© *Jean-Baptiste APOUNG KAMGA <jean-baptiste.apoung@math.u-psud.fr>*

Elliptic and parabolic problems, implementation with **FreeFem++**

Thème - 1 *Model problem and variational formulation*

Let $\Omega \subset \mathbb{R}^2$ be a bounded Lipschitz domain, and f, g_D be two functions in $\mathbb{R}^2 \to \mathbb{R}$. Consider the problem:

Find a function $u : \mathbb{R}^2 \to \mathbb{R}$ such that

$$\left\{ \begin{array}{rcll} -\Delta u = & f & \text{dans} & \Omega, \\ u = & g_D & \text{sur} & \partial\Omega, \end{array} \right. \tag{1}$$

where $\Delta v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$ and $\nabla v = (\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y})$.

Exercice-1 **:** Assume $f \in L^2(\Omega)$, $g_D \in H^{\frac{1}{2}}(\partial\Omega)$, show that the variational formulation of problem (5) is given by:

$$\left\{ \begin{array}{l} \text{Seek } u \in u_D + V \text{ such that} \\ a(u,v) = l(v) \quad \forall v \in V. \end{array} \right. \tag{2}$$

where

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \; dx, \quad l(v) = \int_\Omega f v \; dx, \tag{3}$$

u_D is the continuous lifting of g_D in $H^1(\Omega)$ and V is the vectorial subspace of $H^1(\Omega)$ to be made precise.

Exercice-2 **:** Show the existence and the uniqueness of the solution of the problem (2).

Thème - 2 *Discretization by the P1-Lagrange finite elements*

We wish to approximate the variational problem (2) by using the P1 Lagrange finite element methods.

To this end, we introduce a triangular mesh $\tau_h = \{T_k\}_{1 \le k \le Nma}$ of the domain Ω that is a partition oh Ω made of triangles such that no vertex of one triangle lies in the interior of an edge of another triangle.

For any triangle T, denote by h_T its diameter (i.e its longest edge), and by ρ_T the diameter of the largest circle inscribed in T. The index h in the definition of τ_h is defined by $h = \max_{T \in \tau_h} h_T$.

We introduce the following spaces:

$$\begin{array}{l} H_h = \{v_h \in C^0(\overline{\Omega}) \quad \text{such that} \quad v_h|_{T_k} \in P_1(T_k), \forall T_k \in \tau_h\}, \\ V_h = H_h \cap V. \end{array}$$

Where $P_1(T_k) = \{p$ defined on T_k such that $\exists (a_0, a_1, a_2) \in \mathbb{R}^3 : p(x, y) = a_0 + a_1 x + a_2 y\}$.

The discrete problem reads:

$$\begin{cases} \text{Seek } u_h \in u_{hD} + V_h \text{ such that} \\ a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h, \end{cases} \tag{4}$$

where $u_{hD} \in H_h$ is an interpolate of u_D .

Exercice-1 : Show that V_h is a subspace of V of finite dimension.

Exercice-2 : Show that the problem (4) admits a unique solution.

Exercice-3 : Assuming that the mesh τ_h is regular (i.e $\exists \beta > 0$ such that $\frac{h_T}{\rho_T} < \beta \quad \forall T \in \tau_h$). Show that there exists a constant $C > 0$ independent of h such that

$$\|u - u_h\|_{H^1(\Omega)} \leq Ch|u|_{H^1(\Omega)}.$$

One shall assume the following approximation property of the Pk-Lagrange finite elements:

$$\exists C > 0 \quad \text{such that} \quad \forall v \in H^{l+1}(\Omega) \text{ with } 1 \leq l \leq k, \|v - \pi_h^k v\|_{L^2(\Omega)} + h|v - \pi_h^k v|_{H^1(\Omega)} \leq Ch^{l+1}|u|_{H^{l+1}(\Omega)}.$$

where π_h^k denotes the Pk-Lagrange interpolation operator.

Thème - 3 *Implementation using* ***FreeFem++***

The effective resolution of the problem (4) can be done in two-dimensional and three-dimensional spaces by using the toolboxes **freeFEM** (`http://www.freefem.org/`). More particularly the **FreeFEM++** version (`http://www.freefem.org/ff++/index.htm`). The aims of this toolbox is to let write the finite element code with a syntax as close as possible to that of the variational formulation.

Let us describe the steps to follow in order to solve the problem (4) with the toolbox **FreeFEM++**.

1 Building the mesh of the domain

The construction of the mesh is done in the following steps.

1. **Define the boundary of the domain**: (in 2D) the boundary of the domain is made of a collection of curves. Each curves being defined in a parametric way.

   (a) **Parametric definition of each portion of the boundary:** recall that a parametric curve is be defined by: $[t_i, t_f] \ni t \mapsto \begin{cases} x = f(t) \\ y = g(t) \end{cases}$
   We can define the four portions of the boundary (i.e. the four sides) of the unit square in **FreeFEM++** by:

Listing 1: Parametric definition of a square: File Laplace.edp

```
//CONSTRUCTION DU MAILLAGE
//definition des frontieres du carre
```

```
border abottom(t=0,1){x = t; y = 0; label = 1;};//cote inferieur
border aright(t=0,1) {x = 1; y = t; label = 2;};//cote droit
border atop(t=1,0)   {x = t; y = 1; label = 3;};//cote superieur
border aleft(t=1,0)  {x = 0; y = t; label = 4;};//cote gauche
```

Note that we have assigned a reference (e.g. **label = 1**) to each curve, in order to identify the corresponding portion of the boundary when enforcing the boundary conditions.

(b) **Discretization of each portion and collecting them:** We decide the number of points each curve will receive and we collect all the curves in order to define the boundary of the domain. As illustration, we can draw the boundary of the above unit square by the following program:

Listing 2: Draw the boundary of the unit square: Laplace.edp

```
//maillage et dessin de la frontiere du carre
int n=2;
plot(abottom(10*n) + aright(10*n) + atop(10*n) + aleft(10*n), ↩
    wait=1);
```

2. **Meshing of the enclosed area**: We mesh the enclosed area described by the boundary. This assumes that the boundary, as defined above, must be a closed curve oriented such that the enclosed area remains on our left when we walk round the curve. The following program builds the mesh, plots the built mesh and stores it in an ascii file for further usage.

Listing 3: Generating the mesh. File Laplace.edp

```
//generation du maillage de surface
mesh Th = buildmesh( abottom(10*n) + aright(10*n) + atop(10*n) + ↩
    aleft(10*n) );
//dessin du maillage
plot(Th,wait=1);
//sauvegarde du maillage
savemesh(Th,"carre.msh");//on pourra lire le maillage par: Th =↩
    readmesh("carre.msh");
```

2 Definition of the finite element space

The construction of the finite element space V_h is done as follows:

Listing 4: Finite element space. File: Laplace.edp

```
//CONSTRUCTION DE L'ESPACE ELEMENT FINI
fespace Vh(Th,P1);  //On utilise le maille du carre et l'element fini P1 ↩
    Lagrange
```

Here we have selected the P1-Lagrange finite elements. Replacing **`P1`** by **`P2`** will result in selecting P2-Lagrange finite elements.

3 Definition of the variational problem

The definition of the variational problem closely follows that of the continuous setting. Note how the boundary conditions are introduced. They are indeed introduced by penalization, the penalization coefficient being assigned through the variable and keyword (**tgv**) (*très grosse valeur sur la diagonale* i.e. very big value on the diagonal). The numerical method for solving the linear system is also provided through the keyword **solver**. In the present example the selected method is the Krylov method known as GMRES (Generalized

Minimal ReSidual). Several other methods are available : CG (conjuguate gradient), UMFPACK (direct method for sparse matrices using multi-frontal factorization) . . . .

Listing 5: Problème variationnel fichier: Laplace.edp

```
//DEFINITION DU PROBLEME VARIATIONEL
Vh uh,vh;            // uh est l'inconnu et vh la fonction test
func f  = -4.;       //second membre
func gd = x*x +y*y; //condition de dirichlet

problem Laplace(uh,vh,solver=GMRES,tgv=1e5) =  //definition du èproblme
    int2d(Th)(dx(uh)*dx(vh) + dy(uh)*dy(vh))    //forme bilineaire
    -int2d(Th)(f*vh)                            //forme lineaire
    + on (1,2,3,4, uh = gd);                     //condition aux limite
```

Note that the Dirichlet boundary condition is imposed on all the portion of the boundary. They are four of them with labels (1,2,3,4) see the keyword **on** in the program.

In order to treat cases where the expression of `gd` changes on some portions of the boundary, one can replace the last line by

Listing 6: Condition aux limites fichier: Laplace.edp

```
 + on(1, uh = gd) + on(2, uh = gd) + on(3, uh = gd) + on(4, uh = gd)
```

4 Solving the discrete problem

In order to solve the defined problem, just enter its name followed by a semicolon:

Listing 7: Résolution fichier: Laplace.edp

```
//RESOLUTION DU PROBLEME
Laplace;
```

The linear system is thus solved and the solution is available in the variable **uh** which is the unknown of the problem, namely the first argument of the constructor of the problem.

5 Post-Processing

For post-processing in **FreeFem++**, one can either display graphically the computed solution or compute and print the error, in different norms, between the computed solution and the exact solution if this last one is available.

Listing 8: Post-Processing. Fichier: Laplace.edp

```
//POST PROCESSING
//Dessin de la solution
plot(Th,uh,ps="Laplace.eps", fill=false, value=true,cmm="solution Laplace"↩
    );
//Calcul de l'erreur
Vh errh = uh - (x*x + y*y); //solution exacte
real errL2 = sqrt(int2d(Th)( errh*errh));          //erreur en norme L2
real errH1 = sqrt(int2d(Th)( dx(errh)* dx(errh)    //erreur en seminorm H1
                      + dy(errh) * dy(errh)
                      ));
cout<<"Erreur L2 "<< errL2<<"\n";
```

```
cout<<"Erreur H1 "<< errH1<<"\n";
```

Thème - 4 *Exercises*

Exercice-1 **:** **Boundary conditions of mixed types.** Modify the previous problem such as to solve for a Neumann boundary condition on a part of the boundary .
Recall that to compute integrals on the portion of the boundary identified by its label **`lab`**, *we write* **`int1d(Th,lab)(...)`**.

Exercice-*2* **:** *Compare the solutions obtained using par P1 and P2 Lagrange finite elements.*

Exercice-*3* **:** *Solve the same problem on a unit disk centered at* $(0,0)$.

Thème - *5* *Heat equation*

Consider the heat equation given by (5) where $\partial\Omega = \Gamma^D \cup \Gamma^N$*:*

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla\cdot(K\nabla u) = & f & \textit{dans} &]0,T[\times\Omega, \\ (K\nabla u)\cdot n = & g_N & \textit{sur} & [0,T]\times\Gamma^N, \\ u = & g_D & \textit{sur} & [0,T]\times\Gamma^D, \\ u(0,\cdot) = & u_0 & \textit{sur} & \Omega, \end{cases} \tag{5}$$

Using what was done above, explain how can we solve this problem using **`FreeFem++`**. *Validate the idea on an example.*