Introduction to FreeFem++

Noureddine Igbida [1]

1. Institut de recherche XLIM, UMR-CNRS 6172, Faculté des Sciences et Techniques, Université de Limoges 123, Avenue Albert Thomas 87060 Limoges, France. Email : noureddine.igbida@unilim.fr

Plan :

- Introduction : variational formulation and Galerkin approximation
- Basic setting for FreeFem++
- How to use FreeFem++
- Basic commands
- Example

Introduction : variational formulation and Galerkin approximation

FreeFem++ is a Free software to solve PDE using the Finite Element Method

- To use FreeFem++ we need to use the variational formulation (for the PDE)
- V a Hilbert space on $\mathbb{R}$ equipped with the norm $\|.\|$, its dual space V' equipped with the norm $\|.\|_*$. Let

$$\begin{array}{ll} a : V \times V \to \mathbb{R} & \text{a bilinear form} \\ l : V \to \mathbb{R} & \text{a linear form.} \end{array}$$

- Consider the problem :

$$(VP) \qquad \text{Find } u \in V \quad \text{such that} \quad a(u, v) = l(v) \quad \text{for any } v \in V.$$

- **Galerkin approximation :** The solution of (VP) is approximated by the solution in a finite dimensional space :
  - Replace V by a discrete approximation V_h and solve (VP) in V_h rather than V :

$$(VP)_h \qquad \text{Find } u_h \in V_h \quad \text{such that} \quad a(u_h, v) = l(v) \quad \text{for any } v \in V_h.$$

  - Choose the family of spaces V_h for some parameter $h \to 0$ (think of h as a mesh size) such that the solution converges to u in a reasonable sense.

Basic setting for FreeFem++

PDE's that can be handled by variational formulation :

Let $L : U \to V$ be a linear operator where U and V are Hilbert spaces. Then :

$$\Big\Updownarrow \quad \begin{array}{l} \min_{u\in U} \mathcal{P}[u] := \frac{1}{2}\|L[u]\|^2 - \Big\langle f, u \Big\rangle \\ \\ K[u] = f \quad \text{where} \quad K = L^* \circ L, \\ \\ a(u,v) = l(v) \quad \text{for any } v \in U, \end{array}$$

where

$$a : U \times U \to \mathbb{R} \quad \text{the bilinear form given by}$$
$$a(u,v) = \Big\langle L(u), L(v) \Big\rangle.$$

and

$$l : U \to \mathbb{R} \quad \text{the linear form given by } \Big\langle \mathcal{L}u, \xi \Big\rangle = \Big\langle f, \xi \Big\rangle.$$

In this case

$$\mathcal{P}[u] = \frac{1}{2} a(u,u) - l(u), \quad \text{and} \quad \Big\langle \mathcal{P}'(u), w \Big\rangle = \Big\langle K[u] - f, w \Big\rangle.$$

Approximation space V_h : finite element $P0$, $P1$, $P2$, and many others $P1dc$ ($P1$ discontinuous), $P1b$ (P1 bulle), $P2b$, $P2dc$, $RT0$ (Raviart-Thomas), $P1inc$ (P1 non conforme).

How to use FreeFem++

- Download : http ://www.freefem.org/ff++
- Install FreeFem++
- It runs on Windows, Linux and Mac OS
- It uses basically C_{++}
- FreeFem++ :
  - Generates Meshes
  - Build automatically the matrix M (the so called Mass/Rigid Matrix)
  - Build automatically second member
  - Use diverse Linear Solvers (integrated)
  - Work for 2d or 3d problems
  - Generate Graphic/Text/File outputs
  - Do more ...
- How to use it
  - Use a text editor to write your script
  - Save it with file extension .edp (toto.edp)
  - Run it by clicking on it under Mac OS (also Windows), or by executing the command *FreeFem++ toto.edp* under Linux.
- As most of the Softwares, a full documentation and many examples are included in FeeFem++

Basic commands

The skeleton of the script solving a PDE :

- **variable declaration**
- **define the domain**
- **define the mesh**
- **define the set of finite element**
- **define the variational problem + boundary condition**
- **solve the problem**
- **plot the solution**
- **Save the results ! ! ! !**

Basic commands

- You can use all C++ commands : variables (var ...) , type of variables (int, real, bool), functions (func), tables, matrix, loops and control statements (for ..., while ...., if ...)
- **Mesh generation :** $\Omega = (x_0, x_1) \times (y_0, y_1)$
  - Regular mesh $n \times m$, use the command :

    mesh meshname = **square** $(n, m, [x_0 + (x_1 - x_0) * x, y_0 + (y_1 - y_0) * y])$;

    External borders are defined counterclockwise
  - Triangular mesh using border : More general meshes can be generated by using keywords **border** and **buildmesh**. It enables you to define mesh for open sets whose boundaries are described by parametrized curves.
    - Define a boundary by using :

      border curvename $(t = t_0, T)$ $\{x = x(t); y = y(t)$; **label**=*numlabel*$\}$;
    - Generate a mesh of the domain Ω that has a boundary giving by the previous curve (named name) by using the command

      buildmesh meshname=buildmesh(curvename (z)) ;

      where z is the number of node on the boundary name.

- **Solving a PDE**
  - Define the space of finite element by using the command

    fespace spacename (meshname , typeoffiniteelement) ;

    The typeoffiniteelement is a keyword in the following list : $P0$, $P1$, $P1dc$ ($P1$ discontinuous) $P1b$ ($P1$bubble), $P2$, $P2b$, $P2dc$, $RT0$ (Raviart-Thomas), $P1inc$ ($P1$ non-compliant).

  - **Define the variational problem associated with PDE**
    - Define the problem by using the command

      problem pbname$(u, v) = a(u, v) - l(v) +$ (boundrycond) ;
    - Solve the problem by using the command :

      pbname ;

Basic commands

- **Derivative** : use **dx** and **dy** to derive with respect to x and with respect to y respectively. For instance to compute $u_x v_x + u_y v_y$, use the command

$$dx(u) * dx(v) + dy(u) * dy(v).$$

- **Bilinear form** : $\int_{mesh} Auv$

$$\textbf{int2d}\ (meshname)(A * u * v);$$

- **Boundary condition** : use the command

on (num1 , numk , u=g) ;

- **To plot the result :** use the command

plot (var1 , [var2 , var3] , . . . [options list]) ;

  - **wait**=true/false : determines whether the graphic window closes immediately or not. If we chose true, then the program waits for a keyboard action type :
    - +/- for to zoom in / out
    - **r** to refresh the window
    - **p** to save in postscript

    The default value is false.
  - **value**=true/false : show or hide the legend of color contour of the curve. The value Default is false.
  - **fill**=num : if num = 1 then the space between the contours is filled with color
  - **ps**=nom fichier : saves the curve in postscript.

Example

$$\begin{cases} -\Delta u = f & \text{in } \Omega := [0,1]\times[0,1] \\ u = 0 & \text{in } \{0,1\}\times[0,1]\cup[0,1]\times\{0,1\} \end{cases} \Leftrightarrow \begin{cases} \int\int_\Omega (u_x v_x + u_y v_y - fu)\, dxdy = 0 \\ \text{for any } v \in H_0^1(\Omega). \end{cases}$$

```
mesh Sh= square(10,10);                          // mesh generation of a square
fespace Vh(Sh,P1);                               // space of P1 Finite Elements
Vh u,v;                                          // u and v belongs to Vh
func f=cos(x)*y;                                 // f is a function of x and y
problem Poisson(u,v)=                            // Definition of the problem
      int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))         // bilinear form
            -int2d(Sh)(f*v)                      // linear form
                  +on(1,2,3,4,u=0);              // Dirichlet Conditions
Poisson;                                         // Solve Poisson Equation
plot(u);                                         // Plot the result
```

Example

example.edp

```
mesh Sh= square(10,10);                                    // mesh generation of a square
fespace Vh(Sh,P1);                                         // space of P1 Finite Elements
Vh u,v;                                                    // u and v belongs to Vh
func f=cos(x)*y;                                           // f is a function of x and y
problem Poisson(u,v)=                                      // Definition of the problem
        int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))                 // bilinear form
            -int2d(Sh)(f*v)                                // linear form
                +on(1,2,3,4,u=0);                          // Dirichlet Conditions
Poisson;                                                   // Solve Poisson Equation
plot(u);                                                   // Plot the result
```

- After writing the script by using a text editor
- Save it with file extension .edp (example.edp)
- Run it by clicking on it under Mac OS (also Windows), or by executing the command *FreeFem++ example.edp* under Linux.

More ....

include "Poisson.edp" ; // include previous script

plot(u,ps="result.eps") ; // Generate eps output

savemesh(Sh,"Sh.msh") ; // Save the Mesh

ofstream file("potential.txt") ; // To direct the output to a file

ifstream file("potential.txt") ; // To get the input from file

mesh Sh=**readmesh**("Sh.msh") ; // Read the Mesh

macro Grad(u)[dx(u),dy(u)] // a macro for to define the vector ∇u

......

example2.edp

```
border a(t=0,2.*pi){x=cos(t);y=sin(t);};
mesh Sh1=buildmesh(a(20));
plot(Sh1,wait=1,cmm="The unit disk");

border b(t=0,2.*pi){x=0.2*cos(t)+0.3;y=sin(t)*0.2+0.3;};
mesh Sh2=buildmesh(a(20)+b(-20));
plot(Sh2,wait=1,cmm="The unit disk with a hole");

mesh Sh3=buildmesh(a(20)+b(20));
plot(Sh3,wait=1,cmm="The unit disk with an inclusion");
```

example3.edp

```
int Dirichlet=1;                                          // For label definition
border a(t=0,2.*pi){x=cos(t);y=sin(t); label=Dirichlet;};
border b(t=0,2.*pi){x=0.2*cos(t)+0.3;y=sin(t)*0.2+0.3; label=Dirichlet;};
mesh Sh=buildmesh(a(80)+b(-20));                          // The Unit Disk with a hole
fespace Vh(Sh,P1);
Vh u,v;
func f=cos(x)*y;
problem Poisson(u,v)=
                 int2d(Sh)(dx(u)*dx(v)+dy(u)*dy(v))
                 -int2d(Sh)(f*v)
                 +on(Dirichlet,u=0);                      // u=0 label=Dirichlet=1
Poisson;
plot(u);
```