Programming for Engineerings and Scientist
Universitat Politecnica de Catalunya

# Assignment 2.a

Federico Parisi
Jor Fergus Dal
Nadim Saridar
Aren Khaloian


20-05-2020

# Contents

**1 Abstract** **1**

**2 The Code** **2**

2.1 Structure . . . . . 2

2.2 Inputs . . . . . 2

2.3 Classes . . . . . 2

2.3.1 Reference Elements . . . . . 2

2.3.2 Matrices and Vectors . . . . . 3

2.4 Post-Process . . . . . 3

# 1. Abstract

In this report we will describe the design of a finite elements (FE) code in C++. Taking into account the type of programming language, there will be described the main features that an Object Oriented code should have. Thus, we will briefly describe the aim of the code and its structure, focusing on the definition of the main classes and their interaction.

# 2. The Code

## 2.1 Structure

Being an Object Oriented (OO) language, all the code works on an interaction between classes. Each class is defined by the data and the functions that will give, as an output, the data needed in order to make the whole code work.

The linear system to be solved is $Ku = f$, 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. This is a peculiarity of OO languages. 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.

## 2.2 Inputs

The inputs needed are the same as the MATLAB FEM code. It will be needed the X ( coordinates ) , T ( connectivity ) , equation parameters , Boundary conditions.

## 2.3 Classes

In a OO program, classes are useful to be defined for data structures that are frequently used or of which there will be several instances. For the FEM this means the principal components of the method which have specific values and interact in meaningful ways. In a FEM we can categorize the components into two object classes, elements and matrices.

### 2.3.1 Reference Elements

In this class, we are going to be define all the elements and the related parameters. These parameters will be the shape functions, the type of element, the degree of freedom of each element, the Gauss Points with their weights and the Jacobian.

As input, the object will receive the degree of the element, the type (quadrilateral, linear, triangular etc) and the dimensions. As output it will give all those information stored, relative to the element needed.

The different kinds of elements will be defined as subclasses of the Reference Element class, and hence will inherit basic features common to all. Then the element specific to the problem will be created from one of these subclasses with the information read from the dimensions of X and T.

### 2.3.2 Matrices and Vectors

As a general class we define the matrix data structure. In this class we have the number of columns and rows of the matrix. From this class we define several subclass matrices like the stiffness matrix, the force vector and the nodal solutions which inherit the main matrix structure but also add their unique features and methods. The five objects of this class would be the coordinate matrix (X), connectivity matrix (T), stiffness matrix (K), force vector (f), and the nodal solutions (u).

X, the coordinate matrix is defined by the values given as an input to the code. From its dimensions we can extract information on the number of nodes in our mesh.

T, the connectivity matrix is defined by values given as an input, and from its dimensions we know how many elements our mesh had and how many nodes per element, which gives us the element class inherit to the problem.

K, the stiffness matrix reads and copies the number of rows of X to create its dimensions. Next it extracts values from the given element class of the problem, like Gauss points and the Jacobian, in order to assemble.

f, the force vector uses the entry values of the boundary conditions and problem statement to assemble.

u, the nodal solution has integrated methods which read the values of f and K in order to solve the linear system and give the final results.

## 2.4 Post-Process

The solution field will be stored inside the class defined before. To show it, can be implemented a Matplotlib library. On the other hand, it can be written a *.vtk* file in order to export the results in Paraview, in the same way as done in the MATLAB written code.