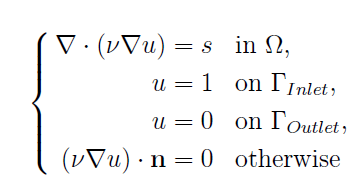
|  |
| --- |
|  |
| Poisson problem in C++ |
| Programming for Engineers and Scientist |
|  |
| **Sònia Garrido Ballart** |
| **Alba Navarro Casanova** |

|  |
| --- |
|  |

# Statement of the Problem

We are asked to implement a FE code for Poisson problem using C++ in a similar way as we did in the first assignment of the course.

The problem that we aim to solve is:



Where the source term s = 0 and the diffusivity ν = 1 are given.

The main difference with the Matlab code that we have developed in the first part of the course is that the code has to be adapted to the C++ language and we are explicitly asked to write and object oriented program. In the following section we will explain which classes are created and how the program is structured.

# Structure of the Program

As we have told to develop the code to solve the Poisson problem we are using object oriented programming. As this was really new for us we have only implemented one of the eight problems that we were solving in the first assignment. That means that our problem is only for 2D structures using triangular linear elements. The 2D problem with triangular quadratic elements and the quadratic linear and quadratic elements will be solved in a really similar way but they are not implemented. For the 3D case changes the way that we solve the linear system, this case is not implemented.

To solve the FEM problem we need input files with the coordinates of the nodes of the mesh of our problem and another with the connectivity matrix that relates the elements with the nodes of the mesh. This information is stored in text files that have to be read by our program. We also need to impose the boundary conditions, in this particular case we only have Dirichlet boundary conditions in the inlet () and on the outlet () nodes, this information is also stored in a text file and has to we read. The diffusion coefficient and the source term also have to be determined, in this case we consider and for all the nodes.

Depending on the problem that we are solving we will have different integration points and integration weights as so for the shape functions and its derivatives. We need these integration parameters and shape functions to generate the stiffness matrix K and the vector f. This is done in different ways depending on the dimensions of the element that we are studying.

The C++ program does the function of calculate the solution of the problem; we use Paraview as a postprocessor. To see the results of our unknown we need to write them in a .vkt file that can be opened in Paraview.

From all we want to do we extract what classes we are going to define to solve the Poisson problem. To define classes we need a header file and a source file that includes the header. The header file is where the information of the class goes, is like the function declaration, and the source file is like the function definition. We have defined the following classes:

1. class BC\_Class 🡪 class that allows to read the boundary conditions of our problem that have been read form a text file.
2. Class\_aux 🡪 auxiliary class that allows to enter to the function *read\_files.* This function is used on the *main* for reading and storing the information of the files as matrixes or vectors. As we have told the coordinate file, the connectivity matrix, the boundary conditions and the diffusivity of the problem have to be read form an external file.
3. class ClassElem\_2D\_tri\_linear 🡪class that allows to create the vector of element objects, it depends on the number of elements that we have, that for the 2D triangular linear elements it is 404 elements.
4. class ClassNodes 🡪 Similarly to what we have done with the elements we create a vector of node objects ….
5. class postprocess\_class
6. class Solver

For this practice we have used the Eigen library, this is a [C++](https://en.wikipedia.org/wiki/C%2B%2B) library of template headers for [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra), [matrix](https://en.wikipedia.org/wiki/Matrix_%28mathematics%29) and [vector](https://en.wikipedia.org/wiki/Vector_%28mathematics_and_physics%29) operations, geometrical transformations, [numerical solvers](https://en.wikipedia.org/wiki/Numerical_analysis) and related algorithms.

Eigen is implemented using the [expression templates](https://en.wikipedia.org/wiki/Expression_templates) [metaprogramming](https://en.wikipedia.org/wiki/Template_metaprogramming) technique, meaning it builds expression trees at compile time and generates custom code to evaluate these. Using expression templates and a [cost model](https://en.wikipedia.org/wiki/Analysis_of_algorithms#Cost_models) of [floating point](https://en.wikipedia.org/wiki/Floating_point) operations, the library performs its own [loop unrolling](https://en.wikipedia.org/wiki/Loop_unrolling) and [vectorization](https://en.wikipedia.org/wiki/Automatic_vectorization).

# Conclusions and Difficulties

Thanks to this work we have learned how to program in C ++, the language used. We have learned how to structure a program with different features and options, involving different calculations but the same basic structure.

We must be aware that our program does not work but still believe that the most important and leading more learning is to learn the language in C ++, as well as see the advantages and inconveientes that has this type of programming. It is very important to know how to create an efficient structure depending on the type of problem you have. We believe that the idea of creating a good structure has been fulfilled, even though the program does not work because of various errors that we are not able to solve.

After doing the same problem with both Matlab and C++ we have know that this type of program is much better to do in C ++ language because it is [Object-oriented programming](https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiL_LqL-sXMAhUCshQKHfpJBVMQFggcMAA&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FObject-oriented_programming&usg=AFQjCNHPqTWrYhVHoaic6XQm6axOMD4ICA&sig2=nhzwktNoCls4FmDM5b1J4A&bvm=bv.121421273,d.d24), the most important advantages of OOP are:

* Reusability. When we designed properly classes, they can be used in different parts of the program and in many projects.
* Maintainability. Because of the simplicity for abstracting the problem, object-oriented programs are easier to read and understand, they allow us to hide implementation details leaving visible only the most relevant details.
* Modifiability. The facility to add, delete or modify new objects allows us to make changes very easily.
* Reliability. By dividing the problem into smaller parts we can test them independently and isolate any errors that may arise much more easily.

But on the other hand this Object-oriented programming also has some disadvantages such as:

* Change in the mindset of traditional programming object-oriented.
* The execution of object-oriented programs is slower.
* The need to use class libraries requires learning and training.