Universitat Politècnica de Catalunya
Numerical Methods in Engineering
Programming for Engineers and Scientists

Design of a FE program in C++

May 25, 2020

Sebastian Ares de Parga	aresdepargas@gmail.com
Eduard Gómez	eduard.gomez.escandell@gmail.com
Federico Valencia	fvalenciao95@gmail.com

1 Objects

Our matlab implementation of the FEM program was already making use of classes, however we didn't use inheritance to our advantage. Here we propose a modified model where it is used.

1.1 Elements

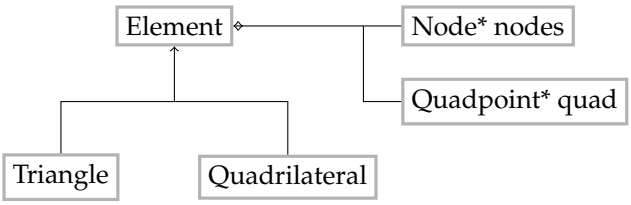

Figure 1: Objects used to describe elements

Triangular and quadrilateral elements are quite different so they need different objects. The element class will need empty virtual functions for calculating the area and the jacobian matrix., which will be implemented for each element subclass.

1.2 Point-like objects

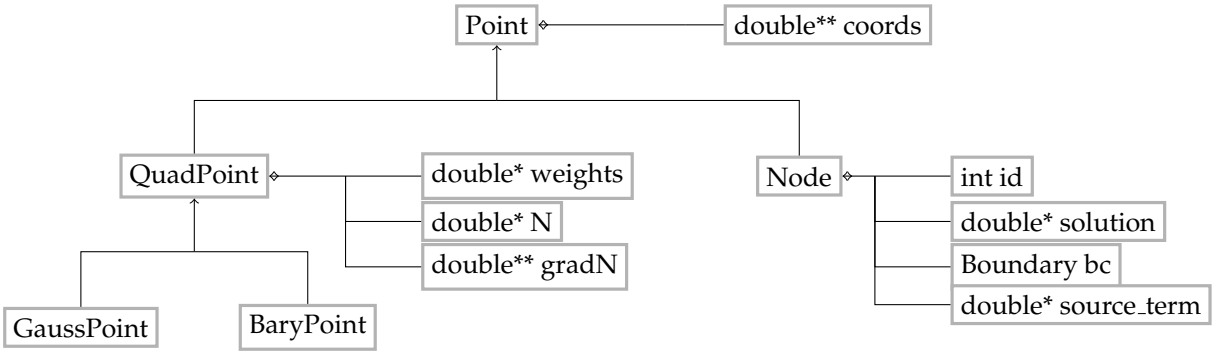

Figure 2: Point-like objects

There are mainly two distinct types of points: quadrature and mesh. Quadrature points can store their weights, the values of the shape functions and the gradients. They will also need a function to translate from isoparametric $f : (\xi, \eta) \mapsto (x, y)$ for quadrilateral elements and from barycentric coordinates $f : (\lambda_1, \lambda_2, \lambda_3) \mapsto (x, y)$ for triangular elements. Mesh points, or nodes, can store their IDs, the value of the solution (or a pointer to it), a boundary condition value, and the values of other functions $f(x, y)$ such as the source term, viscosity, thermal coefficient, etc.

1.3 Other classes

We will also use other classes to hold information such as the mesh or the system of equations, but we don't predict the need of inheritance for now. We will have the following classes:

- Domain: will contain the instances of elements and nodes, and a method to import the mesh, which will read a slightly modified version of the mesh files used in our Matlab implementation.
- System Of Equations: will contain the global system of equations and methods to manipulate it (assembly, imposition of boundary conditions, solving, etc.).

2 Output

There are multiple options regarding the output of the code, such as implementing a small python subroutine using the Matplotlib library. This would require to output a table of nodes, connectivities and solution vector. The same could be achieve implementing the plots in Matlab.

Alternatively, we could aim to use "Paraview" in the same manner we did for the previous implementation. This, however is more time-consuming and we consider post-processing not a priority, therefore we will prioritize improving other parts of the code before implementing a `.vtk` output.