

FINITE ELEMENT ANALYSIS

ARDEN RASMUSSEN

APRIL 14, 2019

ABSTRACT. Finite Element Analysis (FEA) is the computational process prescribed by Finite Element Methods (FEM) in order to solve for numerical solutions to boundary value problems for partial differential equations. Finite Element Analysis is used in many specializations of mechanical engineering, and is used frequently in that discipline, and occasionally in other disciplines as well. This paper provides an introduction to the mathematics of Finite Element Methods, and why they are useful. Then, it provides and explains the simplistic implementation of Finite Element Analysis. The implementation provided with this paper is written in C++, and is commented for better readability of the code.

1. INTRODUCTION

Finite element methods is a computational process used for the determination of an approximate solution to a given partial differential equation, with given forcing term and boundary conditions. Finite element analysis is the practical implementation of finite element methods computationally on a physical problem in order to ascertain more insight into the physical system in question. We will first explain finite element methods, and provide a thorough introduction into the mathematics behind the methods. Then we will also provide an explanation of finite element analysis, demonstrating the mathematics of FEM with respect to an example problem, and develop a computational program that we will use to solve the partial differential equation using FEM.

Finite element methods is very useful, due to the fact that there are only a few situations where a solution to a partial differential equation can be found analytically, so we need methods to find a close approximation of the actual solution. Currently there are

a handful of methods that can be used for this purpose, however finite element methods are currently the best methods that we know of. Meaning that the error between the analytical solution and the solution constructed by finite element analysis converges to zero faster than other methods.

The process that we utilize for constructing the finite element methods is called the Galerkin method, and is presented by [KH15]. We will closely follow the method outlined by their paper. For this introduction we will limit the scope to two dimensions, but the mathematics can be easily generalized to higher dimensions, and we will make note of the sections that would have to change for higher dimension problems.

We divide this paper into two major parts. Part 1 explains the process of FEM, and demonstrates the mathematics and the concepts behind the method, and how it can be used to solve the partial differential equations. Part 2 explains the process of FEA, and discusses the algorithms, data structures, and optimizations that can be done to construct a program to efficiently compute the solution utilizing the previously discussed FEM.

Part 1. Mathematics

2. PROBLEM STATEMENT

Throughout this paper and our sample code, we implement finite element methods using the general form of the heat equation as our differential equation to consider. We define the domain to be Ω such that $\Omega \in \mathbb{R}^2$ is a bounded 2D domain with boundary Γ .

The general form of the heat equation takes the form

$$\frac{\partial u}{\partial t} = Lu + f,$$

where L is an operator of the form

$$L = \sum_{\alpha, \beta=1}^2 A^{\alpha\beta} \frac{\partial}{\partial x^\alpha} \frac{\partial}{\partial x^\beta} + \sum_{\alpha=1}^2 B^\alpha \frac{\partial}{\partial x^\alpha} + C$$

and f is some forcing function. We will also enforce that A is a positive definite matrix. And where u is a function that takes the position and the time and returns a real $u(\vec{x}, t) : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$. Note that $\vec{x} \in \Omega$, and $t \in \mathbb{R}$.

Rewriting this expression into a single equation we can express our problem statement as

$$(2.1) \quad \frac{\partial u}{\partial t} = \sum_{\alpha, \beta=1}^2 A^{ij} \frac{\partial}{\partial x^\alpha} \left[\frac{\partial u}{\partial x^\beta} \right] + \sum_{\gamma=1}^2 B^\gamma \frac{\partial u}{\partial x^\gamma} + Cu + f$$

Note that for this process to function, we must also require some conditions. We first need the boundary conditions in order to determine any solution. Therefore we require that $u(\partial\Omega, t)$ is provided. Note that this could mean that the boundary is split into several sections, each one with different conditions, but the value of u at any point along the boundary must be known.

If considering the more general time dependent form of the equation we are also required to know initial conditions. That is to say that $u(\Omega, 0)$ must also be provided. Using the initial conditions and the boundary conditions, we are able to construct an approximation through the use of finite element methods.

Given A, B, C, f , and appropriate conditions, we must construct a solution for u . This is the key purpose of finite element methods, as for most circumstances there does not exist an analytical solution

to this differential equation. Thus we must utilize finite element analysis in order to construct the best possible approximation of the solution to the differential equation, and then it is possible to use that approximation for further analysis.

3. CONDITIONS

For our proposed problem statement, we will require some conditions in order to make it possible to determine a solution to the problem. The conditions that we will require are boundary conditions and if we are considering a time dependent system, we will also need an initial condition.

3.1. Boundary Conditions. We must prescribe boundary conditions on our problem statement, to ensure that there is a single solution. For now we will only proceed utilizing Dirichlet boundary conditions. It should be noted that other conditions are possible with this method, such as Neumann, and Newton boundary conditions. These other boundary conditions require more changes to the formulation of the problem later on and require extra modifications to the implementation. Because of the extra alterations that they would necessitate, we will only focus on the Dirichlet boundary conditions.

For the Dirichlet boundary conditions we must define the value of our solution at every point on the boundary. We define this as

$$\partial u(\vec{x}, t) \quad \vec{x} \in \partial\Omega.$$

We will use this notation to denote the function defining the values of the solution to the problem along the boundary at time t . This can be written as

$$u(\partial\Omega, t) \equiv \partial u(\partial\Omega, t).$$

It is important to note that there are no restrictions to the form of ∂u . The user defined function can take any form, and thus requiring the solution to also fit any form of boundary conditions.

3.2. Initial Conditions. If the problem being considered were to be time dependent, then it is also necessary to define an initial condition to the problem. That is to say that it is required to provide some function that defines the value of u at $t = 0$. We define this function as

$$u_0(\vec{x}) \quad \vec{x} \in \Omega.$$

We use this notation to denote the function defining the values of the solution to the problem, that is to say

$$u(\Omega, t = 0) \equiv u_0(\Omega).$$

Note that this function also has no restrictions other than that it must be defined on the entirety of the domain Ω , although it need not be continuous on the domain.

If the problem being considered is not time dependent, then the initial condition can be ignored, and only the boundary conditions must be specified.

Using these prescribed conditions, both the boundary and the initial conditions, it is now possible to utilize finite element methods to attain an approximation of the solution to the problem statement.

4. VARIATIONAL FORMULATION

In order to construct the variational formulation of our problem statement, we must first introduce the concept of a residual. The residual is the construction of a function R such that when our approximation is the exact solution then $R = 0$. Using the residual we now want to construct a function that minimizes the value of the residual. We will construct the residual of equation 2.1 as

$$R(\vec{x}, t) = \frac{\partial u}{\partial t} - Lu - f$$

Clearly by the construction of R , $R(\vec{x}, t) = 0$ if and only if u is the exact solution to the partial differential equation. Now we multiply the residual by some arbitrary test function, and integrate over our domain

$$\int_{\Omega} w(\vec{x}) R(\vec{x}, t) d\vec{x}.$$

For the exact solution u this will always be zero, because $R(\vec{x}, t) = 0$ for all $\vec{x} \in \Omega$. Because of this we are able to select any test function w , and the weighted residual will still be zero, given the exact solution. However, we will restrict our test functions to ones that are zero along the boundaries $\partial\Omega$, this will significantly simplify our computations in the future.

We now expand the weighted residual to obtain

$$\int_{\Omega} w \frac{\partial u}{\partial t} d\vec{x} = \int_{\Omega} w L u d\vec{x} + \int_{\Omega} w f d\vec{x}.$$

Expanding our L operator on u and splitting the integral over the sum, we find

$$\begin{aligned} \int_{\Omega} w \frac{\partial u}{\partial t} d\vec{x} &= \sum_{\alpha, \beta=1}^2 \int_{\Omega} w A^{\alpha\beta} \frac{\partial}{\partial x^{\alpha}} \left[\frac{\partial u}{\partial x^{\beta}} \right] d\vec{x} \\ &+ \sum_{\gamma=1}^2 \int_{\Omega} w B^{\gamma} \frac{\partial u}{\partial x^{\gamma}} d\vec{x} + \int_{\Omega} w C u d\vec{x} + \int_{\Omega} w f d\vec{x}. \end{aligned}$$

We notice that w is always independent of time so that the first integral can be rewritten as

$$\int_{\Omega} w \frac{\partial u}{\partial t} d\vec{x} = \frac{\partial}{\partial t} \int_{\Omega} w u d\vec{x}.$$

We can now apply integration by parts on the second integral, and rewrite this as

$$\begin{aligned} \int_{\Omega} w A^{\alpha\beta} \frac{\partial}{\partial x^{\alpha}} \left[\frac{\partial u}{\partial x^{\beta}} \right] d\vec{x} &= \int_{\partial\Omega} A^{\alpha\beta} w \frac{\partial u}{\partial x^{\beta}} d\vec{x} \\ &- \int_{\Omega} \frac{\partial A^{\alpha\beta} w}{\partial x^{\alpha}} \frac{\partial u}{\partial x^{\beta}} d\vec{x}. \end{aligned}$$

Since our restriction on the selection of test functions w , we know that $w(\vec{x}) = 0 \forall \vec{x} \in \partial\Omega$. Thus the integral over the boundary must be zero. Using this simplification, we obtain the representation of the continuous variational formulation of the problem to be

$$\begin{aligned} (4.1) \quad \frac{\partial}{\partial t} \int_{\Omega} w u d\vec{x} &= - \sum_{\alpha, \beta=1}^2 \int_{\Omega} \frac{\partial A^{\alpha\beta} w}{\partial x^{\alpha}} \frac{\partial u}{\partial x^{\beta}} d\vec{x} \\ &+ \sum_{\gamma=1}^2 \int_{\Omega} B^{\gamma} w \frac{\partial u}{\partial x^{\gamma}} d\vec{x} + \int_{\Omega} C w u d\vec{x} + \int_{\Omega} w f d\vec{x}. \end{aligned}$$

5. DISCRETIZATION

Now that we have a variational formulation of the partial differential equation, we wish to discretize the equation, so that it is possible to apply the method of finite elements.

In order to discretize the problem, we must have an understanding of the space that the solution resides in. It is clear that the solution to the equation will be in the Sobolev space $W^{1,2}(\Omega)$. The Sobolev space is the set of functions that are infinitely differentiable. In this case we can tell that our solution resides in the specific Sobolev space $W^{1,2}(\Omega)$. It is also important to mention that this can be written as $H^1(\Omega)$, because this Sobolev space is also a Hilbert space. A Hilbert space is one that possesses the structure for inner product. Thus we are able to conclude that our solution exists in a space that supports inner product, and is infinitely differentiable.

Since the process of finite element methods is inherently intended for computation, problems arise when infinities come into play. Since the span of the Sobolev space $H^1(\Omega)$ is infinite, we are required to consider a subspace of it. This will be the subspace of the Sobolev space that our approximation of the solution lives in, we will call this subspace $\tilde{V} \subset H^1(\Omega)$.

We will construct a finite set of basis functions for the subspace \tilde{V} to be $\{\varphi_j\}$. For now we will leave these basis functions as arbitrary functions, but the

process for the construction of the basis functions will be presented in a later section.

The approximation of u we define as u , since the approximation should converge to the exact solution, utilizing the same notation does not cause any issues. We construct this approximation such that it is an element of the subspace of the Sobolev space as we have defined. That is to say

$$u \in \tilde{V} \subset H^1(\Omega)$$

Since our approximation is constructed in \tilde{V} , then it must be expressible as a linear combination of the basis function φ_j of the finite subspace \tilde{V} that \tilde{u} is an element of. Thus we can express \tilde{u} as

$$(5.1) \quad u = \sum_{j=1}^N u_j(t) \varphi_j(\vec{x}),$$

where $u_j(t)$ is a function of time that once it is found, it is possible to construct the final approximation. Using this expression for our approximation the goal of the method is to determine the functions $u_j(t)$.

We can now substitute our expression for u into the continuous variational formulation of the problem (4.1).

$$(5.2) \quad \begin{aligned} \frac{\partial}{\partial t} \int_{\Omega} w \left[\sum_{j=1}^N u_j \varphi_j \right] d\vec{x} = & \\ - \sum_{\alpha, \beta=1}^2 \int_{\Omega} \frac{\partial A^{\alpha\beta} w}{\partial x^{\alpha}} \frac{\partial}{\partial x^{\beta}} \left[\sum_{j=1}^N u_j \varphi_j \right] d\vec{x} & \\ + \sum_{\gamma=1}^2 \int_{\Omega} B^{\gamma} w \frac{\partial}{\partial x^{\gamma}} \left[\sum_{j=1}^N u_j \varphi_j \right] d\vec{x} & \\ + \int_{\Omega} C w \left[\sum_{j=1}^N u_j \pi_j \right] d\vec{x} + \int_{\Omega} w f d\vec{x}. & \end{aligned}$$

It is possible to rewrite this expression, with several simplifications. First is to note that since w and φ_j are independent of t , and u is independent of \vec{x} . Using this it is possible to rewrite the first term, by moving constants with respect to the integral (u_j) out of the integral, and then to move constants with respect to the derivative (the integral) out of the derivative. Thus we can write the first term as

$$\frac{\partial}{\partial t} \int_{\Omega} w \left[\sum_{j=1}^N u_j \varphi_j \right] d\vec{x} = \sum_{j=1}^N \left[\frac{\partial u_j}{\partial t} \int_{\Omega} w \varphi_j d\vec{x} \right].$$

Similar process can be applied to the other terms in equation 5.2. By pulling the constants out of the integrals, and splitting the integral over the summation,

we are able to rewrite the expression as

$$(5.3) \quad \begin{aligned} \sum_{j=1}^N \left[\frac{\partial u_j}{\partial t} \int_{\Omega} w \varphi_j d\vec{x} \right] = & \\ - \sum_{j=1}^N \left[u_j \sum_{\alpha, \beta=1}^2 \int_{\Omega} \frac{\partial A^{\alpha\beta} w}{\partial x^{\alpha}} \frac{\partial \varphi_j}{\partial x^{\beta}} d\vec{x} \right] & \\ + \sum_{j=1}^N \left[u_j \sum_{\gamma=1}^2 \int_{\Omega} B^{\gamma} w \frac{\partial \varphi_j}{\partial x^{\gamma}} d\vec{x} \right] & \\ + \sum_{j=1}^N \left[u_j \int_{\Omega} C w \varphi_j d\vec{x} \right] + \int_{\Omega} w f d\vec{x}. & \end{aligned}$$

Now, since we have not placed any restrictions on w , and it can be any function, we will chose the basis functions φ_i as our test functions. Substituting that into the expression we obtain the discretized version of the variational formulation of the problem.

$$(5.4) \quad \begin{aligned} \sum_{j=1}^N \left[\frac{\partial u_j}{\partial t} \int_{\Omega} \varphi_i \varphi_j d\vec{x} \right] = & \\ - \sum_{j=1}^N \left[u_j \sum_{\alpha, \beta=1}^2 \int_{\Omega} \frac{\partial A^{\alpha\beta} \varphi_i}{\partial x^{\alpha}} \frac{\partial \varphi_j}{\partial x^{\beta}} d\vec{x} \right] & \\ + \sum_{j=1}^N \left[u_j \sum_{\gamma=1}^2 \int_{\Omega} B^{\gamma} \varphi_i \frac{\partial \varphi_j}{\partial x^{\gamma}} d\vec{x} \right] & \\ + \sum_{j=1}^N \left[u_j \int_{\Omega} C \varphi_i \varphi_j d\vec{x} \right] + \int_{\Omega} \varphi_i f d\vec{x}. & \end{aligned}$$

It can be useful to examine this equation using the inner products as they act upon functions. That is to say, that we use the L^2 inner product defined as

$$\langle f | g \rangle_{L^2} = \int_{\Omega} f g d\vec{x}.$$

Using this notation, the expression can be rewritten to be

$$\begin{aligned} \sum_{j=1}^N \langle \varphi_i | \varphi_j \rangle \frac{\partial u_j}{\partial t} = & - \sum_{j=1}^N \sum_{\alpha, \beta=1}^2 \left\langle \frac{\partial}{\partial x^{\alpha}} [A^{\alpha\beta} \varphi_i] \left| \frac{\partial \varphi_j}{\partial x^{\beta}} \right. \right\rangle u_j \\ & + \sum_{j=1}^N \sum_{\gamma=1}^2 \left\langle B^{\gamma} \varphi_i \left| \frac{\partial \varphi_j}{\partial x^{\gamma}} \right. \right\rangle u_j \\ & + \sum_{j=1}^N \langle C \varphi_i | \varphi_j \rangle u_j + \langle \varphi_i | f \rangle. \end{aligned}$$

It is now possible to factor out the u_j from the right hand side summations. This will provide the equation

$$\sum_{j=1}^N \langle \varphi_i | \varphi_j \rangle \frac{\partial u_j}{\partial t} = \left[- \sum_{j=1}^N \sum_{\alpha, \beta=1}^2 \left\langle \frac{\partial}{\partial x^\alpha} [A^{\alpha\beta} \varphi_i] \middle| \frac{\partial \varphi_j}{\partial x^\beta} \right\rangle + \sum_{j=1}^N \sum_{\gamma=1}^2 \left\langle B^\gamma \varphi_i \middle| \frac{\partial \varphi_j}{\partial x^\gamma} \right\rangle + \sum_{j=1}^N \langle C \varphi_i | \varphi_j \rangle \right] u_j + \langle \varphi_i | f \rangle.$$

Since it is possible to replace w with *any* of the basis functions, that means that $i = 1, 2, \dots, N$. This provides us with a system of equations, which is then expressible in terms of matrices. The matrix representation of the system is

$$(5.5) \quad G \partial_t U = (\bar{A} + \bar{B} + \bar{C}) U + F.$$

Where U is a vector in \mathbb{R}^N composed of the u_j functions. We use the notation ∂_t to denote the operator of taking the partial derivative of the U vector with respect to time, element wise. The elements of \bar{A} , \bar{B} , \bar{C} , and F are defined as

$$\begin{aligned} \bar{A}^{ij} &= - \sum_{\alpha, \beta=1}^2 \left\langle \frac{\partial}{\partial x^\alpha} [A^{\alpha\beta} \varphi_i] \middle| \frac{\partial \varphi_j}{\partial x^\beta} \right\rangle & i, j = 1, \dots, N \\ \bar{B}^{ij} &= \sum_{\gamma=1}^2 \left\langle B^\gamma \varphi_i \middle| \frac{\partial \varphi_j}{\partial x^\gamma} \right\rangle & i, j = 1, \dots, N \\ \bar{C}^{ij} &= \langle C \varphi_i | \varphi_j \rangle & i, j = 1, \dots, N \\ F &= \langle \varphi_i | f \rangle & i, j = 1, \dots, N. \end{aligned}$$

It is often simpler to consider a matrix M such that $M = \bar{A} + \bar{B} + \bar{C}$, then our expression for the matrix form of the system of equations becomes

$$(5.6) \quad G \partial_t U = MU + F.$$

This matrix representation is how we will continue to consider the problem. For most cases this is the system of equations that will be found. However, this expression is only sufficient for Dirichlet boundary conditions, as the integral over the boundary for other boundary conditions would not be zero. Thus there would be additional terms in the system of equations. This is not an issue with the Dirichlet boundary conditions, so we do not need to consider any additional terms in the matrix representation.

If the problem that is being considered were to be time independent, then the matrix formulation of the

system of equations would be

$$MU = F.$$

We will continue to consider both the time dependent and the time independent problems, as both are useful in different situations. However, they are solved in different ways, some simplifications can be made with one but not the other.

6. LOCALIZATION

While it is occasionally possible to find the elements of the matrices in equation 5.6 on the global domain, this proves to be exceedingly difficult in most if not all cases. To remediate this issue, we implement a generalizable method that will work for any domain and functions that are being considered. Since we have the ability to select any arbitrary basis functions as per our definition of the approximation to the solution, then it is possible to select a basis such that it becomes possible to split the domain into a finite number of subdomains.

We first discretize the domain Ω into N mutually exclusive subdomains. Each of these subdomains is called an *element*. For the purposes of this paper, each element will be defined to be a triangle subdomain of Ω . The set of all elements is called the triangular decomposition or the mesh of the domain, the process for constructing the triangular decomposition is discussed further in section ???. For the localization process, the specifics of the triangular decomposition is not necessary, so we will not go in depth on the structure of the triangular decomposition in this section.

For each element of the mesh, we denote the element as $E^{(e)}$, where $e = 1, \dots, N$. Then each vertex of the triangle is labeled as $E_i^{(e)}$ where $i = 1, 2, 3$. The notation for the x and y values of each vertex we denote as $X_i^{(e)}$ and $Y_i^{(e)}$ respectively, where $i = 1, 2, 3$.

In 1D the process is trivial as all elements have the same dimensions, but in higher dimensional implementation, each element can have variable shapes and sizes. This becomes more difficult because integrating over each element would require entirely different process each time to account for the differences in shape and size. To resolve this issue, we construct a master space, which is possible to transition to from any element, and back to each element. Thus we only need determine a method to integrate over our master element.

6.1. Master Space. The master space is a domain that is constructed in order to simplify the construction of the local approximations. For the situation with triangular elements, we construct the master

space such that the vertices of a triangle lie on the points $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$. Then for each element in the mesh, we construct a transformation between the master space and the element space.

We define the master element to be \hat{E} . An image of the master element can be seen in figure ??.

The master element utilizes what is called a Barycentric coordinate system. The barycentric coordinates are demonstrated by each vertex being its own dimension, so if the element were to be quadrangles, then the barycentric coordinates would be in \mathbb{R}^4 instead of \mathbb{R}^3 as they are for triangles. The conversion between the global coordinates, and the Barycentric coordinates can be done using the following equations

$$\begin{aligned}\lambda_1 &= \frac{(Y_2 - Y_3)(x - X_3) + (X_3 - X_2)(y - Y_3)}{(Y_2 - Y_3)(X_1 - X_3) + (X_3 - X_2)(Y_1 - Y_3)} \\ \lambda_2 &= \frac{(Y_3 - Y_1)(x - X_3) + (X_1 - X_3)(y - Y_3)}{(Y_2 - Y_3)(X_1 - X_3) + (X_3 - X_2)(Y_1 - Y_3)} \\ \lambda_3 &= 1 - \lambda_1 - \lambda_2.\end{aligned}$$

Where X_i and Y_i represent the x and y coordinates of the i th vertex of the element. x , y are the coordinates of the point in the triangle that is to be converted to the Barycentric coordinates. Converting from Barycentric coordinates to the global coordinates uses the following equations

$$\begin{aligned}x &= \lambda_1 X_1 + \lambda_2 X_2 + \lambda_3 X_3 \\ Y &= \lambda_1 Y_1 + \lambda_2 Y_2 + \lambda_3 Y_3\end{aligned}$$

6.2. Local Basis. The first step is to construct a local basis function, that are only defined for the current element under consideration. There should be a basis element for every vertex in the element, so for triangular element, there will be three local basis functions. The notation for the local basis functions is $\varphi_1^{(e)}$, $\varphi_2^{(e)}$, and $\varphi_3^{(e)}$. To make computation simple, the construct the basis function such that they are linear, with the requirement that at the corresponding vertex, the local basis function is 1 and at all other vertices of the element the basis function will be 0.

The conversion from global space to the bilinear coordinates does exactly this task. Thus the definition of the local basis functions are

$$\begin{aligned}(6.1) \quad \varphi_1^{(e)} &= \frac{(Y_2^{(e)} - Y_3^{(e)})(x - X_3^{(e)}) + (X_3^{(e)} - X_2^{(e)})(y - Y_3^{(e)})}{(Y_2^{(e)} - Y_3^{(e)})(X_1^{(e)} - X_3^{(e)}) + (X_3^{(e)} - X_2^{(e)})(Y_1^{(e)} - Y_3^{(e)})} \\ \varphi_2^{(e)} &= \frac{(Y_3^{(e)} - Y_1^{(e)})(x - X_3^{(e)}) + (X_1^{(e)} - X_3^{(e)})(y - Y_3^{(e)})}{(Y_2^{(e)} - Y_3^{(e)})(X_1^{(e)} - X_3^{(e)}) + (X_3^{(e)} - X_2^{(e)})(Y_1^{(e)} - Y_3^{(e)})} \\ \varphi_3^{(e)} &= 1 - \varphi_1^{(e)} - \varphi_2^{(e)}.\end{aligned}$$

Thus for every element there exists three local basis functions. A plot of these local basis functions is shown in figure ?? on an arbitrary triangular element.

6.3. Global Basis. The next step is to construct the global basis functions. Similarly to the strategy for the construction of the local basis functions, each global basis will be associated with a single vertex of the mesh. Thus for a mesh with N elements, then there will be some number of global basis functions less than $3N$. The method to define the global basis function is to define it to be 1 at the associated vertex, and 0 at all other vertices in the mesh.

In order to achieve this, the global basis function can be constructed as a piecewise combination of k different local basis functions, where k is the number of element that share a given vertex. The exact formulation of the global basis functions is dependent on the triangular mesh, and the current vertex index. An example of what a global basis function could be like is depicted in figure ?? . One can intuitively think of the global basis functions as k sided pyramids this their peak directly above the associated vertex.

6.4. Local System. Using the definition of local basis functions that were constructed in section 6.2, the construction of an element-specific system of equations is preformed. The system of equations is identical to the global system, but is simply specified to a single element. Thus the local system of equations in matrix form is

$$G^{(e)} \partial_t U^{(e)} = \left(\bar{A}^{(e)} + \bar{B}^{(e)} + \bar{C}^{(e)} \right) U^{(e)} + F^{(e)}.$$

In this system of equations, the elements of each matrix are only determined by the local basis function on the element e . The size of the matrix is equivalent to the number of vertices in the element, thus for the triangular element case, the matrices have a size of three. The element of these matrices can be found through the following equations

$$\bar{A}_{ij}^{(e)} = - \sum_{\alpha, \beta=1}^2 \int_{E_e} \frac{\partial \varphi_j^{(e)}}{\partial x^\beta} \frac{\partial}{\partial x^\alpha} \left(A_{\alpha\beta} \varphi_i^{(e)} \right) d\vec{x} \quad i, j = 1, 2, 3$$

REFERENCES

- [BBC⁺93] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June M. Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charels Romine, and Hen Van der Vorst. *Templates for the Solution of Linear Systems: Buildign Blocks for Iterative Methods*. Society for Industrial and Applied Mathematics, 2 edition, 1993.
- [Che89] L. Paul Chew. Constrained delaunay triangulations. *Algorithmica*, 4:92–108, 1989.
- [Che93] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. *9th Annual Symposium on Computation Geometry*, pages 275–280, 1993.
- [Dv08] V. Domiter and B. Žalik. Sweep-line algorithm for constrained delaunay triangulation. *International Journal of Geographical Information Science*, 22(4):449–462, 2008.

- [HKA12] Farzana Hussain, M. S. Karim, and Razwan Ahamad. Appropriate gaussian quadrature formulae for triangle. *International Journal of Applied Mathematics and Computation*, 4(1):24–38, 2012.
- [KH15] Dmitri Kuzmin and Jari Hämäläinen. *Finite Element Methods for Computational Fluid Dynamics*. Society for Industrial and Applied Mathematics, Philadelphia, 2015.
- [Kry31] A. N. Krylov. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined. *News of Academy of Sciences of the USSR*, 7(4):491–539, 1931.
- [Rup95] Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.
- [She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.
- [She02] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21,74, 2002.
- [Slo87] S. W. Sloan. A fast algorithm for construction delaunay triangulations in the plane. *Adv. Eng. Software*, 9(1):34–42, 1987.
- [Slo93] S. W. Sloan. A fast algorithm for generating constrained delaunay triangulations. *Computers & Structures*, 47(3):441–450, 1993.
- [SS86] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *Sci. Stat. Comput.*, 7(3):856–869, 1986.
- [Ž05] Borut Žalik. An efficient sweep-line delaunay triangulation algorithm. *Computer-Aided Design*, 37:1027–1038, 2005.