[Name of the Writer]

[Name of Instructor]

[Subject]

[Date]

Finite Element Method

**Weak Form**

 A weak form is an integral form that should be restated to develop the finite element formulation. Both strong and weak form are equal. Weak form is basically a reformulation of the strong form. To create a weak form of an equation we multiply it with an arbitrary function which is also known as weighted function. We will multiply arbitrary value that is v(x,y) to the strong form of Poisson equation (Gagandeep, n.p):

∆u = ∇2u (i)

We will obtain after multiply v(x,y) to equation(i):

∆u = v∇2u= *f*v (ii)

 After multiplying, we will integrate the equation (ii) over omega ():

 (iii)

 First, we assume that v(x,y) is an arbitrary value, but this is not completely right assumption. We will define a space of functions and call it H1 which is defined below (Gagandeep, n.p):

 H1 (Ω) = {v : Ω → R : (iv)

 H1 is a function space where all the functions are restricted. We don't want our functions to get restricted so that we can define operations on them. Now we will define a subspace for our function space H1, and we will name it as X which is defined below (Gagandeep, n.p):

 X = { v ∈ H1 (Ω) : *v*|∂Ω = 0 } (v)

 Now we move back to the equation (iii) first we will integrate that equation (iii) and then describe why we chose H1 as our function space. As we know that

 ∇(v∇u) = ∇v · ∇u + v∇2u, so we can write equation (iii) like this:

 (vi)

We will apply Gauss’s theorem on , the equation (vi) will be reduced to:

 (vii)

We will get at the end:

=

The equation (vii) is the final formulation of the weak form which is equal to the strong from above in the equation (i). We can get back to the original equation if we perform all these steps in reverse order. We chose function space as H1 because you can clearly see that in strong form u had two separate partial derivatives, so strong form required continuous differentiation of u. The new function has helped in lowering the requirement of differentiation by transforming a partial derivative onto the defined weight function v(x,y) (Gagandeep, n.p).

The biggest advantage of the weak form is that we can lower the requirements of the differentiation of strong form using weight function. Now it is also easy to understand the subspace X which is a subspace of H1. The weak form requires that the functions must be in H1 and the strong form requires that u must be zero along the boundary. So, X is the subspace of all the functions that which are zero (Gagandeep, n.p).

In the weak form, we can see that the spatial derivatives which are used are of less order than in the residual because we integrated by parts in the above method to create weak form. This reduction of the derivatives helps in using the lower order polynomials to find out the solution for the approximate function. In Partial differential equations, where the order of highest derivative is even, we will choose the weak form of the weighted residual to obtain the same highest derivative on the weight function and the solution. For example, we are having a fourth-order derivative of a given strong form then, in this case, we will have second order derivatives for the weak form. The weak form will also have second-order derivatives on the weight function and the approximate solution of the weak form (MIT OpenCourseWare, n.p).

**K Matrix**

 A k-matrix is a sort of cube root of the identity matrix, but it is different from the identity matrix because it contains complex terms.

 , this matrix identifies k3=I, where I is known as the Identity matrix.

 In the finite element method, Stiffness matrix is also known as the K matrix. There are many methods by which we can derive the stiffness method which include the Galerkin method, variational principle, etc. Stiffness matrix is symmetric and square. All the diagonal elements are positive in the stiffness matrix. The element stiffness matrix should be replaced in their proper position of each element. To calculate the global stiffness, first we will initialize global stiffness matrix [K] as 0, then we find out the properties of individual element and will find out the local stiffness [k] of that element. The next step will be to add the local stiffness matrix to the global stiffness matrix using the actual locations. We will perform the last two steps which were adding and computing local stiffness until all local stiffness matrix become global stiffness matrix (Zienkiewicz, n.p).

 Now we will discuss the procedure to include the effect of boundary condition in the stiffness matrix for the finite element method. In the finite element method, the partitioning of the global matrix is supported in a systematic way for coding and calculations. For example, let consider the equation of equilibrium is shown as (Stiffness Matrix and Boundary Conditions, n.p):

{F} = [K] {d} (i)

Here K is known as the global stiffness matrix, d is the displacement vector, and F is the load vector. Using the method of partitioning, the equation (i) will be partitioned like this:

 (ii)

 In equation (ii) ‘a’ allude to the displacements that are free to move, and ‘b’ allude recommended support displacement. After expanding equation (ii) we get:

 (iii)

 From the above equation, it is possible to find out the free displacement of the structure {da} which is:

 (iv)

 If the displacement of the vector is zero i-e, , then the equation (iv) will become:

 (v)

 We found out that by rearranging assembled matrix, the unknown displacements from equation (iv) which are corresponding to the portion can be used for the purpose of a solution and this is possible because the displacements were removed as they were set to zero. However, if support has some known displacements, then equation (v) could be used for solution purposes. We can also apply the above method by partitioning stiffness into three parts. When we partition stiffness into three parts, we will be having three coefficients for displacement which will be ‘a', ‘b' and ‘r'. In which ‘a' will refer to known displacement, ‘b' will refer to the known displacement and ‘r' refers to the zero displacements. We can solve all the equations independently to find out the required unknown results (Zienkiewicz, n.p).

There are many other techniques in computer programming to handle boundary conditions. One of the most common and easiest approaches is to set the diagonal element of the stiffness matrix which is similar to zero displacements and equivalent to all off-diagonals elements to zero. The numbering of nodes must have the minimum bandwidth, if general stiffness matrix is to be created in half band form. For this we put the labels in a systematic manner regardless of whether the joint displacements are unspecified or restricted. However, if the unspecified displacements are tag first then the matrix operations can be constricted and due to this overall stiffness matrix in this case might be ignored (Stiffness Matrix and Boundary Conditions, n.p).

**M Matrix**

 M-matrix is the combination of Z-matrix and A in which all the leading principle of A are positive. A can be written in the form A=kI-B, where B is a not a negative matrix but its spectral radius is less than K. All the Eigenvalues of A must be positive in M-matrix. A must be non-, and a-inverse must not be negative. In M-matrix all principal minors of A must be positive. A+kI also must be non-singular for all the k>= 0 in M-matrix (Horn & Johnson, n.p).

 In finite element analysis, the sparse matrix also known as M-matrix. The finite element method has many advantages but the main advantage of the finite element method is the final matrix in FEM is a sparse matrix. This makes FEM very responsive for people and helps the users who want to save storage. There are different methods to store a sparse matrix but one of the best methods is the row-indexed storage method. This method makes it very easy to construct a matrix and also make use of iterative methods very easy. The row-indexed storage method consists of three matrices which are used to store the non-zero elements of a sparse matrix. We assume the number of rows to be N. The first matrix is an N\*1 vector which is used to store the number of non-zero elements in all rows and is known as the Index matrix which consist of integers and it takes very less memory.

The next matrix which is used is N ´ M matrix, where M in any row is the maximum number of elements which are not zero. This matrix is known as a data matrix because it stores all the values which are not zero. The next matrix is known as ColNos which stores the column number that is corresponding to the entries in the data matrix, and its size is equal to the second matrix.

# Works Cited

Module: 2 Finite Element Formulation Techniques Lecture 4: Stiffness Matrix and Boundary Conditions n.p (n.d). Retreived from <https://nptel.ac.in/courses/105105041/m2l8.pdf>

OpenCourseWare, MIT. "Unit 2: Numerical Methods For Partial Differential Equations | Computational Methods In Aerospace Engineering | Aeronautics And Astronautics | MIT Opencourseware." <http://Ocw.mit.edu> N. p., 2019.

R. A. Horn and C. R. Johnson, Topics in Matrix Analysis, Cambridge University Press, Cambridge, 1991.

Singh, Gagandeep. "Short Introduction to Finite Element Method." Norwegian University of Science and Technology(2009).

Zienkiewicz, Olgierd Cecil, et al. *The finite element method*. Vol. 3. London: McGraw-hill, 1977.