Discontinuous Galerkin Methods

Nick Burgess
STC Corp. and U.S. Army Aeroflightdynamics Directorate

August 21, 2013

Contents

1 Introduction **4**

2 Finite-element Methods **6**
2.1 Method of Weighted Residuals 6
2.1.1 Galerkin's Method 8
2.1.2 Briefly on Galerkin's Method and Continuity 11
2.1.3 Other MWR Based Methods 11
2.1.4 A simple example of Galerkin's method for an ODE . . . 12
2.2 Classical Finite-element Methods 13
2.2.1 Implementation 15

3 Mesh Data Structures **18**
3.1 Introduction 18
3.2 Element Types 19
3.2.1 1-D Bar 19
3.2.2 2-D Triangle 19
3.3 Mesh Connectivity 19

Bibliography **22**

List of Figures

2.1 Example of linear Lagrange or so-called hat functions defined locally on the elements in a mesh 13
2.2 Standard element and local basis functions 13
2.3 Local and global node numbering 16

3.1 1-D "bar" element. 19
3.2 Sample Mesh. 20

List of Tables

2.1 Various MWR methods for various choices of weight function . . 11

Chapter 1

Introduction

Discontinuous Galerkin (DG) methods are a particular type of finite-element method that is particularly well suited for convection dominated problems. The name is derived from their close relationship to standard Galerkin finite-element methods. However, in contrast to typical Galerkin finite-element methods no continuity requirements are imposed on the basis functions. DG methods are a particularly attractive numerical method because they sit on a very solid mathematical background. DG methods come equipped with all the standard finite-element method proofs of stability, consistency, and convergence at any discretization order. This is a property that is not shared with finite-volume and similar unstructured grid numerical methods.

DG methods are, in their most basic form, a finite-element method. However, typical finite-element methods are continuous finite-element methods where the basis functions, which approximate the discrete solution, are continuous at the element interfaces. Continuous finite-element methods traditionally have been applied to linear structural and thermal analysis problems that constitute purely elliptic operators, and hence continuous basis functions are appropriate. DG methods employ basis functions that are discontinuous at the element interfaces, which makes DG methods naturally suitable for computing convection dominated problems. DG discretizations are an ideal choice for convection dominated problems because the discontinuous basis functions allow for upwind flux calculations using approximate Riemann solvers. Employing approximate Riemann solvers at the element interfaces is a strategy that is borrowed from finite-volume methods. Thus DG can be thought of as a combination of traditional finite-element and finite-volume methods. The blending of these methods is the result of simultaneously viewing the element as a control volume and as a domain over which interpolation functions (which are also known as basis functions) may be defined. However, since the DG method is a finite-element method, the order of accuracy and number of unknowns are coupled. DG methods attain high-order accuracy by adding additional basis functions within the elements, which results in additional degrees of freedom for increased orders of accuracy. Alternatively, finite-volume and finite-difference methods reconstruct

high-order data from neighboring elements, which does not increase the total number of degrees of freedom. Therefore, finite-volume and finite-difference methods do not couple the order of accuracy with the number of degrees of freedom. The coupling of the order of accuracy and number of unknowns within an element is a non-trivial property of DG methods, which affects many aspects of solver robustness and hence is a recurring theme throughout this work. However, locating extra unknowns within the elements can be advantageous, provided that great care is taken in constructing and implementing these methods.

As problem size increases, the efficient use of parallel computers becomes more important. DG methods add resolution to a given problem via two approaches. DG methods can add resolution by increasing the number of degrees of freedom within the element, which results in increased parallel efficiency over low-order methods for unstructured grids[1]. By locating the DoFs within the element, higher computational density is achieved and proportionally less inter-element data communication is required. This makes high-order DG methods an ideal candidate for large scale parallel computing. Contrarily, while high-order finite-difference methods have been developed, these methods require the construction of extended interpolation stencils. Extending the interpolation stencil can cause parallel scaling to degrade as the order of accuracy is increased. This degradation of parallel efficiency is a result of the stencils of the grid points on partition boundaries relying on information from multiple data points on neighboring processors. Reference [1] has shown that high-order DG methods have the opposite trend, as the order of accuracy increases the parallel scalability increases as well.

Chapter 2

Finite-element Methods

One of the principal reasons for choosing a discontinuous Galerkin(DG) method to approximate the solution of the partial differential equations is the solid mathematical background upon which these methods site. In particular Galerkin's method can be shown to minimize a particular Sobolev norm of the error. Furthermore, it can be shown that for a particular class of approximations, Galerkin's methods obtains the best approximation. However, the price to be paid for this very solid mathematical theory is the mathematical formalism often used in explaining Galerkin's method, as well as the requirement of functional analysis for proving the stability, consistency and convergence of the numerical scheme.

Discontinuous Galerkin (DG) methods are derived using a technique that is known as Galerkin's method. Galerkin's method is a subset of a more general framework known as the method of weighted residuals (MWR). While the MWR is usually introduced in the context of finite-element methods, it is in fact possible to show that finite-difference, finite-volume, spectral-collocation and a host of other methods can be derived using the MWR. This allows one to study the relationship between these discretization methods and to draw conclusions about the suitability of certain methods for fluid dynamics problems.

2.1 Method of Weighted Residuals

The method of weighted residuals is a general approach for solving differential equations. The approach is to define a space in which one seeks an approximate solution (which is equivalent to assuming a functional form of the solution). We tend to denote this space $\mathcal{V}_h^p$. In order to get a solid grasp of how the method of weighted residuals obtains an approximate solution consider the following one dimensional Poisson problem:

$$\frac{d^2u}{dx^2} + f(x) = 0, \quad x \in (a, b) \tag{2.1.1}$$

subject to the Dirchlet boundary conditions

$$\begin{aligned} u(a) &= g \\ u(b) &= h \end{aligned} \tag{2.1.2}$$

where it is assumed that f is a known scalar valued function on the interval x = [a,b], which is the same as writing

$$f(x) : [a, b] \rightarrow \mathbb{R} \tag{2.1.3}$$

which translated into english reads: "f is a function such that(:) the input in a,b (inclusive) is mapped to some value in the space of real numbers".

While this notation can be tedious at first, once you work with finite-element methods enough it becomes quite natural to write things in this mathematical short hand. Also this formality is not always necessary in an engineering context because engineers often assume that their function f(x) works in a certain space, such as the real numbers, as explained by the physics of the system in question. However, for this very simple example we will be very formal so that the reader can appreciate that care must be taken when applying the MWR to solve differential equations.

The method of weighted residuals seeks an approximate solution of this boundary value problem (which is a differential equation and boundary conditions) by considering an approximate solution $u_h \in \mathcal{V}_h^p$ to $u \in \mathcal{V}$, where $\mathcal{V}_h^p$ is a chosen (and known) approximation space that should be contained inside of $\mathcal{V}$, *i.e.* $\mathcal{V}_h^p \subset \mathcal{V}$. If one where to choose to employ monomials $1, x, x^2, x^3...$ to generate u_h then $\mathcal{V}_h^p$ is the space of monomials of degree p defined on $h : x \in [a, b]$. We will represent the discrete solution in a general form as a finite series summation

$$u_h = \sum_{i=1}^{N} \hat{u}_i \phi_i(x) \tag{2.1.4}$$

where the functions $\phi_i(x)$ are known as basis functions. If one substitutes u_h in place of u in equation (2.1.1), then since u_h does not satisfy equation (2.1.1) there will be a residual or remainder defined by

$$r(x) = \frac{d^2 u_h}{dx^2} + f(x) \neq 0 \tag{2.1.5}$$

The basic idea is to find the set of coefficients $\{\hat{u}_1, \hat{u}_2, ...\hat{u}_N\}$ that give you an approximate solution (hopefully this approximate solution is a good one?). The question is: how does one obtain these coefficients, because so far all we have is one residual statement and N unknown coefficients. This is where the weighting comes into play. The MWR defines the unknown coefficients $\{\hat{u}_1, \hat{u}_2, ...\hat{u}_N\}$ by requiring

$$\int_a^b v_i(x) r(x) dx = 0 \quad \forall i = 1...N \tag{2.1.6}$$

Essentially one picks a set of N approximation functions $\{\phi_i(x)\}$ and N weighting functions $\{v_i(x)\}$ and now one can solve a linear system to obtain the unknown coefficients $\{\hat{u}_i\}$. So far we have left the boundary conditions out of the problem. One can either build these into the basis functions or into the coefficients. While it is not immediately obvious from the mathematics so far, all numerical methods for ODEs/PDEs are the result of picking a certain choice of $v_i(x)$ and $\phi_i(x)$.

2.1.1 Galerkin's Method

Now we are interested in Galerkin's method, which is defined by taking $v_i(x) = \phi_i(x)$:

$$\int_a^b \phi_i(x) r(x) dx = 0 \quad \forall i = 1...N \tag{2.1.7}$$

The resulting discrete equations are given by

$$\int_a^b \phi_i \left(\frac{d^2 u_h}{dx^2} + f \right) dx = \int_a^b \phi_i \sum_{i=1}^{N} \hat{u}_i \frac{d^2 \phi_i}{dx^2} + \phi_i f dx = 0 \quad \forall i = 1...N \tag{2.1.8}$$

from which one can easily see that a system of linear algebraic equations for $\{u_i\}$ is obtained. This statement is sometimes referred to as Galerkin orthogonality because it is essentially defining the residual $r(x)$ such it is orthogonal to the space $\mathcal{V}_h^p$.

While this seems like a particularly simple choice of $v_i(x)$ (which it is), it is not necessarily made because it's simple. In fact this is the best choice of weight functions $v_i(x)$[1]. The best choice of weighting functions would be the functions which generate a discrete solution u_h for all $x \in [a, b]$ which has the smallest norm $\|\cdot\|$ of the error $e_h = u - u_h$. Consider the equation governing the error(obtained by subtracting the discrete equation and the continuous one).

$$\frac{d^2 u}{dx^2} + f - \frac{d^2 u_h}{dx^2} - f = \frac{d^2 (u - u_h)}{dx^2} = \frac{d^2 e_h}{dx^2} = 0 \tag{2.1.9}$$

Taking the inner product over the domain [a,b] of the error equation equation (2.1.9) with the error e_h

$$\int_a^b e_h \frac{d^2 e_h}{dx^2} dx = 0 \tag{2.1.10}$$

integrating by parts and assuming no error at the boundaries (*i.e.* we assume that the approximate solution satisfies the boundary conditions exactly) gives

$$\int_a^b \left(\frac{de_h}{dx} \right)^2 dx = 0 \tag{2.1.11}$$

[1]The term test function is often used in the literature as a synonym for weight function either term is correct.

which one should recognize as the H^1 Sobolev norm

$$\|u\|_{H^1} = \left(\int \left(\frac{du}{dx} \right)^2 dx \right)^{1/2} \tag{2.1.12}$$

of the error e to the second power. Recall that e_h is a function of the coefficients $\{\hat{u}_i\}$. Therefore one can minimize the above Sobolev norm with respect to each of the coefficients $\{\hat{u}_i\}$

$$\min_{\{\hat{u}_i\}} \|e_h\|^2_{H^1} \tag{2.1.13}$$

by differentiating the $\|e_h\|^2_{H^1}$ with respect to each of coefficients $\{\hat{u}_i\}$ one obtains

$$\frac{d\left(\|e_h\|^2_{H^1}\right)}{d\hat{u}_i} = 2\int_a^b - \left(\frac{de_h}{dx} \frac{d\frac{du_h}{dx}}{d\hat{u}_i} dx \right) = 0 \tag{2.1.14}$$

using

$$\frac{d\frac{du_h}{dx}}{d\hat{u}_i} = \frac{d\phi_i}{dx} \tag{2.1.15}$$

results in

$$\frac{d\left(\|e_h\|^2_{H^1}\right)}{d\hat{u}_i} = -\int_a^b \frac{d\phi_i}{dx}\frac{de_h}{dx} = \int_a^b -\frac{d\phi_i}{dx}\left(\frac{du}{dx} - \frac{du_h}{dx} \right) = 0 \tag{2.1.16}$$

which using integration by parts

$$\begin{aligned} &\int_a^b \frac{d\phi_i}{dx}\left(-\frac{du}{dx} + \frac{du_h}{dx} \right) = \int_a^b -\frac{d\phi_i}{dx}\frac{du_h}{dx} + \phi_i f dx = \\ &\int_a^b \phi_i \left(\frac{d^2u_h}{dx^2} + f \right) dx = 0 \end{aligned} \tag{2.1.17}$$

which is what the MWR gave us to begin with. Therefore, Galerkin's method gives the coefficients that minimize the error in the Sobolev norm H^1. Furthermore in can be proven that this is the smallest possible error of all functions in $\mathcal{V}_h^p$.

Propositiion 1. *The discrete solution u_h is closer to the exact solution u than any other function in the space $\mathcal{V}_h^p$ under the H_1 Sobolev norm.*

Proof. Staring with the discrete poisson equation given as

$$\int_a^b v_h \left(\frac{d^2u_h}{dx^2} + f(x) \right) = 0 \quad \forall v_h \in \mathcal{V}_h^p$$

since $\mathcal{V}_h^p \subset \mathcal{V}$ it is equally valid to define the weighted residual form of the poisson equation equation (2.1.1) as

$$\int_a^b v_h \left(\frac{d^2u}{dx^2} + f(x) \right) = 0 \quad \forall v_h \in \mathcal{V}_h^p$$

Subtracting these two equations

$$\int_a^b v_h \frac{d^2\left(u-u_h\right)}{dx^2} = 0 \quad \forall v_h \in \mathcal{V}_h^p$$

which is the equation governing the error $e = u - u_h$. Integration by parts results in the following weak form of the error equation

$$\int_a^b \frac{dv_h}{dx}\frac{d\left(u-u_h\right)}{dx} = 0 \quad \forall v_h \in \mathcal{V}_h^p$$

From here we take the Sobolev norm of the error $u - u_h$

$$\left\| \frac{d\left(u-u_h\right)}{dx} \right\|^2 = \int_a^b \left(\frac{d\left(u-u_h\right)}{dx} \right)^2 dx$$

which can be re-written as

$$\int_a^b \left(\frac{d\left(u-u_h\right)}{dx} \right)\left(\frac{d\left(u-v_h\right)}{dx} \right) - \left(\frac{d\left(u-u_h\right)}{dx} \right)\left(\frac{d\left(u_h-v_h\right)}{dx} \right) dx$$

using the fact that

$$u_h, v_h \in \mathcal{V}_h^p \Rightarrow \left(u_h - v_h\right) \in \mathcal{V}_h^p$$

gives

$$\int_a^b \left(\frac{d\left(u-u_h\right)}{dx} \right)\left(\frac{d\left(u_h-v_h\right)}{dx} \right) dx = 0$$

by Galerkin orthongonality. The remaining part of the norm is given as

$$\left\| \frac{d\left(u-u_h\right)}{dx} \right\|^2 = \int_a^b \left(\frac{d\left(u-u_h\right)}{dx} \right)\left(\frac{d\left(u-v_h\right)}{dx} \right) dx \leq \left\| \frac{d\left(u-u_h\right)}{dx} \right\| \left\| \frac{d\left(u-v_h\right)}{dx} \right\|$$

$$\therefore \quad \left\| \frac{d\left(u-u_h\right)}{dx} \right\| \leq \left\| \frac{d\left(u-v_h\right)}{dx} \right\| \quad \forall v_h \in \mathcal{V}_h^p$$

□

One can see that for an arbitrary function $v_h \in \mathcal{V}_h^p$ we can prove that the error in the H^1 norm is larger than the H^1 norm of $u - u_h$. This is really just a formal proof to show that while one can cast the Galerkin method as a minimization problem, one can also prove that this is the best solution in the approximation space. Therefore, while choosing $v_i = \phi_i$ seems arbitrary and particularly simple at first, in fact this choice is the optimal one in some sense.

The theory of MWR and finite-element methods is heavily based in functional analysis and finite-element methods are often used a motivation for mathematicians to examine certain types of functional analysis problems. Vice versa utilizing finite-element and MWR methods require some basic understanding of functional analysis because it gives confidence in the results and provides reasons for why finite-element methods are defined in a particular fashion.

2.1.2 Briefly on Galerkin's Method and Continuity

From the presentation in the previous section one should be tempted to ask: why integrate by parts? This operation seems somewhat arbitrary and it is, to some extent. However, consider that in order to approximate a second derivative the basis functions ϕ_i must be at least twice differentiable if integration by parts is not performed. This is what is know as a continuity requirement. In this case $\mathcal{C}^1$ continuity is required, which means that the function and it's first derivative must be continuous and differentiable. However, if one integrates by parts once, then only first derivatives of the basis functions are required and the continuity requirement is reduced to $\mathcal{C}^0$, which states that the function itself must be continuous and differentiable (implying that the first derivative exists but is not necessarily differentiable). It is convenient to try to minimize the highest derivative in the differential equation by integrating by parts as many times as is advantageous.

2.1.3 Other MWR Based Methods

So far we have seen the general form of the MWR method as well as the particular flavor known as Galerkin's method. However, one can pick a variety of combinations for the basis and weighting functions. For example if one selects the basis functions ϕ_i to be Lagrange polynomials l_i defined at evenly spaced points in the domain $x \in [a, b]$ and the weight functions to be the Dirac delta function $\delta(x - x_i)$ defined at those same equally spaced points, then one obtained the finite-difference method. Or if one picks the test function as the Heaviside function $H(x_i)$ defined on sub intervals of the domain $x \in [x_l, x_h] \subset [a, b]$ one obtains the so called sub-domain method (which can result in finite-volume and spectral volume-methods). Table 2.1 lists a few methods that are weighted residual methods based on the choice of weight function. Obtaining other well known numerical methods often requires specifying the basis function as well as the weight function, e.g. collocation methods become finite-difference methods when the basis functions are Lagrange polynomials defined at the collocation points.

Table 2.1: Various MWR methods for various choices of weight function

Method	Weight Function
Collocation	$\delta(x - x_i)$
Galerkin	Same as basis ϕ_i
Sub Domain	$H(x - x_i)$
Method of Moments	Monomials $1, x, x^2$.....

2.1.4 A simple example of Galerkin's method for an ODE

Consider the differential equation

$$\frac{d^2u}{dx} + u = x^2 \tag{2.1.18}$$

on the domain $\Omega = x \in [0,1]$ subject to the non-homogenous boundary conditions

$$\begin{aligned} u(0) &= 0 \\ \frac{du}{dx}(1) &= 1 \end{aligned} \tag{2.1.19}$$

Up until now we have left the subject of boundary conditions relatively untouched and assumed that we can take these into account with the basis functions. This problem will provide and example of how to accomplish this task. For this second order differential equation we will consider a two term expansion of the form

$$u_h = \hat{u}_i \left(2x - x^2\right) + \hat{u}_2 \left(x^2 + \frac{2}{3}x^3\right) \tag{2.1.20}$$

Examination of these functions shows that they satisfy the boundary condition at $x = 0$ since $\phi_i(0) = 0, i = 1, 2$. However, these functions do not satisfy the boundary condition at $x = 1$ since $\frac{d\phi_i}{dx}(1) = 0$. Therefore in order we will introduce an extra function $\phi_0 = 0$ to represent this boundary condition. Sometimes in the literature adding boundary conditions in this fashion is known as adding a boundary lifting operator to the expansion. Therefore the approximate solution is given by the modified or lifted expansion

$$u_h = x + \hat{u}_1 \left(2x - x^2\right) + \hat{u}_2 \left(x^2 + \frac{2}{3}x^3\right) \tag{2.1.21}$$

where the function $\phi_0 = x$ does not have an unknown coefficient associated with it, rather the value is already known as 1. Applying Galerkin's method

$$\int_0^1 \phi_i \left(\frac{d^2u_h}{dx^2} + u - x^2\right) dx = 0 \quad i = 1, 2 \tag{2.1.22}$$

Integration by parts

$$\int_0^1 -\frac{d\phi_i}{dx}\frac{du_h}{dx} + \phi_i u - \phi_i x^2 dx = 0 \quad i = 1, 2 \tag{2.1.23}$$

evaluating the integrals for i = 1 and i = 2 results in the following system of equations:

$$\begin{aligned} \frac{4}{5}\hat{u}_1 + \frac{17}{20}\hat{u}_2 &= \frac{7}{60} \\ \frac{17}{90}\hat{u}_1 + \frac{29}{315}\hat{u}_2 &= \frac{1}{36} \end{aligned} \tag{2.1.24}$$

Solving this system gives the unknown coefficients $\{\hat{u}_i\}$:

$$\hat{u}_1 = .154, \quad \hat{u}_2 = 4.877e^{-3} \tag{2.1.25}$$

2.2 Classical Finite-element Methods

Classical finite-element methods (FEM), which are sometimes referred to ask continuous Galerkin (CG) methods are simply a subset of the method of weighted residuals. In this case the domain Ω is broken up into a set of canonical know shapes known as elements such that

$$\Omega = \cup_e \Omega_e \tag{2.2.1}$$

where Ω_e is a single element. In one dimension those elements are called bars or bar elements. In two dimensions the elements are usually triangles or quadrilaterals. However, other shapes are possible provided one can define basis functions that satisfy the requirements of the FEM and physical problem. In three dimensions many element shapes are possible including: tetrahedra, hexahedra, pyramids and prisms.

For a CG method the basis functions, which have compact support (a fancy way of saying the basis functions are non-zero on only one element of the domain) are defined the elements and the residual is weighted with all basis functions in the domain. Classically finite-elements using linear Lagrange polynomials which are value one at the one node in the element. Therefore, each element has two non-zero basis functions. The basis functions are illustrated in Figure 2.1.

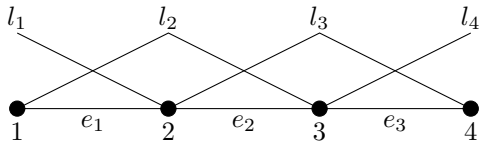

Figure 2.1: Example of linear Lagrange or so-called hat functions defined locally on the elements in a mesh

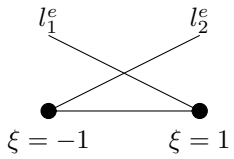

Figure 2.2: Standard element and local basis functions

For an element the basis functions are given by the formulas

$$\begin{aligned} l_1^e &= \frac{1-\xi}{2} \quad \xi \in [-1,1] \\ l_2^e &= \frac{1+\xi}{2} \quad \xi \in [-1.1] \end{aligned} \tag{2.2.2}$$

where these formulas are given in the so called standard element using the standard or parametric coordinate ξ. The standard element depicted in Figure 2.2 is mapped to the physical one using the relation

$$x = l_1^e x_1 + l_2^e x_2 \tag{2.2.3}$$

where x_1 is coordinate at the left end of the element and x_2 is the coordinate at the right end of the element.

Let's consider the Poisson equation in equation (2.1.1) and discretize it using a general finite-element method. Formally we state the finite-element problem as find $u_h \in \mathcal{V}_h$ such that

$$\int_\Omega v_h \left(\frac{d^2 u_h}{dx^2} - f(x) \right) = 0 \quad \forall v_h \in \mathcal{V}_h \tag{2.2.4}$$

While the residual is weighted with each test function and integrated over the domain globally in practice only integrals over the elements need to be computed since most of the basis functions are zero on a particular element. Recalling that the domain Ω is the union of elements Ω_e equation (2.2.4) can be writtten as

$$\sum_e \int_{\Omega_e} v_h \left(\frac{d^2 u_h}{dx^2} - f(x) \right) d\Omega_e = 0 \quad \forall v_h \in \mathcal{V}_h \tag{2.2.5}$$

Applying integration by parts yields:

$$\sum_k \int_{\Omega_e} -\frac{dv_h}{dx}\frac{du_h}{dx} - v_h f(x) d\Omega_e + v_h \frac{du_h}{dx}|_{\xi=-1_k}^{\xi=1} = 0 \quad \forall v_h \in \mathcal{V}_h \tag{2.2.6}$$

where the term evaluated at $\xi = -1, 1$ is evaluated at the element boundaries. The term evaluated at the element boundaries normally cancels out due to the continuity of the basis functions. To see how this is accomplished consider a collection of elements donated $k = 1, 2, 3$ in Figure 2.1. One now selects the v_h as l_2. The summation over elements in equation (2.2.6) becomes

$$\begin{aligned} &\sum_k \int_{\Omega_e} -\frac{dl_2}{dx}\frac{du_h}{dx} - l_2 f(x) d\Omega_e + l_2 \frac{du_h}{dx}|_{x_1}^{x_2} = \\ &\int_{\Omega_1} -\frac{dl_2}{dx}\frac{du_h}{dx} - l_2 f(x) d\Omega_e + l_2 \frac{du_h}{dx}|_{x_1}^{x_2} + \\ &\int_{\Omega_2} -\frac{dl_2}{dx}\frac{du_h}{dx} - l_2 f(x) d\Omega_e + l_2 \frac{du_h}{dx}|_{x_2}^{x_3} \end{aligned} \tag{2.2.7}$$

Since we are using a Galerkin method the form of u_h is given as

$$u_h = \sum_j \hat{u}_j l_j(x) \tag{2.2.8}$$

where $j = 1, 2$ locally on each element. Using the known form of u_h results in

$$\int_{\Omega_1} -\frac{dl_2}{dx}\frac{du_h}{dx} - l_2 f(x) d\Omega_1 + 1\left(\hat{u}_1\frac{dl_1}{dx} + \hat{u}_2\frac{dl_2}{dx}\right) - 0\left(\hat{u}_1\frac{dl_1}{dx} + \hat{u}_2\frac{dl_2}{dx}\right) +$$
$$\int_{\Omega_2} -\frac{dl_2}{dx}\frac{du_h}{dx} - l_2 f(x) d\Omega_2 + 0\left(\hat{u}_2\frac{dl_2}{dx} + \hat{u}_3\frac{dl_3}{dx}\right) - 1\left(\hat{u}_2\frac{dl_2}{dx} + \hat{u}_3\frac{dl_3}{dx}\right) \tag{2.2.9}$$

where we have simply employed the property of the basis functions:

$$l_j(x_i) = \delta_{ij} \tag{2.2.10}$$

This type of patter is repeated from element to element through the domain. As the integrals are performed the boundary terms cancel out. This canceling out is due to the continuity of the basis functions l_j. The method of forming the residual at a node (node 2 in this case) is used as a demonstration of how the inter-element boundary terms cancel out formally.

2.2.1 Implementation

In practice one does not normally go through the formal crossing out of the inter element boundary contributions to the discrete residual. In practice these surface terms are ignored and one forms integrals over the elements and then added the contributions the nodes that make up the element. This gives rise to the so-called element matrices that are so common in textbooks on FEM. Continuing with the poisson equation example the two element residual equations for element $k = 2$ are

$$\begin{aligned} &\int_{\Omega_2} -\frac{dl_2}{dx}\frac{du_h}{dx} - l_2 f(x) d\Omega_2 \\ &\int_{\Omega_2} -\frac{dl_3}{dx}\frac{du_h}{dx} - l_3 f(x) d\Omega_2 \end{aligned} \tag{2.2.11}$$

if one recalls that

$$u_h = \sum_j \hat{u}_j l_j(x) \tag{2.2.12}$$

then the equations for an element are written in matrix form as

$$\begin{bmatrix} -\int_{\Omega_2}\frac{dl_2}{dx}\frac{dl_2}{dx}dx & -\int_{\Omega_2}\frac{dl_2}{dx}\frac{dl_3}{dx}dx \\ -\int_{\Omega_2}\frac{dl_3}{dx}\frac{dl_2}{dx}dx & -\int_{\Omega_2}\frac{dl_3}{dx}\frac{dl_3}{dx}dx \end{bmatrix} \begin{Bmatrix} \hat{u}_2 \\ \hat{u}_3 \end{Bmatrix} = \begin{Bmatrix} \int_{\Omega_2} l_2 f(x) dx \\ \int_{\Omega_2} l_3 f(x) dx \end{Bmatrix} \tag{2.2.13}$$

The matrix in equation (2.2.13) is know as the element stiffness matrix. One should immediately note that element 2 in this case involves the unknowns as nodes 2 and 3. However these nodes also have contributions from other elements. Recall that in reality we should have written the equations on a

per node basis since these are the unknowns for this particular finite-element approach. However, we can properly recover the equations for each node by properly combining the element element matrix equations using a process known as global matrix assembly. One should also be aware that both the element and global matrices are often referred to as stiffness matrices in the literature. This nomenclature is taken from the structural analysis fields where continuous Galerkin methods were first applied.

Global Matrix assembly

In one spatial dimension it is easy to demonstrate how the global matrix assembled from the local element matrices. The matrix equations in equation (2.2.13) are in fact just the local equations for element 2. These equations can be written symbolically as

$$[k^e]\{\hat{u}^e\} = \{f^e\} \tag{2.2.14}$$

for each element e in the mesh. Examination of the finite-element mesh in Figure 2.1 gives an indication of how the global matrix assembly works. In this case each node is connected to two elements. Therefore each node's diagonal matrix entry will be the sum of contributions from both element's connected to it. There will be no off-diagonal addition entries because the elements are only connected by nodes. If this were a 2-D triangular mesh the entries of the matrix that are additions from various elements would be more complicated since elements are connected by edges in this higher dimensional space. The assembly of the global system for the example mesh in Figure 2.1 is illustrated as

$$\begin{bmatrix} k_{11}^1 & k_{12}^1 & 0 & 0 \\ k_{21}^1 & k_{22}^1 + k_{11}^2 & k_{12}^2 & 0 \\ 0 & k_{21}^2 & k_{22}^2 + k_{11}^3 & k_{12}^3 \\ 0 & 0 & k_{21}^3 & k_{22}^3 \end{bmatrix} \begin{Bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \\ \hat{u}_4 \end{Bmatrix} = \begin{Bmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_1^3 \end{Bmatrix} \tag{2.2.15}$$

Essentially this indexing of the global matrix takes advantage of what is know as the mapping from the local node number to the global node number. If one considers an element say element 2 in the global mesh and we observe that local

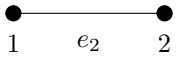

Figure 2.3: Local and global node numbering

node 1 in this case is global node 2 and local node 2 is global node 3.

Each element in the mesh has 2 nodes denoted 1 and 2 and these nodes have a corresponding global node number. The element stiffness matrix entries indexed as k_{ij}^e reflect the local number numbering on each element and then are added to the rows and columns of the global matrix by considering the global node numbers of the nodes belonging to each element. The i, j in the local

element stiffness matrices can be considered as the effect of local node j on the solution at node i. So in the case of element 2 we take the element stiffness matrix $[k^e]$ and add it the global system $[k]$ as:

$$\begin{aligned} k_{22} &+= k^2_{11} \\ k_{23} &+= k^2_{12} \\ k_{32} &+= k^2_{21} \\ k_{33} &+= k^2_{22} \end{aligned} \tag{2.2.16}$$

For more complicated meshes there exists a formal assembly algorithm for 2/3-D meshes employing linear elements. One could in theory develop such an algorithm for any type of continuous finite-element discretization.

Chapter 3

Mesh Data Structures

This chapter describes the mesh data structures that are implemented as part of the class UnstGrid. **UnstGrid** is a C++ class defined in the file UnstGrid.h. If you're interested in the source documentation please see the source code manual.

3.1 Introduction

For the purposes of this document a mesh denotes a collection of nodes and elements. nodes define points in physical space and elements describe how these nodes are connected. The FEM library only supports certain types elements i.e. 1-D lines, 2-D triangles, 2-D quadrilaterals, 3-D tetrahedra, 3-D prisms, and 3-D Hexahedra. First we describe how elements are defined and how they are combined to form unstructured meshes. Then with the theory how meshes are defined in hand we will describe the data structures used in the code to define them and how to acess the data of the mesh using the implemented class functions of **UnstGrid**.

Note that mesh generation is a topic unto itself and the goal of this section is to describe how the numerical integration library interacts with a pre-existing mesh. A pre-existing mesh is one where the nodal coordinates in the physical space (i.e. the one in which we wish to solve the PDE's) represented notionally by the coordinates (x, y, z) are known. Additionally a mesh must supply how these nodes are connected into elements along with how the boundary faces of the domain are connected to the nodes that lie on the domain boundary. A few conditions on the mesh are that is must be completely fill the domain of interest Ω, mathematicaly this is

$$\Omega = \bigcup_e \Omega_e \tag{3.1.1}$$

3.2 Element Types

This section describes the types of elements that are supported by the solver library. In general elements define how nodes in the mesh are connected to each other. Elements are normally defineds in a know standard space defined by greek letters ξ, η, ζ. The positions of the nodes of the elements are known in this standard space and the physical locations of the nodes in (x, y, z) spaces are determined by a mapping function $f_m : \mathbb{R}^d \rightarrow \mathbb{R}^d$ where d is the number of physical dimensions of the domain. In this work the mapping functions f_m take on the form of polynomial interpolations, that are nodally exact at all nodes of the element.

3.2.1 1-D Bar

In one spatial dimension only one type of element is available. This element in known as the "bar" element. The "bar" element (shown in Figure 3.1) is constructured by taking two nodes of the mesh and connecting them by a line segment.

Figure 3.1: 1-D "bar" element.

Recalling that each element has a definition in the standard space of equivalent dimension the coordinates of the nodes are node_1: $\xi = -1$ and node_2: $\xi = 1$ in the standard space spanning $\xi \in (-1, 1)$.

A 1-D "bar" element may have any number of solution unknowns associated with it but it requires a minimum of two uknowns: one for each element of the element. For finite-element methods this results in a linear data representation and an asymptotically second-order accurate solution.

3.2.2 2-D Triangle

In two spatial dimensions there are two element types available one of which is the triangle. The triangle is defined by connecting 3 of the mesh nodes as in Figure **??**.

3.3 Mesh Connectivity

The mesh connectivity data is stored in a linked list structure to facilitate optimal memory footprint. Each connectivity of the mesh is stored as a separate linked list. For example the nodes belonging to an element is one mesh connectivity and is stored in one linked list while the elements surrounding a node is stored in yet another linked list. Each linked list is comprised of two arrays an

"index" array and a "data" array. The index array is normally named with an "i" at the end of the variable name and is composed on integer.

As an example consider the connectivity defining the node numbers of each element in a mesh. To make the explanation concrete consider the simple mesh in Figure 3.2. One should note that this 2-D example contains a mesh of mostly

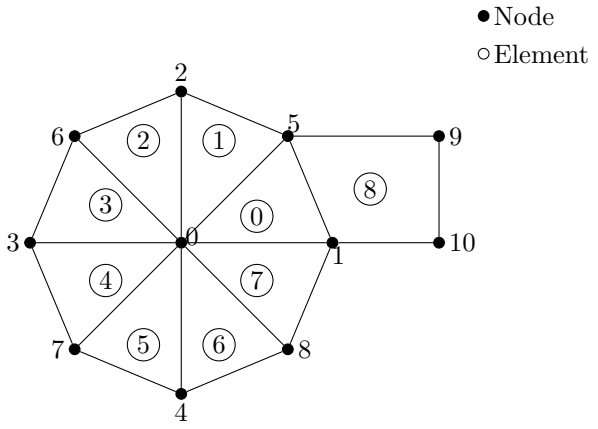

Figure 3.2: Sample Mesh.

triangles with one quadrilateral. The mixed element nature of the mesh prevents the mesh connectivity taking the form of a table (which can be stored in the computer a 2-D array) because some rows need more columns than others. For example all rows of the table corrsponding to triangular elements require 3 columns and rows corresponding to quadrilaterals require 4 columns. Clearly a more elegant data structure is required. In this work a linked list is employed to facilitate the mesh connectivity storage.

Let *element2node* denote the data array that stores the nodes that make up each element and *element2nodei* denote the index array for *elem2node*. The size of *element2node* is the sum of the nodes attached to each element over all elements of the mesh and the size of *element2nodei* is the number of elements + 1. For an element e the nodes that makeup e are stored in the following locations

$$\begin{aligned} &is = elem2nodei[e] \\ &ie = elem2nodei[e+1] - 1 \\ &elem2node[is : ie] : \text{is where the nodes for element } e \text{ are stored} \end{aligned} \tag{3.3.1}$$

In order to make this concrete consider the sample mesh in Figure 3.2. For this mesh the arrays *element2nodei* and *element2node* take the following values:

	0	1	2	3	4	5	6	7	8	9
element2nodei:	0	3	6	9	12	15	18	21	24	28

	0-2	3-5	6-8	9-11	12-14	15-17	18-20	21-23	24-27
element2node:	0 1 5	0 5 2	0 2 6	0 6 3	0 3 7	0 7 4	0 4 8	0 8 1	1 10 9 5

Bibliography

[1] Cristian R. Nastase and Dimitri J. Mavriplis. A parallel hp-multigrid solver for three-dimensional discontinuous galerkin discretizations of the euler equations. In *Proceedings of 45th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2007. AIAA Paper 2007–0512.