Programming for Engineers and Scientists

Yuyang Wang, Jorge Balsa Gonzlez
Master on Numerical Methods in Engineering
Universitat Politcnica de Catalunya,BarcelonaTech
Assignment1

February 26, 2016

In the initial classes of PES we extended an existing Matlab code implementing FE for a Poisson equation

$$\begin{cases} \nabla \cdot (\nu \nabla) = s \quad in \quad \Omega \\ u = 1 \quad on \quad \Gamma_{Inlet} \\ u = 0 \quad on \quad \Gamma_{Outlet} \\ (\nu \nabla u) \cdot n = 0 \quad otherwise \end{cases} \tag{1}$$

where the source term $s = 0$ and the diffusivity$\nu = 1$ are given.

This assignment consist in further extending that code. We ask you to develop one single Matlab code able to solve the Poisson problem, accounting for the following options and feature:

- Read a general mesh and boundary condition from input files.
- Solve 2D/3D and steady state problems.
- Use triangular/tetrahedral, quad/hexahedral using linear or quadratic elements.

1 PROCEDURE AND STRATEGY

1. We firstly extend the main.m file by following way:

   - We use the "input" function to ask the user to input the file name when he/she runs the main file.Totally, four files were needed :the file containing the coordinate of the nodes, the connectivity matrix, the inlet boundary condition and the outlet boundary condition. This line code was designed so that our code can be applied to any element type.
   - After read the four files, the computer can automatically identify the mesh type and the number of dimension for the problem according to the dimension of coordinate matrix and the number of nodes for each element. We designed a "switch" function so that the computer can execute different code according to the type of mesh.
   - Finally, when all the calculation finished, we are going represent the solution in a software called "Paraview[1]". The result should be written a vtk format file in the end of the main.m file. We again use the "switch" function to write vtk file which contain the problem headings, the coordinate of nodes, the connectivity matrix and so on. Because one template vtk file is not suitable for different element whose number of nodes and connectivity matrix is different.

2. Secondly, we extend another MatEI file which is used to compute numerical integration. The original file MatEI was extended to 8 files in order to meet different requirement for the integration. For example, the Number of Gauss Point for a triangle linear element is 1 while for a triangle quadratic is 4. Besides, the jacobi matrix for a 2D element is 2 rows and 2 columns while for 2D element is 3 rows and 3 columns. We just add the dimension for the jacobi matrix and the number of Gauss quadrature.

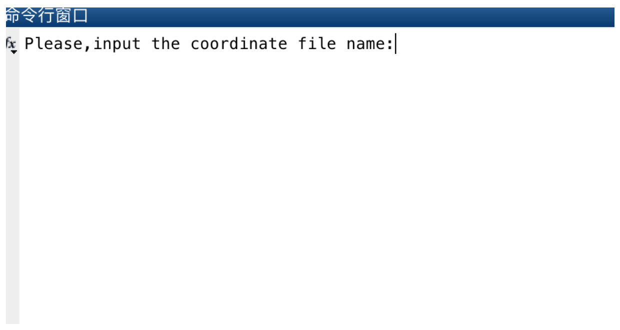
3. Thirdly, we also write some function files such as C2D4, C3D4,where the first number stands for the dimension of the problem and the second number stand for the number of nodes for each element. In these function, we defined the integration point, integration weight, shape function and the derivative of the shape function. By doing this, we are able to obtain some parameter for Quadrature.

2 IMPLEMENTING THE MATLAB CODE

We solved 8 problems using the code we have developed in order to make sure our code works well, and they are:

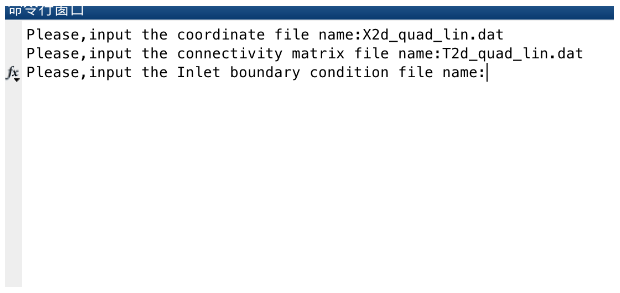
- 2 dimensional linear quadrilateral
- 2 dimensional quadratic quadrilateral
- 2 dimensional linear triangles

- 2 dimensional quadratic triangles
- 3 dimensional linear hexahedral
- 3 dimensional quadratic hexahedral
- 3 dimensional linear tetrahedral
- 3 dimensional quadratic tetrahedral



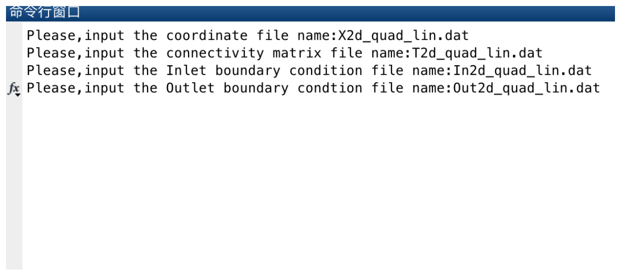
```
   Please,input the coordinate file name:
```

(a) Step 1



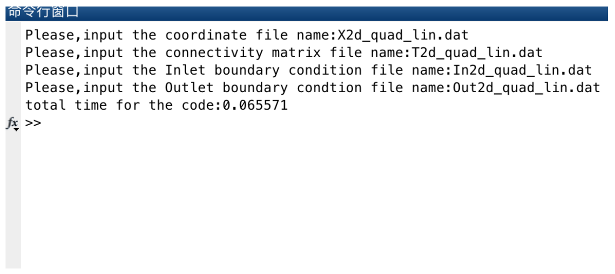
```
   Please,input the coordinate file name:X2d_quad_lin.dat
   Please,input the connectivity matrix file name:T2d_quad_lin.dat
   Please,input the Inlet boundary condition file name:
```

(b) Step 2



```
   Please,input the coordinate file name:X2d_quad_lin.dat
   Please,input the connectivity matrix file name:T2d_quad_lin.dat
   Please,input the Inlet boundary condition file name:In2d_quad_lin.dat
   Please,input the Outlet boundary condtion file name:Out2d_quad_lin.dat
```

(c) Step 3



```
   Please,input the coordinate file name:X2d_quad_lin.dat
   Please,input the connectivity matrix file name:T2d_quad_lin.dat
   Please,input the Inlet boundary condition file name:In2d_quad_lin.dat
   Please,input the Outlet boundary condtion file name:Out2d_quad_lin.dat
   total time for the code:0.065571
   >>
```

(d) Step 4

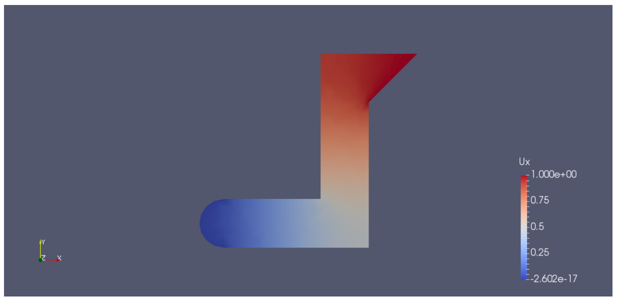
Figure 1: Explanation for the application of the code to 2D quadrilateral linear element

This process is very easy, just do it as we explained in the previous section.Figure1 shows an example for apply our to a 2D quadrilateral linear element problem. Follow the step, we can obtain the final solution and the computer can display the total execution time is 0.065571.

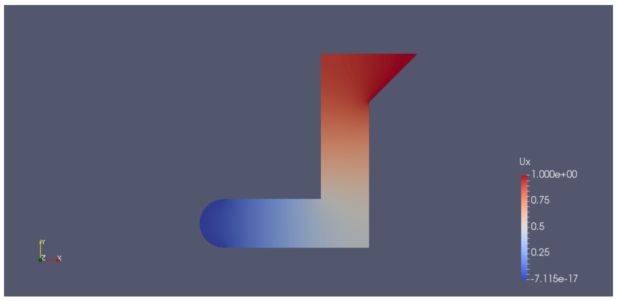
1. If the user run the main.m file in Matlab, he/she would be asked to input relevant files: coordinates, connectivity matrix, the inlet boundary condition and the outlet boundary condition.Then it loads X and T data files if they are in the same folder as main.m.
2. Matlab than identify the element type automatically according to the loaded data and execute using ”switch” and ”if-else” statement loading appropriate function.
3. It create the vtk file which can be read by Paraview program.

3 RESULTS AND DISCUSSIONS

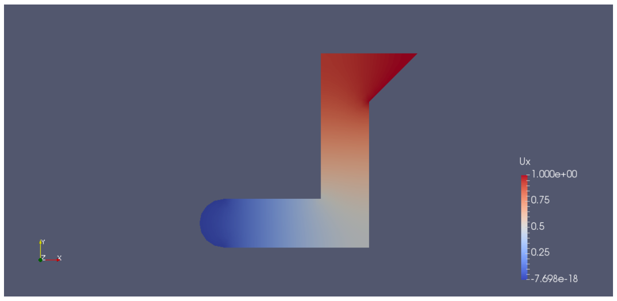
We can represent the vtk file in Paraview and the following figure2 and figure3 show the solution result for Poisson equation from different element type and dimension.



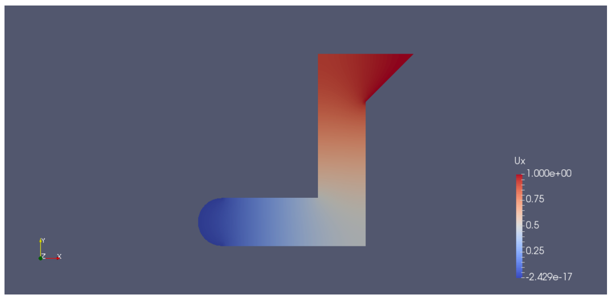
(a) Quadrilateral linear element



(b) Quadrilateral quadratic element



(c) Triangle linear element



(d) Triangle quadratic element

Figure 2: Solution for 2D Poisson equation from different element type

Apparently, we can find that the maximum solution of is 1 and minimum solution is 0 for all 8 different case.This satisfy the boundary condition we have applied. The boundary condition is that

$$u = 1 \quad on \quad \Gamma_{Inlet}$$

$$u = 0 \quad on \quad \Gamma_{Outlet}$$

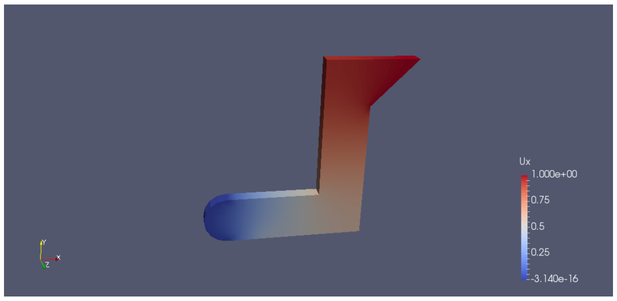
$$(\nu \nabla u) \cdot n = 0 \quad otherwise$$

So our solution verifies the boundary condition.

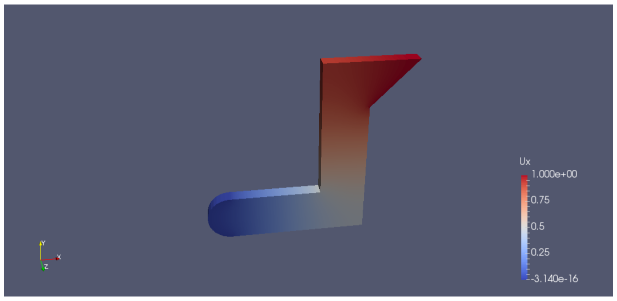
On the other hand, solution obtained from different element type represent almost the same result, so this can also be used as a verification for our code.

In order to compare the computational cost for different mesh, we run 10 times each mesh and compute the average execution time with the same computer.To be more precise, the time have been computed by Matlab inserting the code lines:

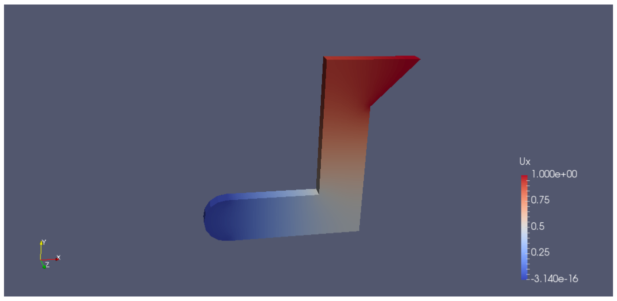
```
tic; t1=clock;
```



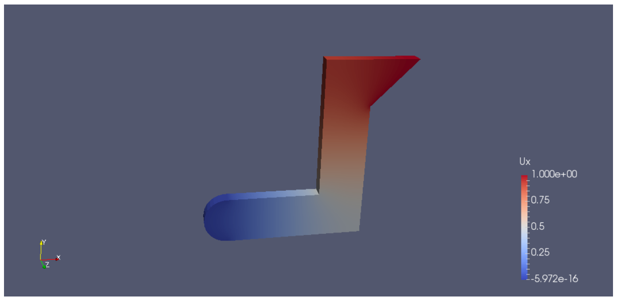
(a) Hexahedral linear element



(b) Hexahedral quadratic element



(c) Tetrahedral linear element



(d) Tetrahedral quadratic element

Figure 3: Solution for 3D Poisson equation from different element type

Just after the user input file names and displaying the value at the end of the program and all computes has been done:disp(['total time for the code:',num2str(etime(clock,t1))]);.

We can see that 3D models takes more time in be executed than 2D models. That is what we would expect because it uses more data information.Besides,we can also notice that quadratic models takes around double or more time than linear ones. It is also what we expected because a quadratic model will increase the number of nodes which may get a better solution but demand more computational cost.

Finally we can see that in 2 dimensions triangle models takes more execution time than quadrilaterals. And in 3 dimensions tetrahedral models takes more time than hexahedral ones.Thus, for element geometry preference, once other things being equal, we prefer quadrilaterals over triangles in 2D and hexahedral over tetrahedra in 3D, which is similar to the conclusion in Felipa's book[2].

4 CONCLUSIONS

We extended the given code to a more flexible code which can be used to solve not only 2D but 3D Poisson equation. The solution result verify very well to the given condition. And besides, we compared the solution result obtained from different element type, we find the color bars were almost the same which is another way to verify the correctness of our code.By

Table 1: Computational cost for each solution

Mesh type	Execution Time(s)
Quadrilateral linear	0.898
Quadrilateral quadratic	2.064
Triangle linear	1.122
Triangle quadratic	2.191
Hexahedral linear	3.630
Hexahedral quadratic	7.253
Tetrahedral linear	5.110
Tetrahedral quadratic	16.509

comparing the computational cost for different element type, we know that a quadrilateral model cost less time than triangle model and hexahedral cost less time than tetrahedra element if other things being equal.

The code is available at github: https://github.com/Yuyang01/YuyangJorge007

5 REFERENCES

[1] Squillacote, Amy Henderson, and James Ahrens. The paraview guide. Vol. 366. Kitware, 2007.
[2]Felippa, Carlos A. "Introduction to finite element methods." Course Notes, Department of Aerospace Engineeing Sciences, University of Colorado at Boulder, available at http://www. colorado. edu/engineering/Aerospace/CAS/courses. d/IFEM. d (2004).