Master's Degree Numerical Methods in Engineering

Programming for Engineering and Scientists

Homework 1: Design of a FE program

Author:
Luis Ángel Avilés Murcia
luis.angel.aviles@upc.edu
Mariano Tomás Fernandez
m.tomasfernandez@gmail.com
Shardool Kulkarni
shardoolkulkarni1996@gmail.com

Professors:
Marco de Corato
Sergio Zlotnik

March 3^{rd}, 2020
Academic Year 2019-2020

Contents

1 Introduction **1**

2 Design of the Code **1**
2.1 Code structure outline 1
2.1.1 User inputs 1
2.1.2 Code functions 1
2.1.3 Structured variables 2
2.1.4 Code Flow Chart 5

3 Comments on the design **7**

1 Introduction

This assignment requires the design of a FEM code to solve the Poisson's problem. The idea of this part is make the code as general as possible in order to deal with different geometries, boundary conditions, loads and materials among others. This design involves no coding but general descriptions of each of the modules to clearly state the way it will work.

2 Design of the Code

2.1 Code structure outline

The code consists in two main parts, the user inputs and the code results.

2.1.1 User inputs

These shall include at least the following:

- Geometry;
- Boundary conditions;
- External forces/heat sources;
- Materials;
- Initial conditions (depending on the nature of the problem);
- Element type: triangular, quadrilateral
- Interpolation type: linear, quadratic
- Decision on whether the problem will be solved using linear or quadratic elements. Also, the mesh needs to be checked by the user to determine if the results will be good enough.

2.1.2 Code functions

The functions helps to compute some characteristics value and arrange the information in a correct way to use an appropriated solver to get the final results. Some of the main functions the code shall include are the following:

- Get node coordinates;
- Element connectivities;
- Shape functions and derivatives;
- Quadratures and weights;
- Element stiffness matrix;
- Stiffness matrix assembly;
- Force vector per element;
- Force assembly;
- Jacobian per element.

A detailed data sheet for every function is presented below including variables needed to implement on the FEM code as well as connections within the functions to work properly.

2.1.3 Structured variables

Elements will be considered as a structure variable which contains several characteristics of the element, for instance material, Gauss points, shape functions, etc. This structure form allows to collect all properties of a variable or object and help having a clearer code. Following the Matlab Sintax, the properties of the object will be name as; `element.shapeFunction` to indicate the shape function of the element, `element.gaussPoint` to indicate the gaussian point of the element and so on.

Nodes Coordinates
This function get the coordinates of the nodes created from the geometry definition of the problem. The mesh is created using linear interpolation.

Function	`getNodeMesh`
Inputs	modelDefinition (Geometry), deltaX, deltaY
Outputs	nodeCoordinates
Description	Get the coordinates of nodes according with type and size of elements
Uses	none
Used by	`elementConnectivity`
Comments	The nodes are distributed equally according with the number of elements or size of element in each direction (deltaX,deltaY). Same nodes can generate either rectangular or triangular elements in 2D case and tetrahedra in 3D.

Table 1: Get node mesh

Element Connectivity
This function saves the information relating the number of nodes with the elements formed in between these nodes. The element are formed following a clockwise direction for both triangular and quadrilateral elements.

Function	`elementConnectivity`
Inputs	Mesh nodes, Element type, Interpolation type
Outputs	connectivityMatrix
Description	Matrix with global numbering nodes conforming the elements
Uses	...
Used by	`elementalMatrix`, `shapeFunction`
Comments	valid element shapes are: linear triangular (3 nodes), quadratic triangular (6 nodes), linear quadrilateral (4 nodes), quadratic quadrilateral (9 nodes), linear tetrahedron (4 nodes) and quadratic tetrahedron (10 nodes)

Table 2: Element Connectivity

Shape Functions
Shape functions can be linear or quadratic. However, linear shape functions are the most used for both linear and non-linear problems. Shape functions can vary from one type of element to other.

Function	`shapeFunctions`
Inputs	elementConnectivity
Outputs	shapeFunctions
Description	To get the interpolation functions from nodes to Gauss points
Uses	is used to obtain the gradients and interpolations of the approximated solution
Used by	`stiffnessMatrix`, `elementForceVector`
Comments	Shape functions can be linear or higher order depending on the element type

Table 3: Shape Functions

Derivative Shape Functions

Function	`shapeFunctionDerivatives`
Inputs	elementConnectivity
Outputs	shapeFunctionDerivative
Description	compute the spatial derivatives of the shape functions
Uses	...
Used by	`elementJacobian`
Comments	Elements with C^0 continuity

Table 4: Shape Function Derivatives

Force vector per element

In the integration of the elements the force vector is checked in every loop, to include the effect of external applied and/or body forces, fluxes and/or source terms. If the element being integrated has any of these the force vector is different than zero. The main characteristics of the function are explained in Table 5.

Function	`elementForceVector`
Inputs	`sourceTerm`,`externalForces`, `shapeFunctions`, `Quadrature`
Outputs	`elementForceVector`
Description	integrates the external forces
Uses	force boundary conditions
Used by	force assembly
Comments	for each node of each element, the function distributes the external force into the node, for example, a two-noded 1D element will distribute the weight half to each node.

Table 5: Element force vector.

Force assembly

Once the elemental forces are obtained, the assembly is done using these forces and the connectivity within elements to go from the discrete problem scheme to a global scheme. Its characteristics are shown in Table 6.

Function	`forceAssemble`
Inputs	`ConnectivityMatrix`, `elementForceVector`
Outputs	`ForceAssemble`
Description	translates the elemental forces to the general problem and sums the contribution of each element to the same node when necessary.
Uses	connectivity matrix and element force vector
Used by	problem solution
Comments	takes the elemental force vector and arranges its contributions to the general problem scheme.

Table 6: Force assembly.

Element Jacobian
Compute the jacobian in each step of the calculation

Function	`elementJacobian`
Inputs	`getNodeMesh`
Outputs	`elementJacobian`
Description	uses the discretization of the mesh in both directions to calculate the jacobian
Uses	used to parametric elements and compute changes in area or volumen
Used by	Element stiffness matrix
Comments	transforms the reference element coordinates to the real coordinates of the problem.

Table 7: Jacobian of the transformation.

Quadrature
Numerical integration is done over Gauss quadrature points

Function	`Quadrature`
Inputs	`elementType`
Outputs	`Weights`, `QuadraturePoints`
Description	This function gives out the Gauss quadrature points and weights
Uses	Used to calculate the value of integrals
Used by	shapeFunctions
Comments	This function calculates the numerical integral based on the order of Gauss integration

Table 8: Quadrature Function

Stiffness Matrix per element
Calculates the individual components of the stiffness matrix, from the nodes, connectivities and element properties.

Function	`Kmatrix`
Inputs	`getNodeMesh`, `elementConnectivities`, `elementJacobian`, material properties
Outputs	Components of the Stiffness matrix
Description	This function calculates the individual components of the stiffness matrix
Uses	getNodeCoordinates, elementConnectivity
Used by	Stiffness Assembly
Comments	Performs numerical integration and calculates every element of the stiffness matrix

Table 9: Stiffness Matrix for each element

Stiffness Assembly
Carries out the assembly of stiffness matrices, before they are used by the solver.

Function	`KAssembly`
Inputs	`getNodeMesh`, `Kmatrix`
Outputs	global stiffness matrix
Description	Performs assembly of stiffness matrix for the solver to solve it
Uses	Kmatrix
Used by	Solver subroutine
Comments	The output is the complete stiffness matrix to the solver.

Table 10: Stiffness Assembly.

2.1.4 Code Flow Chart

The following chart shows the basic steps followed by a FEM code. The connections between components are approximates and can vary according with the type of formulation, problem definition, etc. Additional, more functions can be add at each stage to improve the capabilities of the code.

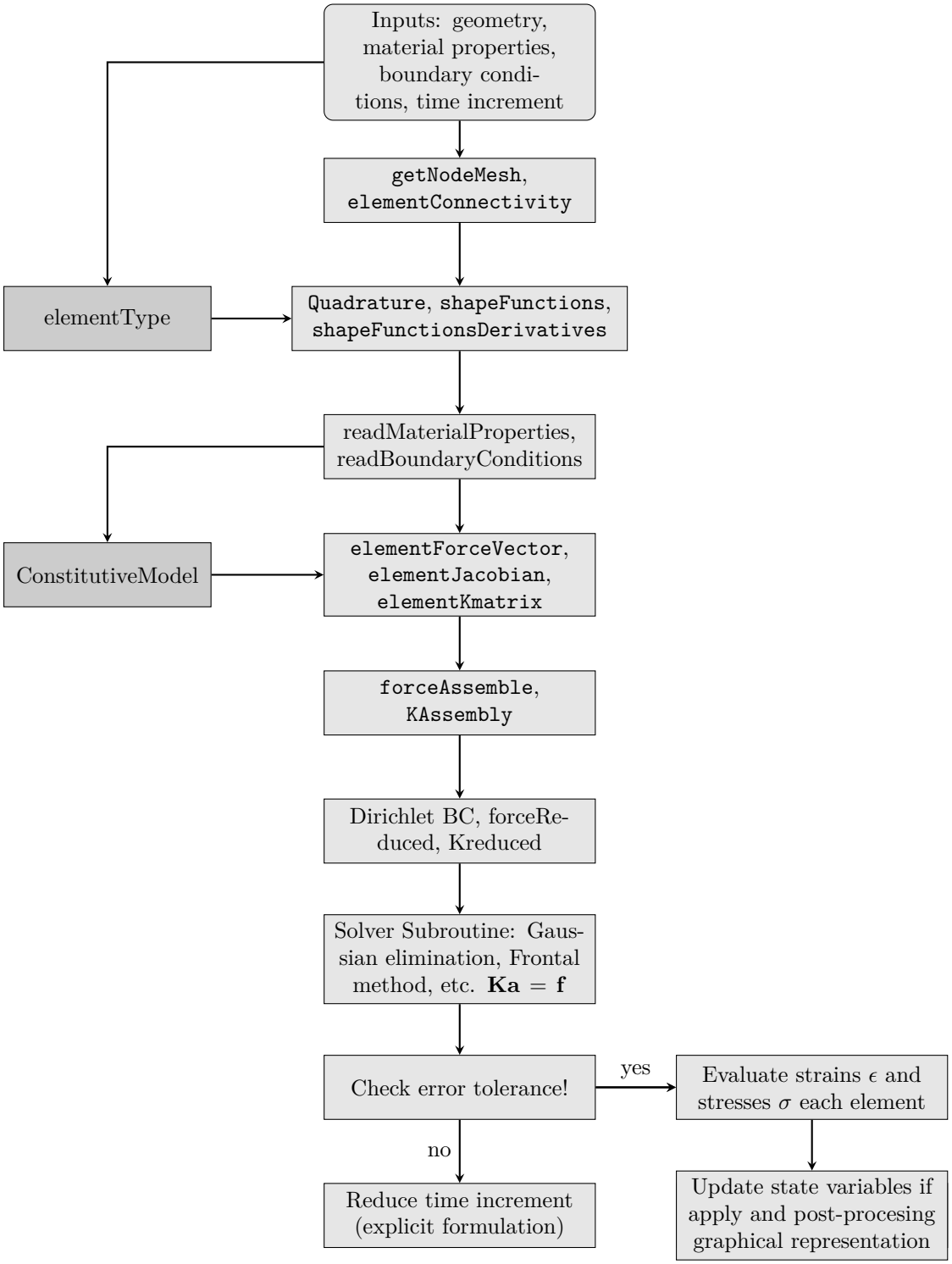

Figure 1: Flow Chart of a FEM code

3 Comments on the design

Some additional features of a FEM code includes:

- This code structure follow the GiD structure of pre-procesing like material definition, boundary conditions specifications and external forces applied.
- The user has to define use Lagrange or Serendipity rectangular mesh in Mesh Menu in GiD – Quadratic Type option.
- Matlab is good for FEM codes with few to medium number of elements. For large mesh and complex problems, Matlab become computational expensive requiring a lot of time to solve the equations.
- For most of the simulation problems, a quadratic element is enough to get accuracy (with a minimum error) in the results. For this reason, we consider that linear and quadratic element type are enough to run a problem and we will no consider cubic o quartic elements.
- The element information such as the degree, gauss points gauss weights, the shape functions the derivatives of the shape functions are stored in struct datatype.
- The variables such as geometric properties, Connectivities stiffness matrices, the load vector are stored in standard MATLAB matrices where each element is a double.