FEM09 - lecture 3

Johan Jansson

jjan@csc.kth.se

CSC

KTH

Outline

- Quadrature
- FEM in 2D/3D
- General assembly algorithm
- Boundary conditions
- FEniCS automated FEM software
- Modules (deadline)

Computer demonstration

Feedback

Integration/Quadrature in 1D

Midpoint rule

$$\int_a^b f(x)\,dx = \sum_{i=1}^{m+1} f(\frac{x_{i-1}+x_i}{2})h_i + E(f)$$

$$|E(f)| \leq \sum_{i=1}^{m+1} \frac{1}{12} h_i^2 \max_{[x_{i-1},x_i]} |f''| h_i.$$

In other words: the rule can integrate linear polynomials exactly.

Possible to generate quadrature rules for any order of (polynomial) accuracy.

Integration/Quadrature in 2D

Midpoint rule

$$\int_K f(x)\,dx = \sum_{1\le i<j\le 3} f\left(a_K^{ij}\right)\frac{|K|}{3} + E(f)$$

$$|E(f)| \le \sum_{|\alpha|=3} C h_K^3 \int_K |D^\alpha f(x)|\,dx$$

In other words: the rule can integrate quadratic polynomials in 2D exactly.

Possible to generate quadrature rules for any order of (polynomial) accuracy in 2D/3D as well.

Piecewise polynomials in 2D

Construct triangulation T of domain Ω

Define size of triangle $K \in T$ is h_K as diameter of triangle

Define N as node (in this case vertex of triangle)

Want to define basis functions for vector space V_h: space of piecewise linear functions on T

Requirement for nodal basis:

$$\phi_j(N_i) = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases} \quad i, j = 1, ..., M \tag{1}$$

Piecewise polynomials in 2D

Define local basis functions v^i on triangle K with vertices $a^i = (a_1^i, a_2^i)$, $i = 1, 2, 3$

v is linear $\Rightarrow v(x) = c_0 + c_1 x_1 + c_2 x_2$

Values of v in vertices: $v_i = v(a^i)$ (1 or 0)

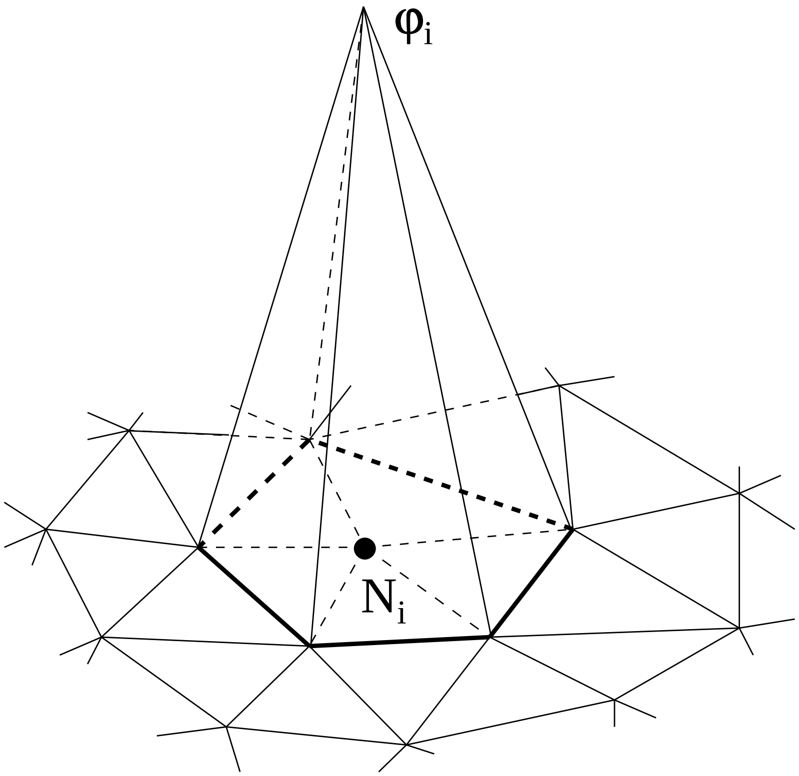
Linear system for coefficients c:

$$\begin{pmatrix} 1 & a_1^1 & a_2^1 \\ 1 & a_1^2 & a_2^2 \\ 1 & a_1^3 & a_1^3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

Piecewise polynomials in 2D

Sum local basis functions:

$$\phi_i = \sum_j v^j, \quad N_i = a_j \tag{2}$$



Poisson in 2D

Automated discretization in FEniCS

We will look at a general algorithm for FEM assembly of matrix/vector.

This algorithm is implemented in FEniCS (software used in the course).

Later in the course we will implement a simple variant of this ourselves.

General bilinear form $a(\cdot, \cdot)$

In general the matrix A_h, representing a bilinear form

$$a(u,v) = (A(u), v),$$

is given by

$$(A_h)_{ij} = a(\varphi_j, \hat{\varphi}_i).$$

and the vector b_h representing a linear form

$$L(v) = (f, v),$$

is given by

$$(b_h)_i = L(\hat{\varphi}_i).$$

Example (Poisson 1D):

$a(u,v) = (u', v') = \int_0^1 u'v'dx, \quad (A_h)_{ij} = a(\phi_j, \phi_i) = \int_0^1 \phi_j' \phi_i' dx$

$L(v) = (f, v) = \int_0^1 f v dx, \quad (b_h)_i = L(\phi_i) = \int_0^1 f \phi_i dx$

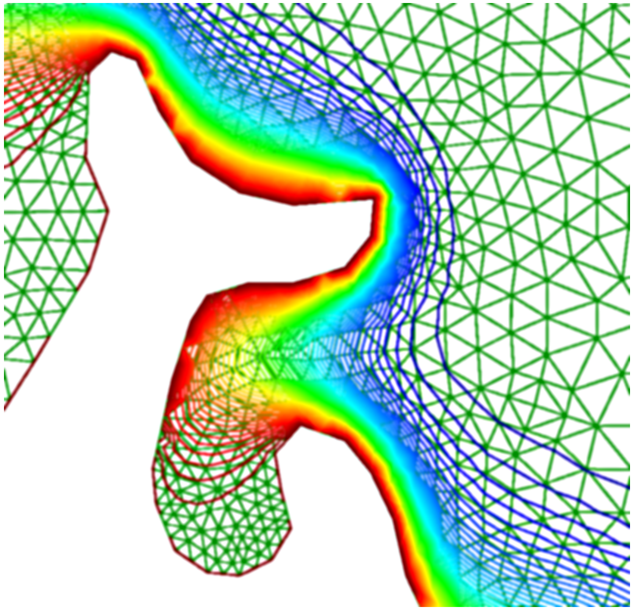
Computing $(A_h)_{ij}$

Note that

$$(A_h)_{ij} \quad = \quad a(\varphi_j, \hat{\varphi}_i) = \sum_{K \in \mathcal{T}} a(\varphi_j, \hat{\varphi}_i)_K.$$

Iterate over all elements K and for each element K compute the contributions to all $(A_h)_{ij}$, for which φ_j and $\hat{\varphi}_i$ are supported within K.

Assembly of discrete system



Noting that $a(v,u) = \sum_{K \in \mathcal{T}} a_K(v,u)$, the matrix A can be assembled by

$A = 0$
`for all elements` $K \in \mathcal{T}$
    A += A^K

The *element matrix* A^K is defined by

$$A_{ij}^K = a_K(\hat{\phi}_i, \phi_j)$$

for all local basis functions $\hat{\phi}_i$ and ϕ_j on K

Assembling A_h

for all elements $K \in \mathcal{T}$

for all test functions $\hat{\varphi}_i$ on K

for all trial functions φ_j on K

1. Compute $I = a(\varphi_j, \hat{\varphi}_i)_K$
2. Add I to $(A_h)_{ij}$

end

end

end

Assembling b

for all elements $K \in \mathcal{T}$

for all test functions $\hat{\varphi}_i$ on K

1. Compute $I = L(\hat{\varphi}_i)_K$
2. Add I to b_i

end

end

Computer demonstration

Boundary conditions

Essential BC:

- Homogenous Dirichlet BC: $u(0) = 0$
  Enforce in function space:
  $V = \left\{ v : \int_0^1 v^2 \, dx < C, \; \int_0^1 (v')^2 \, dx < C, \; v(0) = 0 \right\}$

Natural BC:

- Neumann BC: $-a(0)u'(0) = g_N$
- Robin BC: $-a(0)u'(0) + \gamma(u(0) - g_D) = g_N$
  γ is a penalty parameter, with $\gamma = 0 \Rightarrow$ Neumann and $\frac{1}{\gamma} = 0 \Rightarrow$ Dirichlet

Enforce in weak form:
$\int_0^1 (au')v' - fv dx + au'(1)v(1) - au'(0)v(0) = 0$

Computer demonstration