2016 Master Vietnam-France in HCMC
High Performance computing

About finite elements and Dirichlet boundary condition

1 Settings

Consider the Poisson equation in Ω with source f, Dirichlet condition on Γ_D equal to g, and Neumann condition on Γ_N equal to h. Let w be a H^1 extension of g , then the variational formulation is, with $a(u,v) = (\nabla u, \nabla v)$, $H^1_D(\Omega) = \{v \in H^1(\Omega), v|_{\Gamma_D} = 0\}$.

$$u \in w + H^1_D(\Omega), \quad \forall v \in H^1_D(\Omega), \quad a(u,v) = (f,v)_0 + (h,v)_{L^2(\Gamma_N)}.$$

Now $\mathcal{T}$ is a triangulation of Ω with nodes $(N_i)_{i\in\mathcal{I}}$. $\mathcal{I}$ is partitioned into $\mathcal{I}_0$ for the interior nodes, $\mathcal{I}_N$ for the nodes on Γ_N, and $\mathcal{I}_D$ for the nodes on Γ_D. The discrete space $V^h = \mathbb{P}_1(\mathcal{T}_h)$, and $V^h_D = V^h \cap H^1_D(\Omega)$. The discrete formulation is (with $f_h \sim f$ and $h_h \sim h$)

$$u_h \in w + V^h_D, \quad \forall v \in V^h_D, \quad a(u_h,v) = (f_h,v)_0 + (h_h,v)_{L^2(\Gamma_N)}.$$

A basis of V_h is the set of ϕ_i for $i \in \mathcal{I}$. Define the discrete approximate extension of g in V^h, to be

$$w_h = \sum_{i\in\mathcal{I}_D} g_i\phi_i, \quad \text{then} \quad u_h = \sum_{i\in\mathcal{I}} u_i\phi_i,$$

with the understanding that $u_i = g_i$ whenever $i \in \mathcal{I}_0$. Choose $v = \phi_j$ for $j \in \mathcal{I} \setminus \mathcal{I}_D$ in the variational formulation to get

$$\forall j \in \mathcal{I} \setminus \mathcal{I}_D, \quad \sum_{i\in\mathcal{I}} u_i a(\phi_i,\phi_j) = (f_h,\phi_j)_0 + (h_h,\phi_j)_{L^2(\Gamma_N)}. \tag{1}$$

Lets expand f_h on the basis of the ϕ_i, $f_h = \sum_{i\in\mathcal{I}} f_i\phi_i$, and omit the Neumann term for the moment.Then we write

$$\forall j \in \mathcal{I} \setminus \mathcal{I}_D, \quad \sum_{i\in\mathcal{I}} u_i a(\phi_i,\phi_j) = \sum_{i\in\mathcal{I}} f_i(\phi_i,\phi_j)_0.$$

Attention, the sum in the RHS is over the full $\mathcal{I}$!

2 Matrix form

Introduce the stiffness and mass matrices $K = (a(\phi_i, \phi_j))_{i,j\in\mathcal{I}}$ and $M = ((\phi_i, \phi_j)_0)_{i,j\in\mathcal{I}}$. Suppose now that the nodes are ordered such that the interior nodes come first, then the Neumann nodes, and then the Dirichlet nodes. Split the matrices and vectors in two blocks, one with interior+Neumann, the second with Dirichlet

$$K = \begin{bmatrix} K^1 & K^{1D} \\ (K^{1D})^T & K^D \end{bmatrix}, \quad M = \begin{bmatrix} M^1 & M^{1D} \\ (M^{1D})^T & M^D \end{bmatrix}, \quad U = \begin{bmatrix} U^1 \\ U^D \end{bmatrix}, \quad F = \begin{bmatrix} F^1 \\ F^D \end{bmatrix}$$

Then the system has the matrix form

$$\begin{bmatrix} K^1 & K^{1D} \end{bmatrix} \begin{bmatrix} U^1 \\ U^D \end{bmatrix} = \begin{bmatrix} M^1 & M^{1D} \end{bmatrix} \begin{bmatrix} F^1 \\ F^D \end{bmatrix} = \tilde{F}_1.$$

or equivalently

$$K^1 U^1 = \tilde{F}_1 - K^{1D} G, \quad \tilde{F}_1 = M^1 F^1 + M^{1D} F^D \tag{2}$$

In order, not to care with the boundary conditions in a first step, and not to reorder, most of the fibite element codes proceed as follows : write the complete system as if no Dirichlet boundary conditions were involved

$$KU = \tilde{F}$$

which in our numbering is

$$\begin{bmatrix} K^1 & K^{1D} \\ (K^{1D})^T & K^D \end{bmatrix} \begin{bmatrix} U^1 \\ U^D \end{bmatrix} = \begin{bmatrix} \tilde{F}^1 \\ \tilde{F}^D \end{bmatrix} \tag{3}$$

Then replace the second equation by $U^D = G$, that is solve the system

$$\begin{bmatrix} K^1 & K^{1D} \\ 0 & I \end{bmatrix} \begin{bmatrix} U^1 \\ U^D \end{bmatrix} = \begin{bmatrix} \tilde{F}^1 \\ G \end{bmatrix} \tag{4}$$

which can be solved by the backslash of matlab, but is not symmetric. The first block-equation is the one we were looking for.

3 matlab script `FEPoisson` of Gander/Kwok

The script FEPoisson does
→ Replace F by $\tilde{F} = MF$.
→ In

$$\begin{bmatrix} K^1 & K^{1D} \\ (K^{1D})^T & K^D \end{bmatrix} \begin{bmatrix} U^1 \\ U^D \end{bmatrix} = \begin{bmatrix} \tilde{F}^1 \\ \tilde{F}^D \end{bmatrix}$$

replace $(K^{1D})^T$ by 0 and K^D by I.
→ replace $\tilde{F}_D$ by G.
which leads to

$$\begin{bmatrix} K^1 & K^{1D} \\ 0 & I \end{bmatrix} \begin{bmatrix} U^1 \\ U^D \end{bmatrix} = \begin{bmatrix} \tilde{F}_1 \\ G \end{bmatrix}$$

Then solve the full system with the matlab "backslash" \. It could be solved alternatively by your favorite **non symmetric** method.

$$K^1 U^1 = \tilde{F}_1 - K^{1D} G \tag{5}$$

Remark The first block system is (2).

4 Symmetrization by Penalization : FEPoissonPenalized

Instead of (3), introduce a large number $\frac{1}{\epsilon}$, and solve

$$\begin{bmatrix} K^1 & K^{1D} \\ (K^{1D})^T & K^D + \frac{1}{\epsilon} I \end{bmatrix} \begin{bmatrix} U^1 \\ U^D \end{bmatrix} = \begin{bmatrix} \tilde{F}^1 \\ \tilde{F}^D + \frac{1}{\epsilon} G \end{bmatrix} \tag{6}$$

5 Symmetrization by Reduction to the interior nodes : FEPoissonreduced

In order to have a symmetric system on the interior nodes, one reduces the system to the interior nodes. For that matlab has a very convenient way of doing. Suppose *int* is a logical vector composed of 1 (true) and zero (false). Extract from a vector b a vector be according to *int* is to write $b(i)$ in be whenever int(i)=1.

Exemple

```
>> lint=logical([1 1 1 0 0])
>> b=[1 2 3 4 5]
>> b(lint)

ans =

     1     2     3

>> b(~lint)

ans =

     4     5
```