Programming for Engineers and Scientists

Homework 1 – Design of a FE program

Prof. Marco de Corato

Dal, Jor Fergus	jor.dal@epfl.ch
Parisi, Federico	federicoparisi95@gmail.com
Khaloian, Aren	arenkhaloian@gmail.com
Saridar, Nadim	nadimsaridar@hotmail.ocm

In our code structure, we have opted for a design which relies on distinct functions to create and assemble the different components of the linear system Ku = f, which describes the PDE. This gives the final script a clean look and allows easy access to the different calculations in case of modification, and versatility as these same functions could be used in different scripts.

As a base for our code we have assumed that the problems to be solved are steady state and that the solution only depends on the location in space.

Inputs

X (coordinates) , T (connectivity) , equation parameters , Boundary conditions

The inputs that we need for solving the FEM problem considering that our domain has already been descretesized. First, we need the X matrix which gives us the coordinates of the nodes and the T matrix which gives us the information about the number of elements and what nodes each element is made of. The other input data that we need is the data of the equation. This data can be the coefficients of the problem for example the k for conductivity and the source term of the equation in case it exists. The equation data will be saved in a matrix in case it has different properties over the domain, for example if the conductivity changes inside the domain. The other needed input is the boundary conditions of the given problem. The boundary conditions can be in two forms of Dirichlet and Neuman. These boundary conditions are going to be defined on the nodes, so we are going to define them as matrices with one column for the node numbers and one column for the value of the Neuman and Dirichlet boundary conditions.

Calculating K

[K] = stiffnessMatrix(X, T, conductivity);

The stiffnessMatrix function gives the global K matrix for triangular or quadrilateral elements. Within the function the shape functions, their derivatives and the gauss points for the numerical integration are defined for both kinds of element.

Using the dimensions of X and T, this function will determine what kind of element the mesh is made of (triangular, quadrilateral, linear, linear square etc.) Each element type and their respective reference elements and solutions are separated by a control structure.
The number of gauss points, their coordinates and their weights are all defined in the function and may vary according to the element type. The jacobian is also defined and computed within the function. For the numerical integration, the gauss points are defined in the reference coordinates, and the integration is performed on the reference element, and the jacobian is used to project the results to the real element coordinates. Using the reference element and the gauss points, the function creates the elemental K matrices in a loop and assembles them before returning the global K matrix.

Calculating f

[f] = forceVector(source term,body forces, BC)

At first f will be a zero vector, that has the size of the displacement vector. The source term will be integrated and added to the respective position in the force vector. The boundary conditions are divided into two parts: Dirichlet and Neumann. The Dirichlet boundary conditions are replaced in the displacement vector, and then added to f. This will let the code reduce the system to be solved as it is going to delete the relative row and column, moving the product of the respective values on the right hand side, with the opposite sign. The Neumann boundary conditions are integrated in its given boundary then added to f.

Solving linear system

Once we have K and f, as well as having replaced the Dirichlet BC in u, we can easily solve the linear system by asking MatLab to perform u = inv(K)*f. For this we assume K is easily invertible.

Advantages and disadvantages

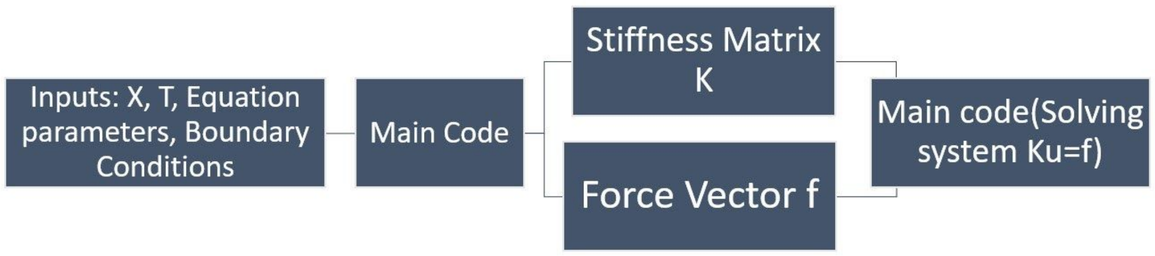
Advantages

- The stiffnessMatrix function can easily be extended to work for any element type by including a switch case and defining the reference element, its shape functions and their derivatives.

- In Script format, the code would be clear and tidy as there would not be too much text.

Disadvantages

- The main disadvantage of our code is that now it is not able to work with combined meshes (in case there are two element types in one domain) and in cases it will not be able to find the element type if the number of element nodes are similar for two cases; for example the case of the second degree triangular element and the linear quadrilateral element. This problem can be solved by improving the meshing code so the connectivity matrix will have an extra column that will indicate the element type, so in this case the function only reads the number in that specific row and identifies the element type and calculates the stiffness matrix.


Inputs: X, T, Equation parameters, Boundary Conditions
Main Code
Stiffness Matrix K
Force Vector f
Main code(Solving system Ku=f)

Conclusion

In general the code gets the connectivity and coordinate matrices, the equation parameters and the boundary conditions as inputs. Then the main code calls the stiffness function and giving it the connectivity and coordinate matrices as inputs it gets the global stiffness matrix as an output of the function. Afterwards it calls the force vector function and gives it the inputs of the boundary conditions and the source term of the equation it gets the force vector as an output. Finally having both the global stiffness matrix and the vector of forces by applying the Dirichlet boundary condition it reduces the system and solves the problem for the needed variable.