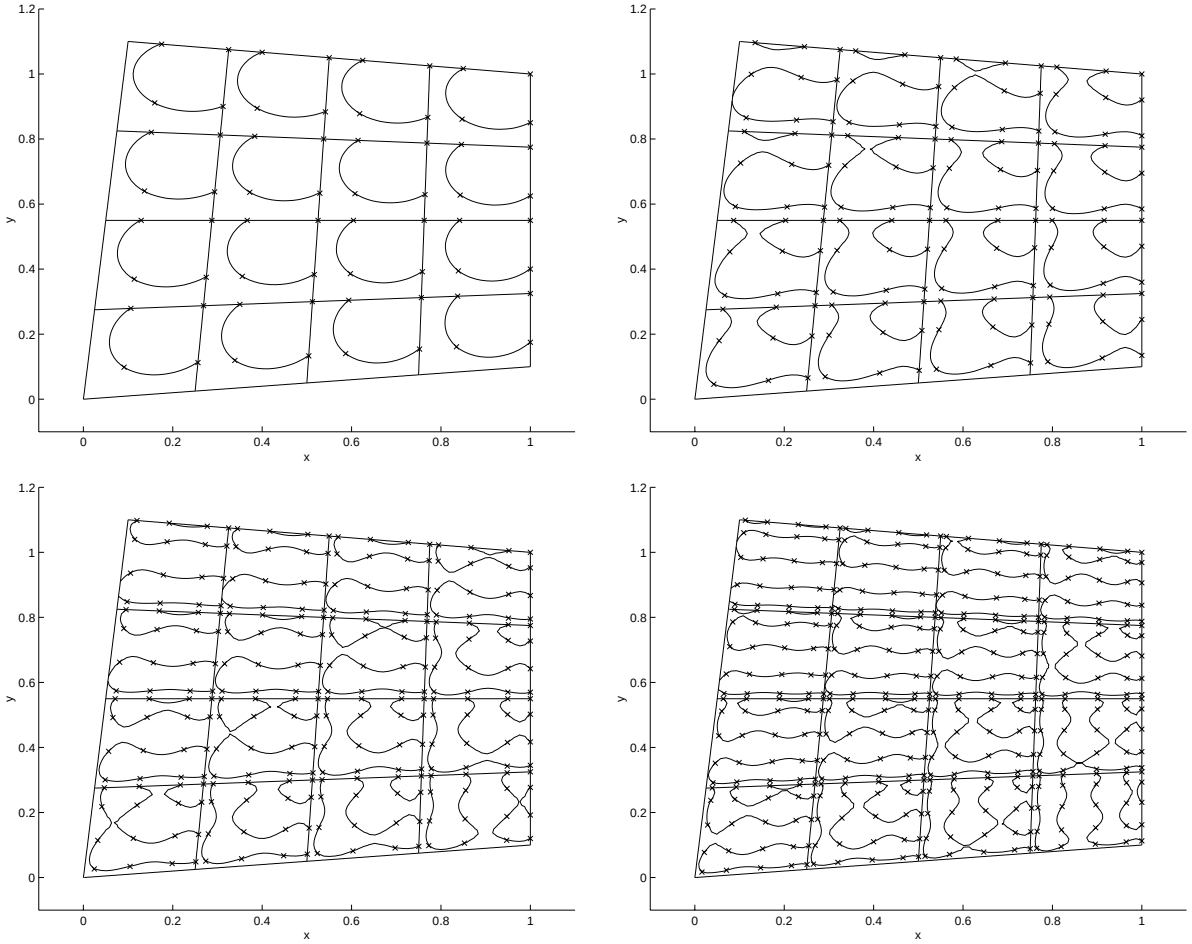


Figure 5.11: Zero-level curves of the contour plot of the DG discretization error for *Example 2* on a 16-element uniform mesh and $p = 1$ to 4 (upper left to lower right) with approximate boundary conditions. Radau points are shown with an '`x`'.

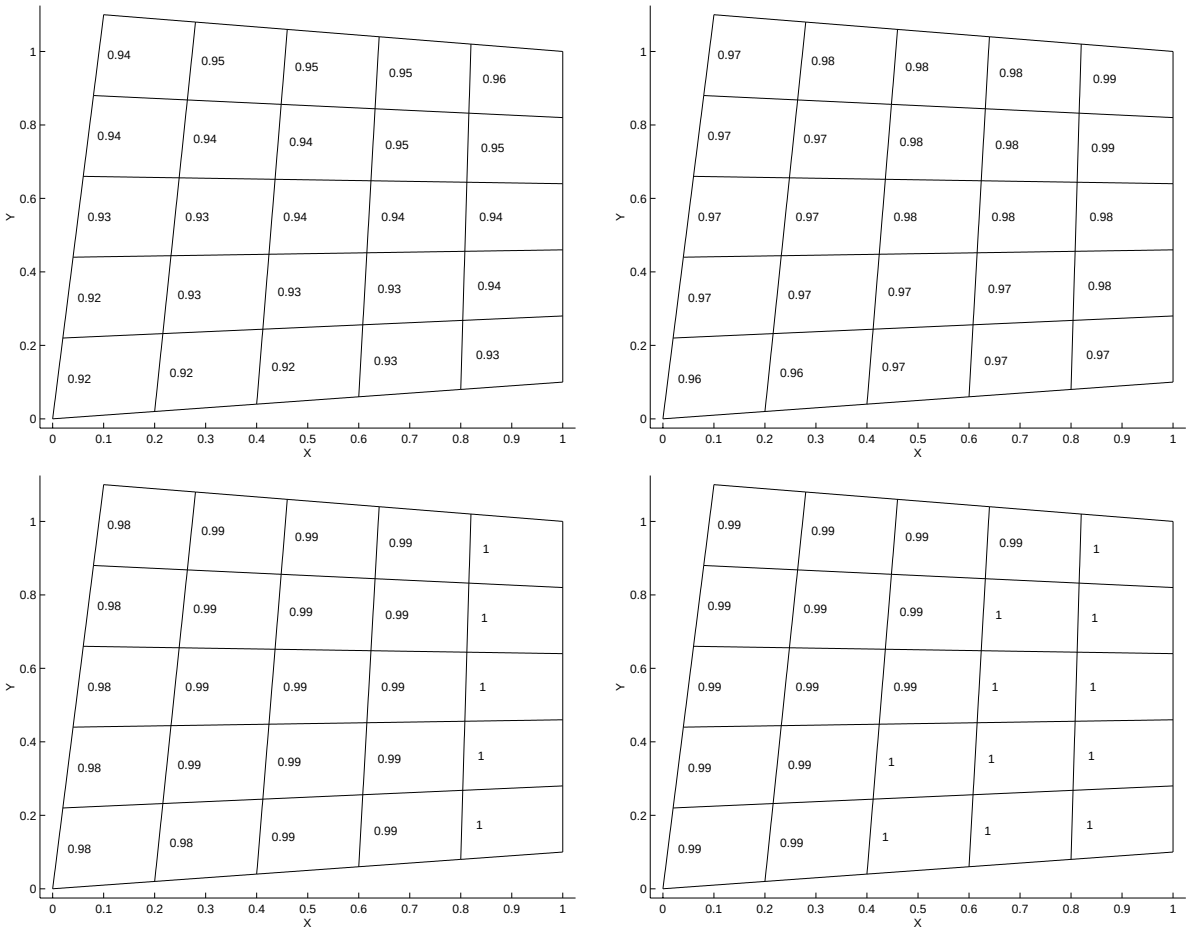


Figure 5.12: Local effectivity index values for *Example 2* on a 25-element uniform mesh for $p = 1$ to 4 from (upper left to lower right) with approximate boundary conditions.

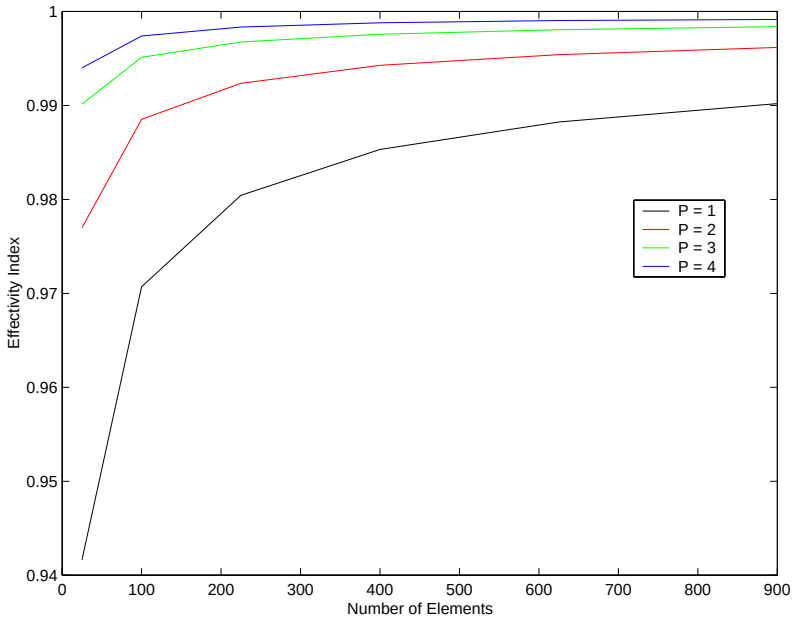


Figure 5.13: Global effectivity index under mesh refinement for *Example 2* with $p = 1$ to 4.

Example 3. We solve (5.1) on a uniform 100-element mesh and compare two cases in which we use nonuniform p approximations. In case one, we begin our computations with $p = 3$ approximation and then gradually decrease the approximation degree as we progress through the elements. In case two, we begin with $p = 1$ approximation and then gradually increase the approximation degree as we progress through the elements. Figure 5.14 illustrates the p's used on each element for both cases. In both situations, we use approximate boundary conditions.

The point of this example is to illustrate that for situations like case one where we start with a high degree approximation and then use progressively lower p's, that our error estimates do not work well on the first element after a transition, but then quickly recover and then maintain their effectiveness. We note that if we interperolated the U^- at the Radau points, as we did for the boundaries, that the recovery of the effectiveness would be faster. The second case in which we start with a low approximation degree and then gradually increase our p, we see that for this example our error estimates do not recover. Figure 5.15 shows the zero-level contour plot of the true error for both cases and Figure 5.16 gives the local effectivity indices. Notice that in the second case that the zero-level contour plots of the errors are parallel to the characterisic lines. This is something we do not fully understand. We also note that in regions where we increase the p, the error itself is smaller. The second case can also represent what happens to the performance of our error estimate in a region with a discontinuity, like we will see in *Examples 4* and *6*.

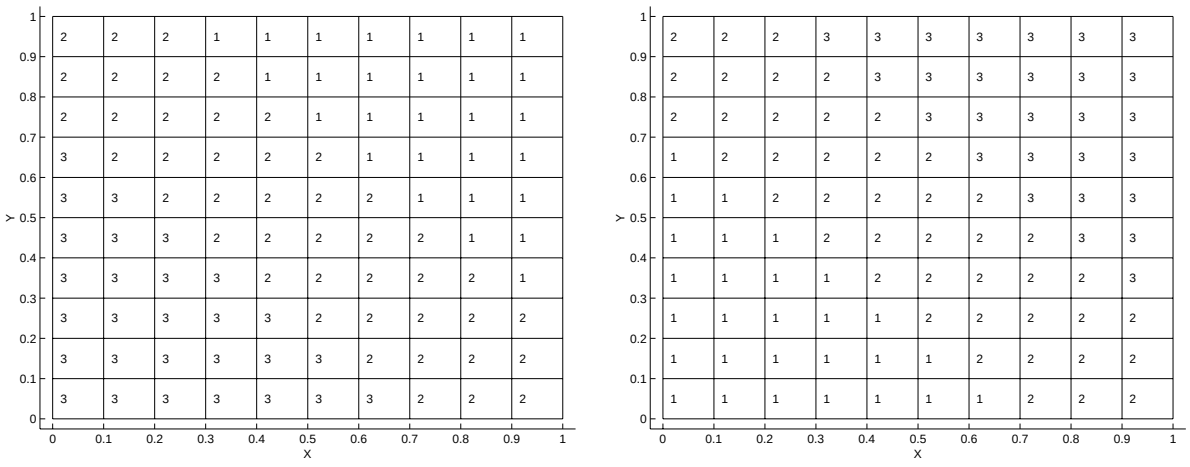


Figure 5.14: Uniform 100-element mesh for *Example 3* showing the approximation degrees on each element for case one and case two (left to right).

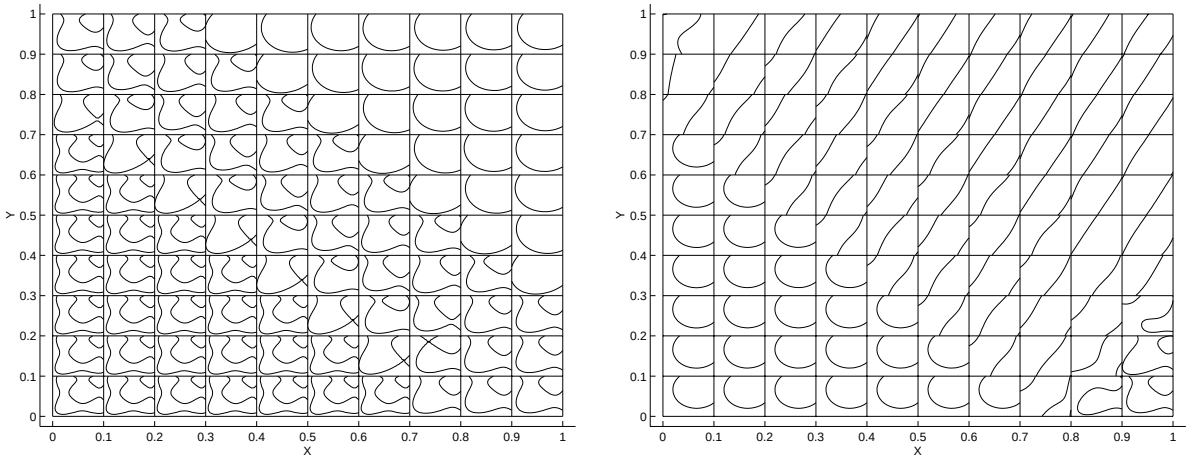


Figure 5.15: Zero-level curves of the error for *Example 3* with the p-distributions of Figure 5.14.

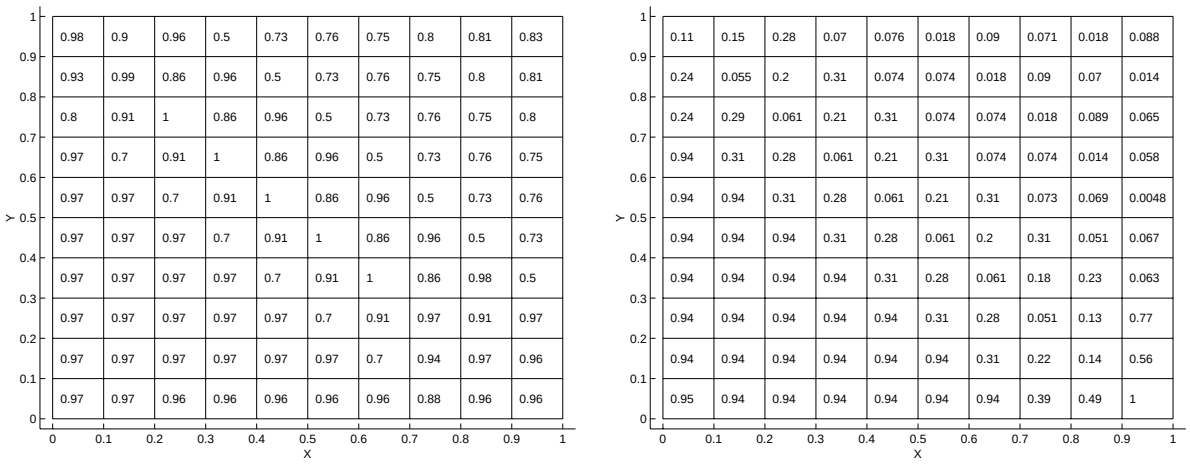


Figure 5.16: Local effectivity indices for *Example 3* and the p-distributions of Figure 5.14.

Example 4. We consider the following problem with a contact discontinuity

$$u_x + 2u_y = 0, \qquad (x, y) \in [0, 1]^2, \tag{5.2a}$$

subject to the boundary conditions

$$\begin{aligned} u(x, 0) &= e^{-2x}, & 0 \leq x \leq 1 \\ u(0, y) &= e^{y} + .25, & 0 < y \leq 1 \end{aligned}. \tag{5.2b}$$

The exact solution is

$$u(x, y) = \begin{cases} e^{-2x+y} + .25 & \text{if } x < y/2 \\ e^{-2x+y} & \text{if } x \geq y/2 \end{cases}. \tag{5.2c}$$

The true solution has a contact discontinuity along $y = 2x$. Therefore, the smoothness assumption of Theorem 4.1 is violated and as a result we expect the *a posteriori* error estimate to perform poorly near the discontinuity.

We begin by solving (5.2) on a uniform 64-element mesh with $p = 1, 2$ and show the true error in Figure 5.17. Then we solve (5.2) on meshes having 32×32 and 200×200 elements again with $p = 1, 2$ and present the local effectivity indices in Figure 5.18. These computational results indicate that the local effectivity indices on elements away from the discontinuity converge to unity under mesh refinement. Our error estimates perform poorly on elements near the discontinuity. Since we are not using limiting to suppress spurious oscillations near the discontinuity, the region around the discontinuity where the error is underestimated gets wider as p increases.

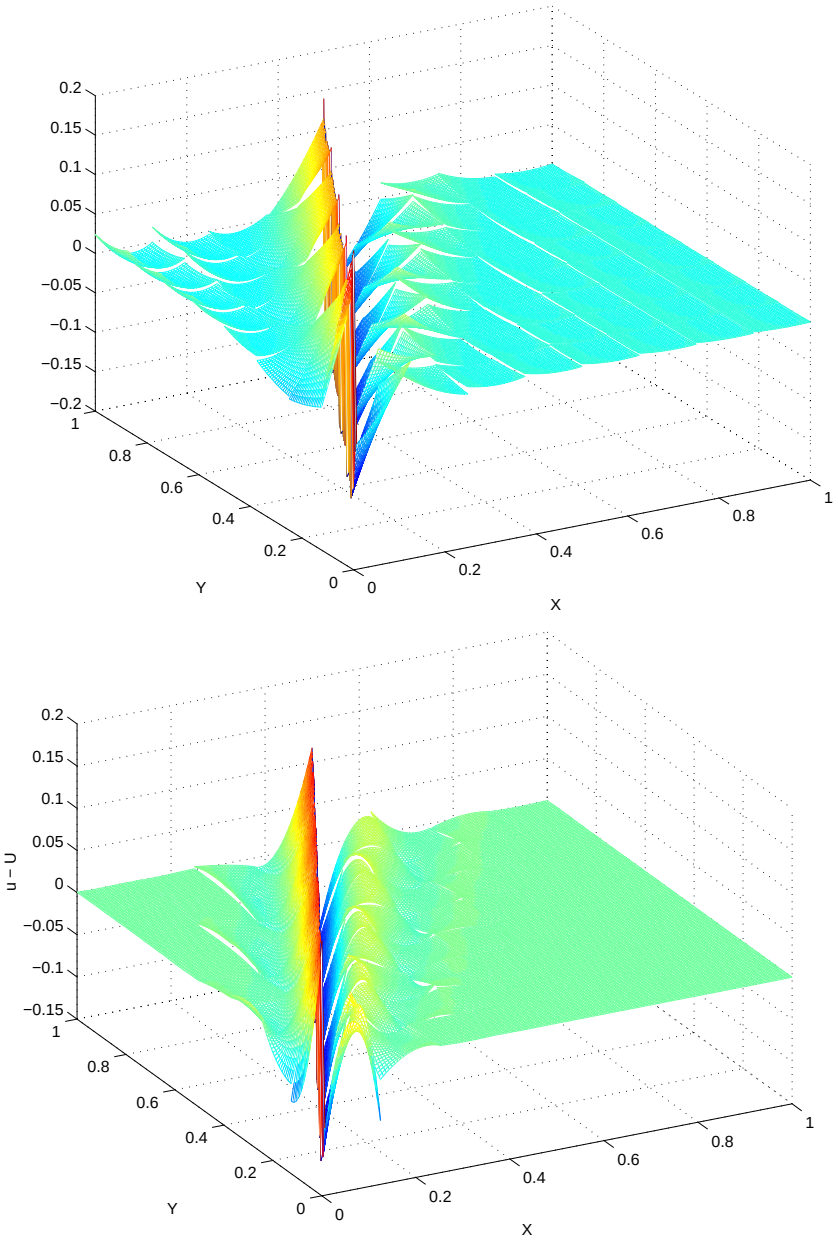


Figure 5.17: Surface plots of the error for *Example 4* with $p = 1, 2$ (top to bottom).

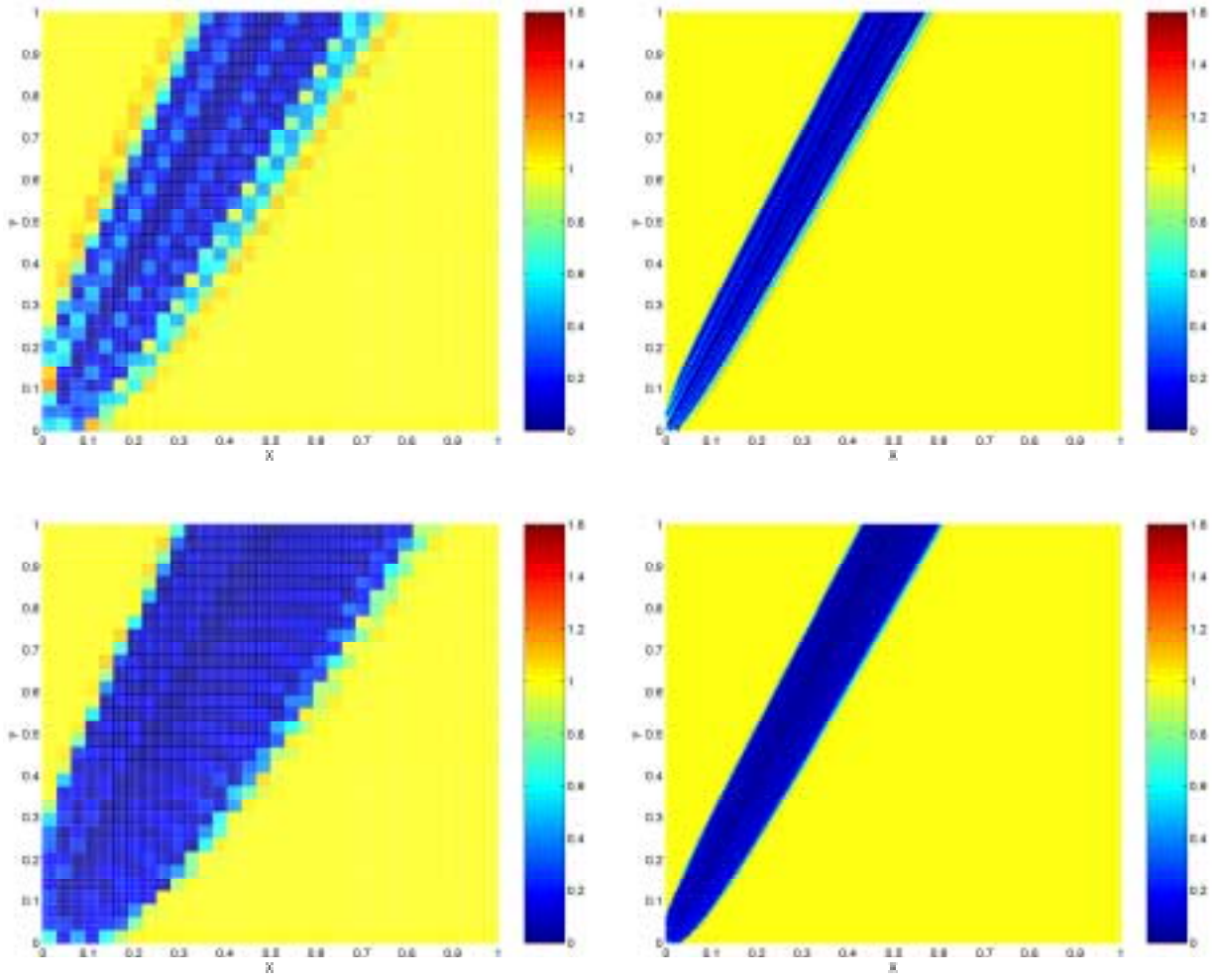


Figure 5.18: Local effectivity indices for *Example 4* with $(N, p) = (1024,1)$, $(40000, 1)$, $(1024,2)$ and $(40000,2)$ (upper left to lower right).

5.2 Nonlinear Examples

Example 5. We consider the inviscid Burger's equation

$$u_y + uu_x = f(x,y), \qquad (x,y) \in [-1,1]^2, \tag{5.3a}$$

subject to the boundary conditions

$$u(x,0) = \phi_1(x), \qquad \textit{and} \;\; u(y,0) = \phi_2(x). \tag{5.3b}$$

We select f, ϕ_1 and ϕ_2 such that the exact solution is

$$u(x,y) = \sqrt{1 + x^2 + 5y^2}. \tag{5.3c}$$

We solve (5.3) on meshes having $N = 35,\ 140,\ 315,\ 560,\ 875,\ 1260, 1715$ rectangular elements with an element aspect ratio $\Delta x/\Delta y = 5/7$ and $p = 1$ to 4, where Δx and Δy denote the length and width, respectively, of an element Δ. In Figure 5.19 we show the true error and the zero level contour plot of the true error for $p = 1$ on a 35-element mesh. We note that the error patterns still resemble a linear combination of $p + 1$-degree Radau polynomials in x and y. We present the $\mathcal{L}_2$ norm of the error in Table 5.14 and note that the convergence rate is slowed due to the stopping criteria used in Newton's method. We compute a posteriori error estimates by solving the linear problem (4.72) and show the effectivity indices in Table 5.15. We apply Newton's iteration to solve the nonlinear problem (4.73) using the linear error estimate (4.72) as an initial guess and present the effectivity indices in Table 5.16. In Figure 5.20 we present the local effectivity indices for both the linear estimate and the nonlinear estimate. While the global effectivity indices for both estimators are within 2% from unity, the error estimates obtained by solving the linear problem (4.72) are more efficient. Not only does the nonlinear error estimator require more computation, it can at times have multiple solutions that are spaced close together. This means that in order to have Newton's method converge to the desired solution we must have a very good initial guess. We note that in some of our numerical experiments while trying to compute the nonlinear error estimates (4.73), we failed to converge to the desired solution even when using the linear estimate (4.72).

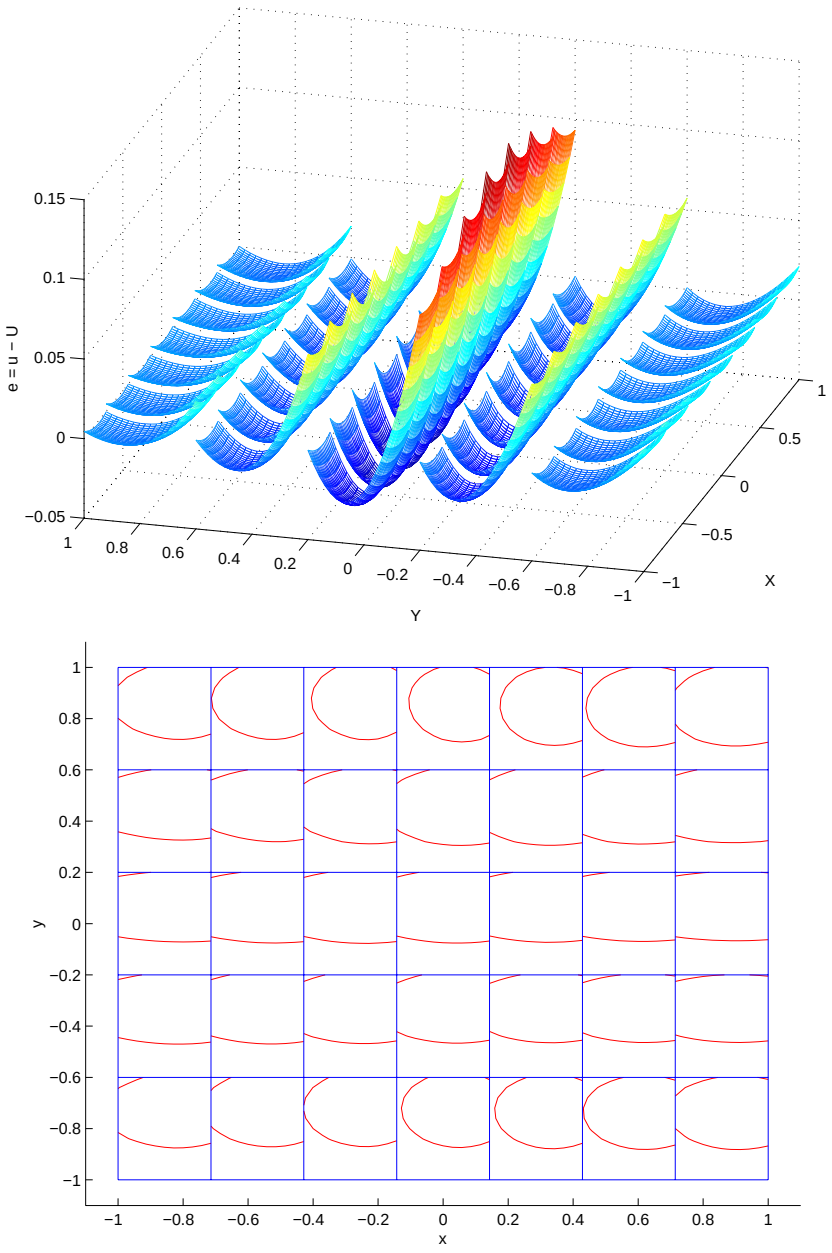


Figure 5.19: Surface plot of the true error (top) and zero-level curves of the true error (bottom) for *Example 5* with $p = 1$ on a 35-element mesh.

N	p = 1	p = 2	p = 3	p = 4
35	4.7081e-002	2.5253e-003	3.6895e-004	9.3047e-006
140	1.1756e-002	3.6035e-004	1.8583e-005	1.2655e-006
315	5.2293e-003	1.0723e-004	3.7464e-006	1.6036e-007
560	2.9423e-003	4.5322e-005	1.1895e-006	3.8297e-008
875	1.8834e-003	2.3224e-005	4.8801e-007	1.2581e-008
1260	1.3080e-003	1.3446e-005	2.3555e-007	5.0627e-009
1715	9.6100e-004	8.4699e-006	1.2721e-007	2.3443e-009

Table 5.14: $\mathcal{L}_2$ norm of the error for *Example 5* on meshes having $N = 35, 140, 315, 560, 875,$ 1260 and 1715 elements and $p = 1$ to 4.

N	p = 1	p = 2	p = 3	p = 4
35	0.996059	1.003180	0.985678	1.287716
140	0.999087	0.998213	1.008134	1.003167
315	0.999579	0.999240	1.008920	1.008087
560	0.999761	0.999572	1.009565	1.008689
875	0.999847	0.999726	1.009887	1.008989
1260	0.999894	0.999810	1.010074	1.009149
1715	0.999922	0.999860	1.010191	1.009245

Table 5.15: Effectivity indices for *Example 5* using the linearized error estimator (4.72) on meshes having $N = 35, 140, 315, 560, 875, 1260$ and 1715 elements and $p = 1$ to 4.

N	p = 1	p = 2	p = 3	p = 4
35	0.997489	1.004721	0.988137	1.280450
140	0.999433	0.998730	1.008672	1.003551
315	0.999733	0.999472	1.009163	1.008251
560	0.999848	0.999704	1.009703	1.008782
875	0.999902	0.999811	1.009975	1.009048
1260	0.999932	0.999869	1.010135	1.009190
1715	0.999950	0.999903	1.010236	1.009275

Table 5.16: Effectivity indices for *Example 5* using the nonlinear error estimator (4.73) on meshes having $N = 35, 140, 315, 560, 875, 1260$ and 1715 elements and $p = 1$ to 4.

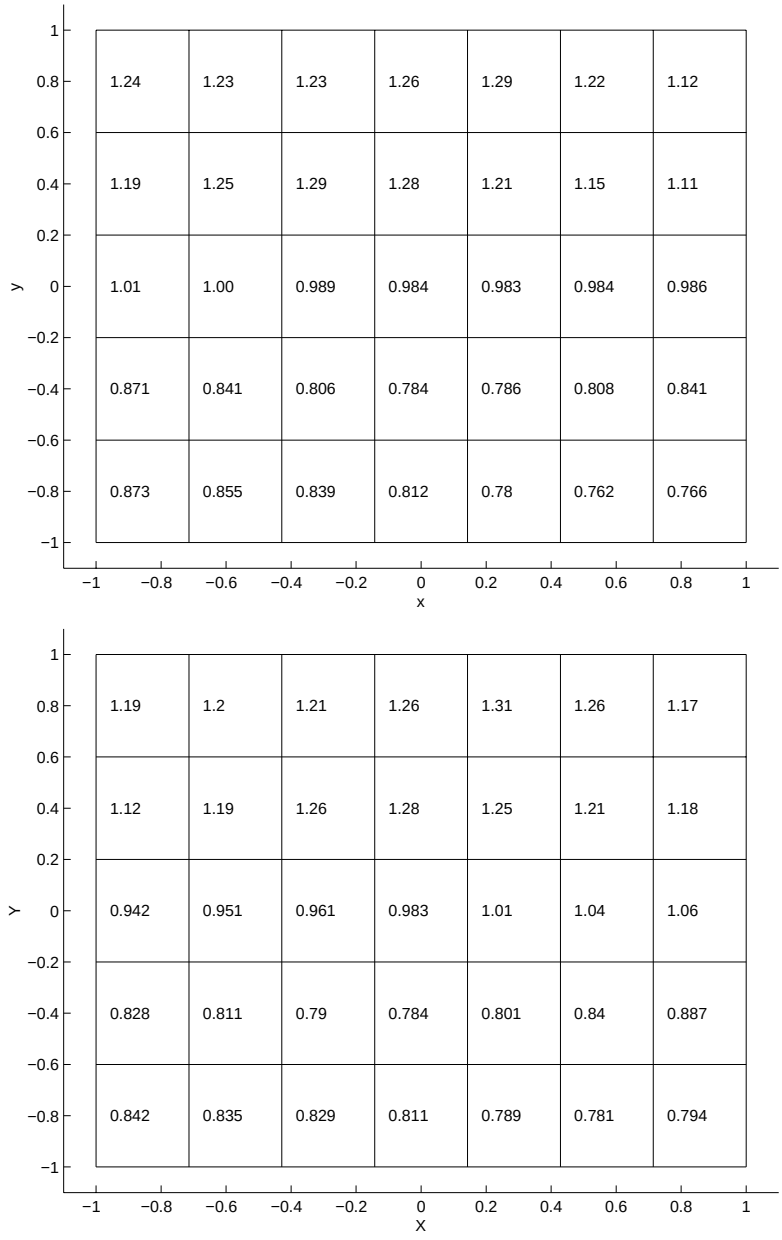


Figure 5.20: Local effectivity indices for *Example 5* with $p = 1$ on a 35-element mesh for the linearized (top) and the nonlinear (bottom) error estimate.

Example 6. Next we consider the homogeneous inviscid Burger's equation (5.3a) with $f(x, y) = 0$,

$$\phi_1(x, 0) = 1 + sin(\pi x)/2 \tag{5.4}$$

and select $\phi_2(0, y)$ such that the true solution is periodic and forms a shock discontinuity at the point $(\frac{2}{\pi} - 1, \frac{2}{\pi})$ which propagates along $y = x + 1$. First, we solve this problem on $\Omega = [-1, 1] \times [0, 0.4]$, where the solution is smooth, on meshes having $N = 35, 140, 315, 560, 875, 1260, 1680$ elements with an element aspect ratio $\Delta x/\Delta y = 25/7$, except for $N = 1680$, and $p = 1$ to 4. We compute an error estimate by solving (4.72) and present effectivity indices in Table 5.17. We apply Newton's iteration to solve the nonlinear problem (4.73) using the linear error estimate (4.72) as an initial guess and present the effectivity indices in Table 5.18. We also give a plot of these effectivity indices in Figure 5.21. This example shows that the effectivity indices converge to one only under h refinement which is due to the steepening of the wave as the shock forms. But again see that our linearized error estimate performs well for a nonlinear problem with a smooth solution.

N	p = 1	p = 2	p = 3	p = 4
35	0.9095	1.0698	0.7355	0.4947
140	1.0215	0.9184	0.7413	0.7246
315	0.9954	0.9443	0.9370	0.7349
560	1.0010	0.9851	0.9129	0.9487
875	1.0020	0.9832	0.9636	0.9269
1260	1.0025	0.9933	0.9680	0.9759
1680	0.9993	0.9866	0.9670	0.9770

Table 5.17: Effectivity indices for homogeneous Burger's equation (5.3a) with initial condition (5.4) on $[-1, 1] \times [0, 0.4]$ using the linearized error estimate (4.72) on meshes having $N = 35, 140, 315, 560, 875, 1260, 1680$ elements and $p = 1$ to 4.

N	p = 1	p = 2	p = 3	p = 4
35	0.8559	0.9823	0.6830	0.4693
140	0.9739	0.8720	0.7126	0.7002
315	0.9651	0.9098	0.9057	0.7176
560	0.9771	0.9535	0.8915	0.9272
875	0.9825	0.9578	0.9424	0.9111
1260	0.9859	0.9710	0.9505	0.9604
1680	0.9828	0.9645	0.9495	0.9615

Table 5.18: Effectivity indices for homogeneous Burger's equation (5.3a) with initial condition (5.4) on $[-1, 1] \times [0, 0.4]$ using the nonlinear error estimate (4.73) on meshes having $N = 35, 140, 315, 560, 875, 1260, 1680$ elements and $p = 1$ to 4.

We conclude by solving the previous problem on $\Omega = [-1, 1] \times [0, 1.999]$ on meshes having $N = 35, 140, 315, 1260$ and 14000 elements with an element aspect ratio $\Delta x/\Delta y =$

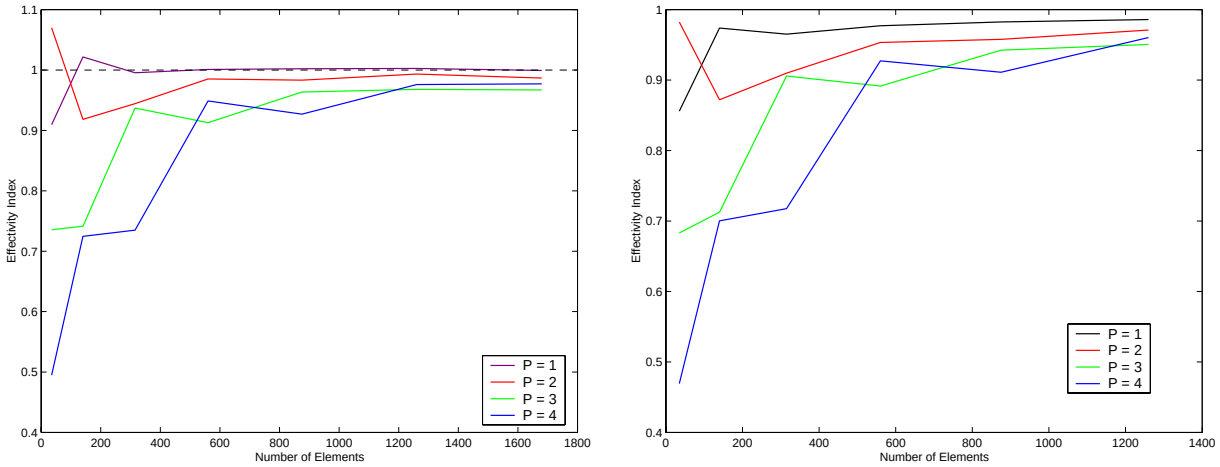


Figure 5.21: Effectivity indices for Example 6 with $\Omega = [-1, 1] \times [0, .4]$, $p = 1$ to 4 and meshes having $N = 35, 140, 315, 560, 875, 1260, 1680$ elements for the linearized (left) and the nonlinear (right) error estimate.

7/5 and $p = 1, 2$. We give a comparison of the local effectivity indices for the nonlinear error estimate and the linearized error estimate while using a $p = 1$ approximation with 35 elements in Figure 5.22. The shock line has been drawn in for the sake of clarity. Next, in Figure 5.23 we plot the zero level curves of the true error using a $p = 1$ approximation with 140 elements and 315 elements. Notice that the zero-level curves tend to be parallel to the contact discontinuity, something we do not fully understand.

Because we are not using slope limiters we see oscillations form in our computed solution due to the discontinuity. These oscillations carry over into the error and as such, destroy the DG error patterns that our estimate is based on. The good news is that these oscillations are limited to a bandwidth along the discontinuity where they destroy our estimate. Beyond the bandwidth the oscillations die out and our estimate performs well. The bandwidth appears not to be affected by the physical domain, but by the size of h. In fact as h shrinks the bandwidth's physical domain width also shrinks. This is evidenced by looking at the local effectivity indices shown in Figure 5.24. Furthermore, as expected, the number of elements involved in the bandwidth increases as p increases, since with higher p the oscillations are more pronounced.

These computational results indicate that the theoretical results of theorem 4.1 are valid for nonlinear problems in regions where the solution is smooth, i.e., the local effectivity indices converge to unity under mesh refinement in regions where the solution is smooth. The regions in the lower right corner of the domain with underestimated errors correspond to regions of relatively small discretization errors and thus, should not affect the quality of the global effectivity index.

In order to get the true solution for Example 6, we used the method of characteristics and solved a nonlinear equation using Newton's method with a very advantageous initial guess suggested by Shu [79]. Additionally, we point out that we used the splitting of the elements

discussed at the end of Chapter 3, to more accurately compute the numerical integrations of the $\mathcal{L}_2$ norm of the error, for the elements where the true solution was discontinuous.

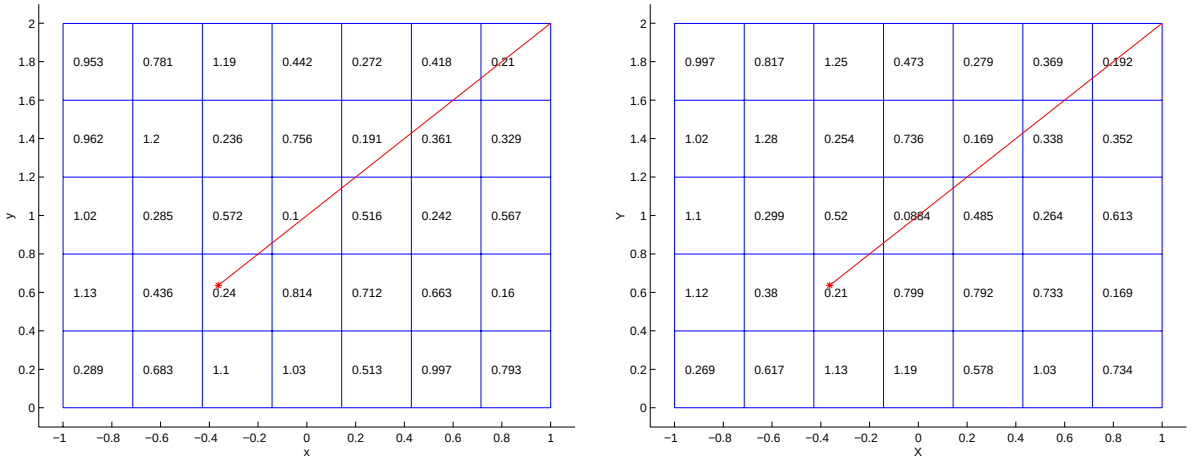


Figure 5.22: Local effectivity indices for *Example 6* with $\Omega = [-1, 1] \times [0, 1.999]$ using $p = 1$ with a 35-element mesh and for the linearized (left) and the nonlinear (right) error estimate.

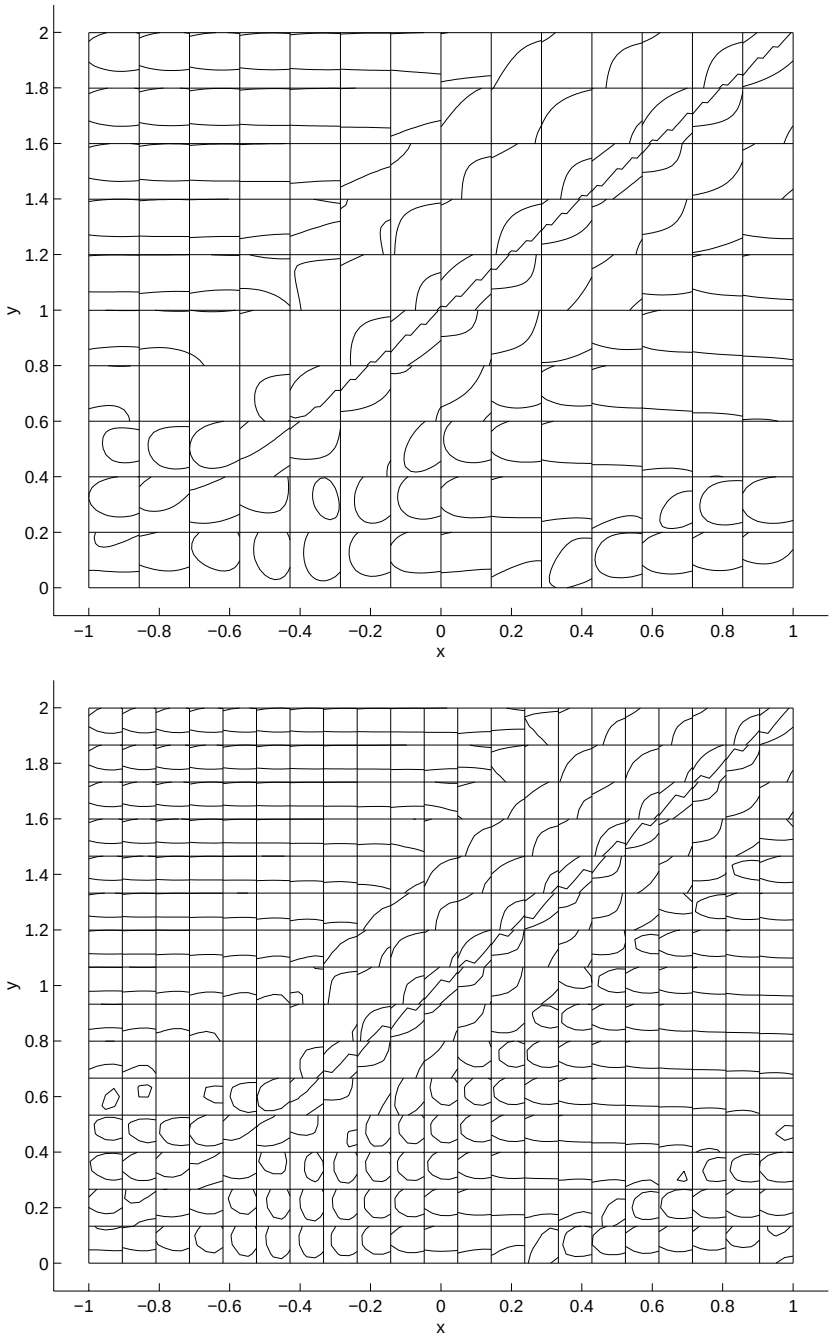


Figure 5.23: Zero-level curves of the error using a $p = 1$ approximation with 140 elements and 315 elements (top to bottom) for *Example 6* with $\Omega = [-1, 1] \times [0, 1.999]$.

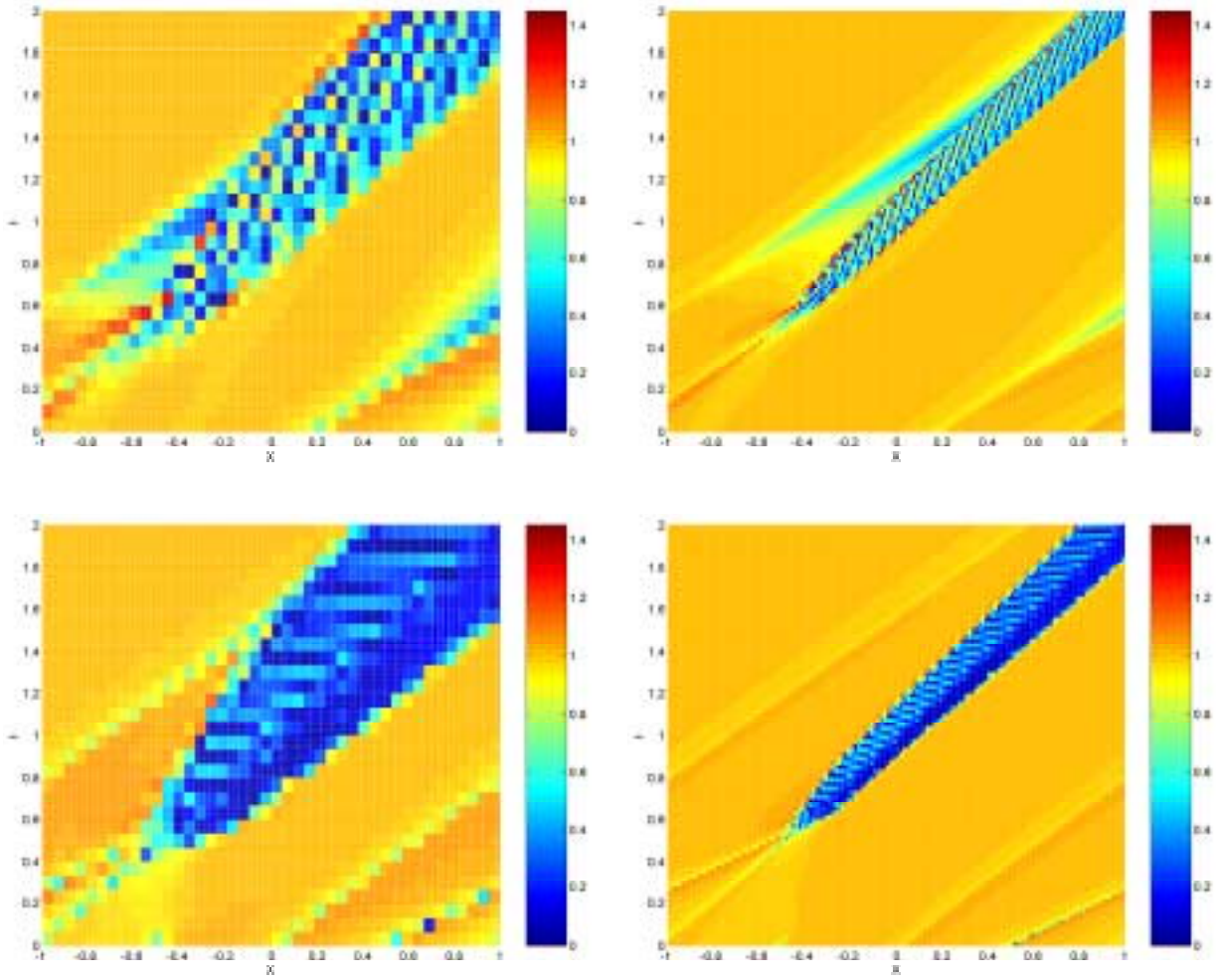


Figure 5.24: Local effectivity indices for the homogeneous Burger's equation (5.3a) with initial condition (5.4) on $\Omega = [-1, 1] \times [0, 1.999]$ using meshes having $(N, p) = (1260,1)$, (14000,1), (1260,2) and (14000,2) (upper left to lower right).

Chapter 6

Contributions and Future Work

6.1 Contributions

We developed a class of finite element methods called the flexible Galerkin finite element method (FGM). The FGM is a novel idea that allows the user to prescribe on each element the degree of the approximation and the level of continuity between elements. In this way, the FGM encompasses both the traditional continuous Galerkin method (CGM) and the now popular discontinuous Galerkin method (DGM). The FGM allows the user to choose the full range of continuity levels between fully continuous and fully discontinuous. Using the FGM we showed the rate of convergence for the $\|e\|_{\mathcal{L}_2(\Omega)}$ was bounded between that of the DGM and the CGM. We illustrated the utility of the FGM by listing the examples of [9, 71, 38] where one uses the CGM on part of the domain and then switches over and uses the DGM without having to stop and start two separate programs. We derived the FGM formulation for two-dimensional hyperbolic problems and we constructed efficient storage and retrieval data structures for the finite element solution.

We then used the FGM applied to hyperbolic problems and designated to run fully discontinuous, i.e., as the DGM, to perform numerical experiments that lead us to prove that the discontinuous Galerkin finite element error can be split into an $O(h^{p+1})$ leading component and a higher order component. We further showed that the leading term is spanned by two $(p+1)-$degree Radau polynomials in the x and y directions, respectively. We showed that the p-degree discontinuous finite element solution is superconvergent at Radau points obtained as a tensor product of the roots of $(p+1)$-degree Radau polynomials. For conservation laws, the p-degree DGM solution flux exhibits a strong $O(h^{2p+2})$ local superconvergence on average at the element outflow boundary. We established an $O(h^{2p+1})$ global superconvergence for the solution flux at the outflow boundary of the domain. For a hyperbolic problems with a nonlinear reaction term we showed a point-wise $O(h^{p+2})$ superconvergence at the Radau points and a strong superconvergence of the flux on the outflow boundary of $O(h^{\min(2p+2,p+4)})$. The strong superconverge of the flux on the outflow boundary indicates that the error in the discontinuous Galerkin methods propagates at a higher order.

We used these result to construct an a posteriori estimate for multi-dimensional discontinuous Galerkin finite element error for hyperbolic problems on rectangular meshes. The error estimation procedure is simple, can be computed in several norms and does not require any communication across neighboring elements. The later property makes this estimation procedure useful for parallel computations. The a posteriori error estimates developed in this paper readily extend to three-dimensional hexahedral elements.

6.2 Future Work

We plan to develop a working code of the FGM applied to systems of convection-diffusion problems using general meshes. In such cases the flux terms could be handled in a similar manner to the flux terms in the LDG method described by Cockburn and Shu [36]. We plan to study the error for the FGM including the rates of convergence for the $\|e\|_{\mathcal{L}_2(\Omega)}$ and hope to derive error estimates.

We plan to extend the error estimates and superconvergence results of the DGM to unstructured tetrahedral meshes for hyperbolic systems. Furthermore, we note that our error estimate does not apply near discontinuities and so far we are not able to construct asymptotically correct error estimates that apply near discontinuities.

Bibliography

[1] S. Adjerid, *A posteriori error estimation for fourth-order elliptic problems*, Computer Methods in Applied Mechanics and Engineering **191** (2002), no. 23-24, 2539–2559.

[2] S. Adjerid, M. Aiffa, and J.E. Flaherty, *High-order finite element methods for singularly perturbed elliptic and parabolic problems*, SIAM Journal on Applied Mathematics **55** (1995), no. 2, 520–543.

[3] ———, *Hierarchical finite element bases for triangular and tetrahedral elements*, Computer Methods in Applied Mechanics and Engineering **190** (2000), no. 22-23, 2925–2941.

[4] S. Adjerid, B. Belguendouz, and J.E. Flaherty, *A posteriori finite element error estimation for diffusion problems*, SIAM Journal on Scientific Computing **21** (1999), no. 2, 728–746.

[5] S. Adjerid, K.D. Devine, J.E. Flaherty, and L. Krivodonova, *A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems*, Computer Methods in Applied Mechanics and Engineering **191** (2002), no. 11-12, 1097–1112.

[6] S. Adjerid, J.E. Flaherty, and I. Babuška, *A posteriori error estimation for the finite element method-of-lines solution of parabolic problems*, Mathematical Models and Methods in Applied Sciences **9** (1999), no. 2, 261–286.

[7] S. Adjerid, J.E. Flaherty, P.K. Moore, and Y.J. Wang, *High-order adaptive methods for parabolic systems*, Physica D **60** (1992), 94–111.

[8] S. Adjerid, J.E. Flaherty, and Y.J. Wang, *A posteriori error estimation with finite element methods of lines for one-dimensional parabolic systems*, Numerische Mathematik **65** (1993), 1–21.

[9] P. Alotto, A. Bertoni, H. Perugia, and D. Schötzau, *Discontinuous finite element methods for the simulation of rotating electrical machines*, COMPEL: International Journal for Computation and Mathematics in Electrical and Electronic Engineering **20** (2001), no. 2, 448–462.

[10] I. Babuša, C.E. Baumann, and J.T. Oden, *A discontinuous hp finite element method for diffusion problems: 1-D analysis*, Computers and Mathematics with Applications **37** (1999), 103–122.

[11] I. Babuška, M. Feistauer, and P. Šolín, *On one approach to a posteriori error estimates for evolution problems solved by the method of lines*, Numerische Mathematik **89** (2001), no. 2, 225–256.

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

[13] C.E. Baumann and J.T. Oden, *A discontinuous hp finite element method for convection-diffusion problems*, Computer Methods in Applied Mechanics and Engineering **175** (1999), 311–341.

[14] K.S. Bey, *An hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws*, Ph.D. thesis, University of Texas as Austin, 1994.

[15] K.S. Bey and J.T. Oden, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws*, Computer Methods in Applied Mechanics and Engineering **133** (1996), 259–286.

[16] K.S. Bey, J.T. Oden, and A. Patra, *A parallel hp−adaptive discontinuous Galerkin method for hyperbolic conservation laws*, Applied Numerical Mathematics **20** (1996), 321–386.

[17] K.S. Bey, A. Patra, and J.T. Oden, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws: A parallel adaptive strategy*, International Journal for Numerical Methods in Engineering **38** (1995), 3889–3908.

[18] R. Biswas, K.D. Devine, and J.E. Flaherty, *Parallel, adaptive finite element methods for conservation laws*, Applied Numerical Mathematics **14** (1994), 255–283.

[19] C.A. Blain and M. Cobb, *Shelf-scale model simulations of nearshore flows: Part I. longshore currents*, Journal of Atmospheric and Oceanic Technology (2002), in review.

[20] K. Böttcher and R. Rannacher, *Adaptive error control in solving ordinary differential equations by the discontinuous Galerkin method*, Tech. Report 96-53 preprint, IWR, Universität Heidelberg, 1996.

[21] D. Braess, *Finite elements*, Cambridge University Press, Cambridge, 1997.

[22] S.C. Brenner and L.R. Scott, *The mathematical theory of finite element methods*, Springer-Verlag, New York, 1994.

[23] P. Castillo, B. Cockburn, D. Schötzau, and C. Schwab, *Optimal a priori error estimates for the hp-version of the local discontinuous Galerkin method for convection diffusion problems*, Journal of Mathematics of Computation **71** (2002), 455–478.

[24] Z. Chen, *On the relationship of various discontinuous finite element methods for second-order elliptic equations*, East-West Journal on Numerical Math. **9** (2001), 99–122.

[25] ———, *Characteristic mixed discontinuous finite element methods for advection-dominated diffusion problems*, Computer Methods in Applied Mechanics and Engineering **191** (2002), no. 23-24, 2509–2538.

[26] S. Chippada, C. Dawson, M.L. Martinew, and M.F. Wheeler, *A Godunov-type finite volume method for the system of shallow water equations*, Computer Methods in Applied Mechanics and Engineering **151** (1998), 105–129, Proceedings of the 1997 Symposium on Advances in Computational Mechanics.

[27] B. Cockburn, *A simple introduction to error estimation for nonlinear hyperbolic conservation laws*, Proceedings of the 1998 EPSRC Summer School in Numerical Analysis, SSCM, volume 26 of the Graduate Student's Guide for Numerical Analysis, pages 1-46 (Berlin), Springer, 1999.

[28] B. Cockburn and D. Dawson, *Some extensions of the local discontinuous Galerkin method for convection-diffusion equations in multidimensions*, Tech. Report 99-27, Texas Institute for Computational and Applied Mathematics, 1999.

[29] B. Cockburn and P. A. Gremaud, *Error estimates for finite element methods for nonlinear conservation laws*, SIAM Journal on Numerical Analysis **33** (1996), 522–554.

[30] B. Cockburn, G. Kanschat, I. Perugia, and D. Schötzau, *Superconvergence of the local discontinuous Galerkin method for elliptic problems on cartesian grids*, SIAM Journal on Numerical Analysis **39** (2002), no. 1, 264–285.

[31] B. Cockburn, G. Karniadakis, and C.-W. Shu (eds.), *Discontinuous Galerkin methods: Theory, computation and applications*, Lecture Notes in Computational Science and Engineering, no. 11, Springer, New York, 2000.

[32] B. Cockburn, S.Y. Lin, and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin methods of scalar conservation laws III: One dimensional systems*, Journal of Computational Physics **84** (1989), 90–113.

[33] B. Cockburn, M. Luskin, C.-W. Shu, and E. Süli, *Enhanced accuracy by post-processing for finite element methods for hyperbolic equations*, Journal of Mathematics of Computation (2002), to appear in print.

[34] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework*, Mathematics of Computation **52** (1989), no. 186, 411–435.

[35] ———, *The Runge-Kutta local projection P^1-discontinuous Galerkin finite element method for scalar conservation laws*, RAIRO Modél. Math. Anal. Numér. **25** (1991), 337–361.

[36] ———, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM Journal on Numerical Analysis **35** (1998), no. 6, 2440–2463.

[37] C. Dawson, *High resolution upwind-mixed finite element methods for advection-diffusion equations with variable time-stepping*, Numerical Methods for Partial Differential Equations **11** (1995), 525–538.

[38] C. Dawson and J. Proft, *Coupling of continuous and discontinuous Galerkin methods for transport problems*, Computer Methods in Applied Mechanics and Engineering **191** (2002), 3212–3231.

[39] K.D. Devine and J.E. Flaherty, *Parallel adaptive hp−refinement techniques for conservation laws*, Computer Methods in Applied Mechanics and Engineering **20** (1996), 367–386.

[40] K.D. Devine, J.E. Flaherty, R.M. Loy, and S.R. Wheat, *Parallel partitioning strategies for the adaptive solution of conservation laws*, Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations (New York) (I. Babuška, J.E. Flaherty, W.D. Henshaw, J.E. Hopcroft, J.E. Oliger, and T. Tezduyar, eds.), vol. 75, Springer-Verlag, 1995, pp. 215–242.

[41] S. Dey, M.S. Shephard, and J.E. Flaherty, *Geometry representation issues associated with p-version finite element computations*, Computer Methods in Applied Mechanics and Engineering **150** (1997), 39–55.

[42] Arnold D.N., F. Brezzi, B. Cockburn, and L.D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM Journal on Numerical Analysis **39** (2002), 1749–1779.

[43] D.A. Dunavant, *High degree efficient symmetrical gaussian quadrature rules for the triangle*, International Journal for Numerical Methods in Engineering **21** (1985), 1129–1148.

[44] K. Ericksson and C. Johnson, *Adaptive finite element methods for parabolic problems I: A linear model problem*, SIAM Journal on Numerical Analysis **28** (1991), 12–23.

[45] _____, *Adaptive finite element methods for parabolic problems II: Optimal error estimates in $l_\infty l_2$ and $l_\infty l_\infty$*, SIAM Journal on Numerical Analysis **32** (1995), 706–740.

[46] G. Evans, *Practical numerical integration*, John Wiley & Sons, New York, 1993.

[47] L.C. Evans, *Partial differential equations*, American Mathematical Society, Providence, RI, 1998.

[48] S.J. Farlow, *Partial differential equations for scientists and engineers*, Dover, New York, 1993.

[49] J.E. Flaherty, *Finite difference*, Finite Difference Lecture Notes, Rensselaer Polytechnic Institute, 2000.

[50] _____, *Finite element analysis*, Finite Element Lecture Notes, Rensselaer Polytechnic Institute, 2000.

[51] J.E. Flaherty, R. Loy, M.S. Shephard, B.K. Szymanski, J. D. Teresco, and L. H. Ziantz, *Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws*, Journal of Parallel and Distributed Computing **47** (1997), 139–152.

[52] P.R. Garabedian, *Partial differential equations*, AMS Chelsea Publishing, Providence, RI, 1998.

[53] R. Haberman, *Elementary applied partial differential equations with fourier series and boundary value problems*, third ed., Prentice-Hall, Upper Saddle River, N.J., 1998.

[54] O.J. Hammer, P.C.and Marlowe and A.H. Stroud, *Numerical integration over simplexes and cones*, Mathematical Tables and Other Aids to Computation **10** (1956), no. 55, 130–137.

[55] P.C. Hammer and A.H. Stroud, *Numerical integration over simplexes*, Mathematical Tables and Other Aids to Computation **10** (1956), no. 55, 137–139.

[56] P. Houston, C. Schwab, and E. Süli, *Discontinuous hp-finite element methods for advection-diffusion problems*, Tech. Report 2000-07, Numerical Analysis Group, 2000.

[57] P. Houston and E. Süli, *hp-adaptive discontinuous Galerkin finite element methods for first-order hyperbolic problems*, Tech. Report 2001-05, Numerical Analysis Group, 2001.

[58] G. Jiang and C.-W. Shu, *On a cell entropy inequality for discontinuous Galerkin methods*, Mathematics of Computation **62** (1994), no. 206, 531–538.

[59] C. Johnson, *Numerical solution of partial differential equations by the finite element method*, Cambridge University Press, New York, 1992.

[60] C. Johnson and J. Pitkäranta, *An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation*, Mathematics of Computation **46** (1986), no. 173, 1–26.

[61] L. Johnson and D. Riess, *Numerical analysis*, second ed., Addison-Wesley, Reading, Massachusetts, 1982.

[62] D. Kincaid and W. Cheney, *Numerical analysis*, second ed., Brooks-Cole Publishing Company, Pacific Grove, CA, 1996.

[63] L. Krivodonova and J.E. Flaherty, *Error estimation for discontinuous Galerkin solutions of multidimensional hyperbolic problems*, submitted, 2002.

[64] S. Lang, *Calculus of several variables*, third ed., Springer-Verlag, New York, 1987.

[65] M. Larson and T. Barth, *A posteriori error estimation for adaptive discontinuous Galerkin approximation of hyperbolic systems*, Proc. International Symposium on Discontinuous Galerkin Methods Theory, Computation and Applications (Berlin) (B. Cockburn, G. E. Karniadakis, and C. W. Shu, eds.), Springer, 2000.

[66] P.D. Lax, *The formation and decay of shock waves*, American Mathematical Monthly **79** (1972), no. 3, 227–241.

[67] R.B. Lowrie, *Compact higher-order numerical methods for hyperbolic conservation laws*, Ph.D. thesis, University of Michigan, 1996.

[68] R.A. Luettich, J.J. Westerink, and N.W. Scheffner, *ADCIRC: An advanced three-dimensional circulation model for shelves, coasts, and estuaries*, Tech. report, Dept. of the Army, U.S. Army Corps of Engineers, Washington, DC, 1991.

[69] J.T. Oden, I. Babuška, and C.E. Baumann, *A discontinuous hp finite element method for diffusion problems*, Journal of Computational Physics **146** (1998), 491–519.

[70] LeSaint P. and P.A. Raviart, *On a finite element method for solving the neutron transport equation*, Mathematical Aspects of Finite Elements in Partial Differential Equations (New York) (C. de Boor, ed.), Academic Press, 1974, pp. 89–145.

[71] H. Perugia and D. Schötzau, *On the coupling of local discontinuous Galerkin and conforming finite element methods*, Journal of Scientific Computing **16** (2001), no. 4, 411–433.

[72] W.H. Reed and T.R. Hill, *Triangular mesh methods for the neutron transport equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[73] M. Renardy and Rogers R.C., *An introduction to partial differential equations*, Springer-Verlag, New York, 1993.

[74] G.R. Richter, *An optimal-order error estimate for the discontinuous Galerkin method*, Mathematics of Computation **50** (1988), no. 181, 75–88.

[75] B. Rivière and M.F. Wheeler, *A posteriori error estimation and mesh adaptation strategy for discontinuous Galerkin methods applied to diffusion problems*, TICAM Report 00-10, 2000.

[76] D. Schötzau and C. Schwab, *An hp a-priori error analysis of the dg time-stepping method for initial value problems*, CALCOLO **37** (2000), no. 4, 207–232.

[77] ———, *Time discretization of parabolic problems by the hp-version of the discontinuous Galerkin finite element method*, SIAM Journal on Numerical Analysis **38** (2000), no. 3, 837–875.

[78] M.S. Shephard, S. Dey, and J.E. Flaherty, *A straightforward structure to construct shape functions for variable p-order meshes*, Computer Methods in Applied Mechanics and Engineering **147** (1997), 209–233.

[79] C.-W. Shu, *A FORTRAN routine of Newton's method for solving Burger's equation with a shock*, 2002, Personal communication.

[80] M.R. Spiegel, *Mathematical handbook of formulas and tables*, Schaum's Outline Series, McGraw-Hill, New York, 1995.

[81] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, second ed., Springer-Verlag, New York, 1993.

[82] A.H. Stroud, *Approximate calculation of multiple integrals*, Prentice-Hall, Englewood Cliffs, N.J., 1971.

[83] A.H. Stroud and D. Secrest, *Gaussian quadrature formulas*, Prentice-Hall, Englewood Cliffs, N.J., 1966.

[84] E. Süli, *A posteriori error analysis and adaptivity for finite element approximations of hyperbolic problems*, An introduction to recent developments in theory and numerics for conservation laws, volume 5 of Lecture Notes in Computational Sciences and Engineering (Berlin) (D. Kröner, M. Ohlberger, and C. Rhode, eds.), Springer, 1999.

[85] B. Szabó and I. Babuška, *Finite element analysis*, John Wiley & Sons, New York, 1991.

[86] V. Thomée, *Galerkin finite element methods for parabolic problems*, Springer Series in Computational Mathematics, Springer-Verlag, New York, 1997.

[87] T. Werder, K. Gerdes, D. Schötzau, and C. Schwab, *hp discontinuous Galerkin time stepping for parabolic problems*, Computer Methods in Applied Mechanics and Engineering **190** (2001), no. 49-50, 6685–6708.

[88] M.F. Wheeler, *An elliptic collocation-finite element method with interior penalties*, SIAM Journal on Numerical Analysis **15** (1978), 152–161.

[89] R. Winther, *A stable finite element method for initial-boundary value problems for first-order hyperbolic systems*, Mathematics of Computation **36** (1981), no. 153, 65–86.

Vita

Thomas Christopher Massey is the only son of Thomas and Darlene Massey and was born on December 7th, 1970 in Opelika, Alabama. In 1989 he graduated with honors from Opelika High School. He then attended Troy State University in Troy, Alabama where he earned a Bachelor of Science degree in Mathematics and Physics Education in 1993. He worked as a high school math and science teacher at Charles Henderson High School in Troy, Alabama from 1993 to 1996. During this same time he continued his education at Troy State University where in 1996 we earned a Master of Science degree in Mathematics Education.

In May 1996 the author was admitted to the Master of Science Program in Mathematics at Virginia Polytechnic Institute and State University, Blacksburg, VA. In 1999, he received his Master of Science in Mathematics degree and was admitted to the Doctor of Philosophy Program in Mathematics at Virginia Polytechnic Institute and State University. While pursuing his degrees at Virginia Tech, Chris worked as a GTA and GRA for the Math department where he was the recipient of the 2000 Outstanding Graduate Teaching Assistant award.

In July 2002, Chris earned his Doctor of Philosophy in Mathematics from Virginia Tech. He plans to move to Louisiana and work at the Naval Research Laboratory located at the John Stennis Space Center in Mississippi, where he will continue to work on discontinuous Galerkin finite element methods.