Finite Element Method Programming

Praveen. C
praveen@math.tifrbng.res.in

Tata Institute of Fundamental Research
Center for Applicable Mathematics
Bangalore 560065
http://math.tifrbng.res.in/~praveen

March 21, 2013

Assembly process

Find $u_h \in V_h$ such that

$$a(u_h, v_h) = \ell(v_h) \qquad \forall\ v_h \in V_h$$

where

$$a(u, v) = \int_\Omega \ldots \qquad \text{and} \qquad \ell(v) = \int_\Omega \ldots + \int_{\partial\Omega} \ldots$$

Write u_h in terms of Lagrange basis functions

$$u_h = \sum_{j=1}^{N_h} u_j \varphi_j(x)$$

The Galerkin problem can be stated as

$$a(u_h, \varphi_i) = \ell(\varphi_i) \qquad \text{for} \quad i = 1, \ldots, N_h$$

Then the system of Galerkin equations are

$$\sum_{j=1}^{N_h} a(\varphi_j, \varphi_i) u_j = \ell(\varphi_i) \qquad \text{for} \quad i = 1, \ldots, N_h$$

Assembly process

This is a matrix equation

$$AU = b, \qquad A_{ij} = a(\varphi_j, \varphi_i), \qquad b_i = \ell(\varphi_i)$$

Each φ_i has support in a few elements (see figure).

$$A_{ij} = a(\varphi_j, \varphi_i) = \sum_{K \in \mathrm{supp}(\varphi_i) \cap \mathrm{supp}(\varphi_j)} \int_K \ldots = \sum_{K \in \mathrm{supp}(\varphi_i) \cap \mathrm{supp}(\varphi_j)} a_K(\varphi_j, \varphi_i)$$

$$\begin{aligned} b_i = \ell(\varphi_i) &= \sum_{K \in \mathrm{supp}(\varphi_i)} \int_K \ldots + \sum_{e \in \mathrm{supp}(\varphi_i)} \int_{\partial\Omega} \ldots \\ &= \sum_{K \in \mathrm{supp}(\varphi_i)} \ell_K(\varphi_i) + \sum_{e \in \mathrm{supp}(\varphi_i)} \ell_e(\varphi_i) \end{aligned}$$

Assembly process

We can write the following algorithm to compute A and b:

- For $i = 1, 2, \ldots, N_h$
  - Compute $b_i = \ell(\varphi_i)$
  - For $j = 1, 2, \ldots, N_h$
    - Compute $A_{ij} = a(\varphi_j, \varphi_i)$

But this approach requires complicated data structures to know the support information and is not computationally efficient in terms of quadrature.

Assembly process

A better approach is to loop over the elements and assemble the contributions to A and b.

- Set $A = 0$ and $b = 0$
- For each $K \in \mathcal{T}_h$
  - D_K = dofs supported on K
  - For each $i \in D_K$
    - Compute $\ell_K(\varphi_i)$
    - $b_i = b_i + \ell_K(\varphi_i)$
    - For each $j \in D_K$
      - Compute $a_K(\varphi_j, \varphi_i)$
      - $A_{ij} = A_{ij} + a_K(\varphi_j, \varphi_i)$
- For each $e \in \partial\mathcal{T}_h$
  - D_e = dofs supported on e
  - For each $i \in D_e$
    - Compute $\ell_e(\varphi_i)$
    - $b_i = b_i + \ell_e(\varphi_i)$

Remark: If $a(\cdot, \cdot)$ is symmetric, we need to compute only the upper part of the matrix A.

An example

For $\Omega \subset \mathbb{R}^2$, $\partial\Omega = \Gamma_D \cap \Gamma_N$ and given $f \in L^2(\Omega)$, $u_D \in H^{\frac{1}{2}}(\partial\Omega)$, $g \in L^2(\partial\Omega)$, consider

$$\begin{aligned} -\Delta u &= f \quad \text{in} \quad \Omega \\ u &= u_D \quad \text{on} \quad \Gamma_D \\ \frac{\partial u}{\partial n} &= g \quad \text{on} \quad \Gamma_N \end{aligned}$$

Let $u_D \in H^1(\Omega)$ be a lifting of $u_D \in H^{\frac{1}{2}}(\partial\Omega)$ so that we write $u = \tilde{u} + u_D$ with $\tilde{u} \in H_0^1(\Omega)$. The weak formulation is: find $\tilde{u} \in H_0^1(\Omega)$ such that

$$a(\tilde{u}, v) = \ell(v) \qquad \forall\ v \in H_0^1(\Omega)$$

where

$$a(\tilde{u}, v) = \int_\Omega \nabla\tilde{u} \cdot \nabla v \qquad \text{and} \qquad \ell(v) = \int_\Omega fv - \int_\Omega \nabla u_D \cdot \nabla v + \int_{\Gamma_N} gv$$

Galerkin formulation

Let us number the dofs so that those corresponding to Γ_D are at the end

$$\{1, 2, \ldots, M_h, \underbrace{M_h + 1, \ldots, N_h}_{\text{supported on } \Gamma_D}\}$$

Write the Galerkin solution as

$$u_h = \tilde{u}_h + u_{D,h} = \sum_{j=1}^{M_h} u_j \varphi_j + \sum_{j=M_h+1}^{N_h} u_{D_j} \varphi_j$$

Then find $\tilde{u}_h$ such that

$$a(\tilde{u}_h, \varphi_i) = \ell_h(\varphi_i), \qquad i = 1, 2, \ldots, M_h$$

where

$$\ell_h(v) = \int_\Omega fv - \int_\Omega \nabla u_{D,h} \cdot \nabla v + \int_{\Gamma_N} gv$$

Remark: In actual practice, we do not do any renumbering of dofs.

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

coordinates.dat

1	0	0
2	1	0
3	1.59	0
4	2	1
5	3	1.41
6	3	2
7	3	3
8	2	3
9	1	3
10	0	3
11	0	2
12	0	1
13	1	1
14	1	2
15	2	2

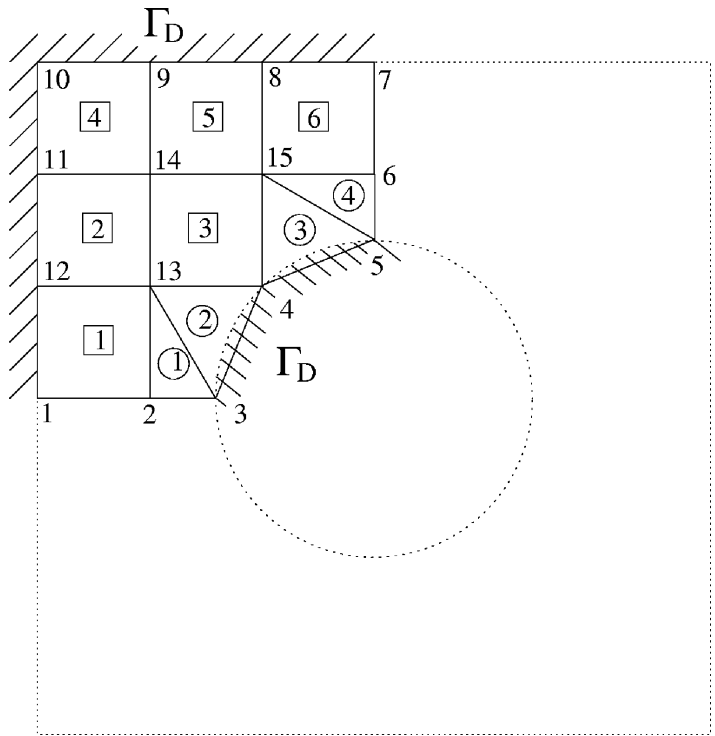


Figure 1. Example of a mesh.

elements3.dat

1	2	3	13
2	3	4	13
3	4	5	15
4	5	6	15

elements4.dat

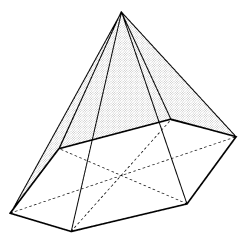
1	1	2	13	12
2	12	13	14	11
3	13	4	15	14
4	11	14	9	10
5	14	15	8	9
6	15	6	7	8

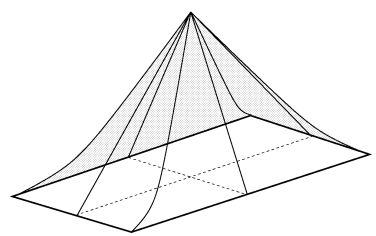
neumann.dat

1	5	6
2	6	7
3	1	2
4	2	3

dirichlet.dat

1	3	4
2	4	5
3	7	8
4	8	9
5	9	10
6	10	11
7	11	12
8	12	1





Jochen Alberty, Carsten Carstensen, Stefan Funken, "Remarks Around 50 Lines of MATLAB: A Short Finite Element Implementation", Numerical Algorithms, Volume 20, Number 2-3, August 1999, pages 117-137.

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

1. In each element, vertices are arranged so that they are ccw
   `elements3(i,1) -> elements3(i,2) -> elements3(i,3)`
2. For each edge, nodes are numbered so that domain is on the left when we go from `dirichlet(i,1) -> dirichlet(i,2)` and `neumann(i,1) -> neumann(i,2)`

List of programs:

- `fem_50.m` : Main program
- `stima3.m` : local stiffness matrix on triangle
- `stima4.m` : local stiffness matrix on parallelogram
- `f.m` : right hand side function in Laplace equation
- `u_d.m` : Dirichlet boundary values
- `g.m` : Neumann boundary values

You can also get the code from here:
`http://people.sc.fsu.edu/~jburkardt/m_src/fem_50/fem_50.html`

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

Assembling stiffness matrix on a triangle:

$$\varphi_j = \det \begin{bmatrix} 1 & x & y \\ 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \end{bmatrix} / \det \begin{bmatrix} 1 & x_j & y_j \\ 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \end{bmatrix}$$

$$\nabla\varphi_j = \frac{1}{2|K|} \begin{bmatrix} y_{j+1} - y_{j+2} \\ x_{j+2} - x_{j+1} \end{bmatrix} = \text{const on } K, \quad |K| = \frac{1}{2} \det \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}$$

$$\begin{aligned} A_{jk}^K &= a_K(\varphi_k, \varphi_j) = \int_K (\nabla\varphi_j)^\top (\nabla\varphi_k) \mathrm{d}x \\ &= \frac{|K|}{(2|K|)^2} [y_{j+1} - y_{j+2},\ x_{j+2} - x_{j+1}] \begin{bmatrix} y_{k+1} - y_{k+2} \\ x_{k+2} - x_{k+1} \end{bmatrix} \end{aligned}$$

$$A^K = \frac{|K|}{2} G G^\top \qquad \text{where} \qquad G = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

Call `stima3` with

$$\texttt{vertices} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}$$

```
function M = stima3 ( vertices )
d = size ( vertices, 2 );
G = [ ones(1,d+1); vertices' ] \ [ zeros(1,d); eye(d) ];
M = det([ ones(1,d+1); vertices' ]) * G * G' / prod(1:d);
return
```

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

Assembling stiffness matrix on a parallelogram:

Affine transformation

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = F_K(\hat{\mathbf{x}}) = \begin{bmatrix} x_2 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_4 - y_1 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = B_K \hat{\mathbf{x}} + b_K$$

Basis functions on reference element

$$\hat{\varphi}_1 = (1-\hat{x})(1-\hat{y}), \quad \hat{\varphi}_2 = \hat{x}(1-\hat{y}), \quad \hat{\varphi}_3 = \hat{x}\hat{y}, \quad \hat{\varphi}_4 = (1-\hat{x})\hat{y}$$

$$\begin{aligned} A_{jk}^K &= \int_K (\nabla\varphi_j)^\top (\nabla\varphi_k) \mathrm{d}\mathbf{x} = \int_{\hat{K}} (B_K^{-1}\nabla_{\hat{\mathbf{x}}}\hat{\varphi}_j)^\top (B_K^{-1}\nabla_{\hat{\mathbf{x}}}\hat{\varphi}_k) \det(B_K) \mathrm{d}\hat{\mathbf{x}} \\ &= \det(B_K) \int_{\hat{K}} (\nabla_{\hat{\mathbf{x}}}\hat{\varphi}_j)^\top (B_K^\top B_K)^{-1} (\nabla_{\hat{\mathbf{x}}}\hat{\varphi}_k) \mathrm{d}\hat{\mathbf{x}} \end{aligned}$$

$$(B_K^\top B_K)^{-1} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

$$\nabla\hat{\varphi}_1 = \begin{bmatrix} -(1-\hat{y}) \\ -(1-\hat{x}) \end{bmatrix}, \quad \nabla\hat{\varphi}_2 = \begin{bmatrix} 1-\hat{y} \\ -\hat{x} \end{bmatrix}, \quad \nabla\hat{\varphi}_3 = \begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix}, \quad \nabla\hat{\varphi}_4 = \begin{bmatrix} -\hat{y} \\ 1-\hat{x} \end{bmatrix}$$

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

Local stiffness matrix

$$A^K = \frac{\det(B_K)}{6} \begin{bmatrix} 3b+2(a+c) & -2a+c & -3b-(a+c) & a-2c \\ -2a+c & -3b+2(a+c) & a-2c & 3b-(a+c) \\ -3b-(a+c) & a-2c & 3b+(a+c) & -2a+c \\ a-2c & 3b-(a+c) & -2a+c & -3b+2(a+c) \end{bmatrix}$$

Function `stima4` returns the matrix A^K

```
function M = stima4 ( vertices )
  D_Phi = [ vertices(2,:) - vertices(1,:); ...
            vertices(4,:) - vertices(1,:) ]';
  B = inv ( D_Phi' * D_Phi );
  C1 =  [  2, -2; -2,  2 ] * B(1,1) ...
      + [  3,  0;  0, -3 ] * B(1,2) ...
      + [  2,  1;  1,  2 ] * B(2,2);
  C2 =  [ -1,  1;  1, -1 ] * B(1,1) ...
      + [ -3,  0;  0,  3 ] * B(1,2) ...
      + [ -1, -2; -2, -1 ] * B(2,2);
  M = det ( D_Phi ) * [ C1 C2; C2 C1 ] / 6;
end
```

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

Assembling right hand side:

Mid-point rule for integration over K

$$(x_K, y_K) = \text{centroid of triangle } K$$

$$\ell_K(\varphi_j) = \int_K f\varphi_j \approx f(x_K, y_K) \int_K \varphi_j = \frac{1}{C_K} \det \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} f(x_K, y_K)$$

For a triangle $C_K = 6$ and for a parallelogram $C_K = 4$. If j, k, l are vertices of K then $\ell_K(\varphi_j) = \ell_K(\varphi_k) = \ell_K(\varphi_l)$ due to mid-point rule.

Mid-point rule for integration over boundary edges

$$(x_e, y_e) = \text{center of edge } e$$

$$\ell_e(\varphi_j) = \int_e g\varphi_j \mathrm{d}s \approx g(x_e, y_e) \int_e \varphi_j \mathrm{d}s = \frac{|e|}{2} g(x_e, y_e)$$

If j, k are the vertices of edge e then $\ell_e(\varphi_j) = \ell_e(\varphi_k)$ due to mid-point rule.

Example of $\mathbb{P}_1/\mathbb{Q}_1$ Galerkin method

Applying Dirichlet boundary condition: We assemble the stiffness matrix A for all dofs. Then the equations corresponding to Dirichlet boundary are deleted; this corresponds to removing the rows of A. We also remove the columns corresponding to Dirichlet dofs. E.g., equation corresponding to φ_2 is

$$A_{2,2}u_2 + A_{1,2}u_{D_1} + A_{3,2}u_{D_3} + A_{12,2}u_{D_{12}} + A_{13,2}u_{13} = b_2$$

$$A_{2,2}u_2 + A_{13,2}u_{13} = b_2 - (A_{1,2}u_{D_1} + A_{3,2}u_{D_3} + A_{12,2}u_{D_{12}})$$

- $(1 : N_h)$ = total number of dofs = total number of vertices
- `BoundNodes` = indices of vertices on Dirichlet boundary
- `FreeNodes` = $(1 : N_h)$ `- BoundNodes`
- `u` = zeros$(1 : N_h)$, then fill in the Dirichlet boundary values
  `u(BoundNodes) = u_d(coordinates(BoundNodes, :))`
- Modify the right hand side vector `b = b - A * u`
- Then solve for the unknown dofs
  `u(FreeNodes) = A(FreeNodes, FreeNodes) \ b(FreeNodes)`