Università della Svizzera italiana

Facoltà di scienze informatiche

Software Atelier: Differential Equations **Academic Year 2016/2017**

Instructor: Dr. Drosos Kourounis TA: Hardik Kothari

Assignment 4 - FEM Solution of Poisson's equation in 3D

Name: Juraj Kardos
Discussed with:

1. The Problem

We seek the discrete solution of Poisson's equation

$$-\nabla^2 u(x,y,z) = f(x,y,z),\ (x,y,z) \in \Omega \tag{1}$$

$$\frac{\partial u}{\partial n} = 0, \quad y = 1, z = 1, \tag{2}$$

$$u(x,y,z) = u_0(x,y,z), \quad \text{otherwise.} \tag{3}$$

We assume unit cube domain $\Omega = [0,1]^3$ and use quadrilateral $N_x \times N_y \times N_z$ grid of trilinear hexahedron elements.

We want the exact solution of the PDE to be

$$u_0(x,y) = xe^{-(y-1)^2(z-1)^2}. \tag{4}$$

so we can compute $f(x,y)$ as following:

```
syms x;
syms y;
syms z;
u = @(x,y,z) x * exp(- (y-1)^2 * (z-1)^2 );

u_xx = diff(u,x,2);
u_yy = diff(u,y,2);
u_zz = diff(u,z,2);

f = -(u_xx + u_yy + u_zz);
```

2. FEM Solution

2.1. Mesh Generation

The solution starts with generation of the mesh that approximates our domain. We assume unit square with non-overlapping elements. The domain is discretized by N_x by N_y by N_z grid of nodes. The elements are defined by enumerating their vertices, we follow convention when enumerating the corners of the hexahedra element using VTK convention for element of type `VTK_HEXAHEDRON`. For example, the bottom face of a single element would look like following:

```
elements(id_elem,:) = [e, e+1, e+N_x+1, e+N_x,... ]
```

2.2. Assembly of Discrete Operators

Next step in FEM is assembly of discrete operators, namely mass matrix M, laplacian matrix K and discretized RHS b. The idea of the assembly is to construct the local versions of the matrices and insert them into the global structure.

2.2.1. Mass Matrix

Trilinear hexahedral element is defined by the following nodal functions:

$$N_1 = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta), \tag{5}$$

$$N_2 = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta), \tag{6}$$

$$N_3 = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta), \tag{7}$$

$$N_4 = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta), \tag{8}$$

$$N_5 = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta), \tag{9}$$

$$N_6 = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta), \tag{10}$$

$$N_7 = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta), \tag{11}$$

$$N_8 = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta). \tag{12}$$

In order to compute integral over the domain of the element, we first project the hexahedral element E wit coordinates $(x_i, y_i, z_i), i \in [0, \ldots, 7]$ into reference element E_0 with coordinates (ξ, η, ζ)in rage $[-1, 1]$. The mapping is

$$x = x_i + \frac{dx}{2}(\xi + 1), \tag{13}$$

$$y = y_i + \frac{dy}{2}(\eta + 1), \tag{14}$$

$$z = z_i + \frac{dz}{2}(\zeta + 1). \tag{15}$$

For example, for reference coordinates of node $(\xi, \eta, \zeta) = (-1, -1, -1)$ we get coordinates of the node (x_0, y_0, z_0) and for $(\xi, \eta, \zeta) = (1, -1, -1)$ we get node $(x_0 + dx = x_1, y_0, z_0)$.

We define Jacobian of this transformation with respect to the reference coordinate frame (ξ, η, ζ), we get

$$J = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{pmatrix} = \begin{pmatrix} \frac{dx}{2} & 0 & 0 \\ 0 & \frac{dy}{2} & 0 \\ 0 & 0 & \frac{dz}{2} \end{pmatrix}. \tag{16}$$

Computing integral over the reference integral is much easier, but we need to consider the transformation we did. This gives us

$$m_{i,j} = \int_E N_i(x, y, z) N_j(x, y, z) dE \tag{17}$$

$$m_{i,j} = \int_{E_0} J^{-1} N_i(x, y, z) J^{-1} N_j(x, y, z) |J| dE_0 \tag{18}$$

$$m_{i,j} = \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} J^{-1} N_i(\xi, \eta, \zeta) J^{-1} N_j(\xi, \eta, \zeta) |J| d\xi d\eta d\zeta \tag{19}$$

$$m_{i,j} \approx \sum_{k_1=1}^{n} \sum_{k_2=1}^{n} \sum_{k_3=1}^{n} h_1 h_2 h_3 J^{-1}_{k1,k2,k3} N_i(\xi_{k1}, \eta_{k2}, \zeta_{k3}) J^{-1}_{k1,k2,k3} N_j(\xi_{k1}, \eta_{k2}, \zeta_{k3}) |J_{k1,k2,k3}| \tag{20}$$

The code implementing this is as following:

Listing 1. Local assembly of mass matrix

```
xi = sym('xi', 'real');
eta = sym('eta', 'real');
zeta = sym('zeta', 'real');

dx = sym('dx', 'real');
dy = sym('dy', 'real');
dz = sym('dz', 'real');

%jacobian of the transformation
J(1,1) = 0.5*dx;
J(2,2) = 0.5*dy;
J(3,3) = 0.5*dz;

%nodal functions
c=[-1 -1 -1; 1 -1 -1; 1 1 -1; -1 1 -1; -1 -1 1; 1 -1 1; 1 1 1; -1 1 1];
for i=1:8
    N(i) = 1/8*( 1+c(i,1)*xi )*( 1+c(i,2)*eta )*( 1+c(i,3)*zeta );
end

%gaussian quadrature
F = det(J) * N' * N;
M = int(int(int(F, 'xi', -1, 1), 'eta', -1, 1), 'zeta', -1, 1);
```

2.2.2. Laplacian matrix

In a similar manner we construct the local Laplacian matrix. The integral over the element becomes

$$k_{ij} = \int_E \nabla N_i(x,y,z) \nabla N_j(x,y,z) dE \tag{21}$$

$$k_{ij} = \int_{E_0} J^{-1} \nabla N_i(\xi,\eta,\zeta) J^{-1} \nabla N_j(\xi,\eta,\zeta) |J| dE_0 \tag{22}$$

$$k_{ij} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} J^{-1} \nabla N_i(\xi,\eta,\zeta) J^{-1} \nabla N_j(\xi,\eta,\zeta) |J| d\xi d\eta d\zeta \tag{23}$$

$$k_{ij} \approx \sum_{k1=1}^{n}\sum_{k2=1}^{n}\int_{k3=1}^{n} h_1 h_2 h_3 J^{-1}_{k1,k2,k3} \nabla N_i(\xi_{k1},\eta_{k2},\zeta_{k3}) J^{-1}_{k1,k2,k3} \nabla N_j(\xi_{k1},\eta_{k2},\zeta_{k3}) |J_{k1,k2,k3}| d\xi d\eta d\zeta \tag{24}$$

Listing 2. Local assembly of Laplatian matrix

```
xi = sym('xi', 'real');
eta = sym('eta', 'real');
zeta = sym('zeta', 'real');

dx = sym('dx', 'real');
dy = sym('dy', 'real');
dz = sym('dz', 'real');

%jacobain of the transformation
J(1,1) = 0.5*dx;
J(2,2) = 0.5*dy;
J(3,3) = 0.5*dz;

%nodal functions
c=[-1 -1 -1; 1 -1 -1; 1 1 -1; -1 1 -1; -1 -1 1; 1 -1 1; 1 1 1; -1 1 1];
for i=1:8
    N(i) = 1/8*( 1+c(i,1)*xi )*( 1+c(i,2)*eta )*( 1+c(i,3)*zeta );
end

%gradient of nodal funcitons
Nx = diff(N, 'xi');
Ny = diff(N, 'eta');
Nz = diff(N, 'zeta');

%gaussian quadrature
dN = [Nx; Ny; Nz];
Jd = inv(J) * dN;
F = det(J)*Jd'*Jd;
I = int(int(int(F, 'xi', -1, 1), 'eta', -1, 1), 'zeta', -1, 1);
```

After forming the local mass and Laplacian matrix for each element, we need to insert these into the global matrices M and K. We do not insert elements one by one but we keep track of local indices and corresponding global indices in format of Matlab sparse matrix and we create matrix after all of the indices are available.

2.3. Boundary Conditions

We need to modify the equations for the points that lie on the boundary of our domain. The Dirichlet boundary specifies exact value of the target profile, that is $u(x,y,z) = u_0(x,y,z)$ for every $(x,y) \in \partial\Omega$ except for $y = 1, z = 1$ where Neumann BC apply.

Neumann BC specify not the target profile of the domain, but the rate of the change of the domain. Basically, a Dirichlet BC means imposing the value for the temperature (or some other physical quantity) in one edge of

the dimensional domain (a metal bar, for example), i.e. fixing the value for the temperature for the solution at that edge while Newmann BC mean to impose the flux of heat through that edge. If Newmann BC is set to zero (homogeneous BC) it means that the edge of the bar is isolated and no flux of heat enters or outputs the bar.

To verify that NBC hold we should check that the following relations are satisfied:

$$\frac{\partial u}{\partial n_{y=1}} = 0 \tag{25}$$

$$\frac{\partial u}{\partial n_{z=1}} = 0 \tag{26}$$

The unit normal n at the faces of the cube where $y = 1$ and $z = 1$ is simply y and z respectively, so we need to do partial derivatives. In this case the BC are homogeneous, that is they are equal to 0 so they do not influence the solution. But if this was not the case we would have to adjust the RHS term at appropriate positions (after deriving the weak formulation Neumann BC show up on RHS).

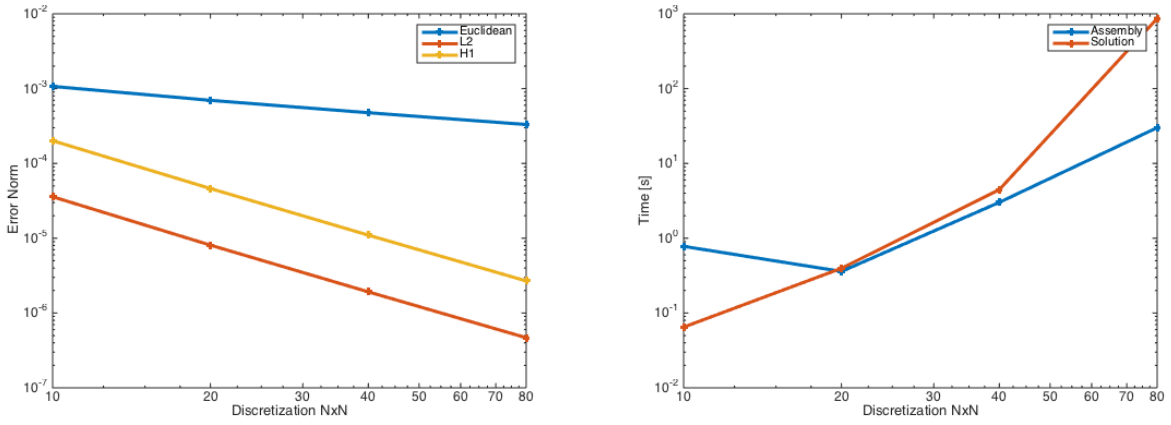
```
% impose boundary conditions for Dirichlet boundaries
I = find(mesh.PointMarkersDiri);

%adjust K
K(I,:) = 0;
for i = 1:length(I);
  K(I(i),I(i)) = 1.0;
end

%adjust RHS
X = mesh.Points(I,1);
Y = mesh.Points(I,2);
Z = mesh.Points(I,3);
b(I) = u0(X,Y,Z);
```

2.4. Solution

Having the system ready at hand, we need to solve it. In vector u we will have approximate solution to our problem.

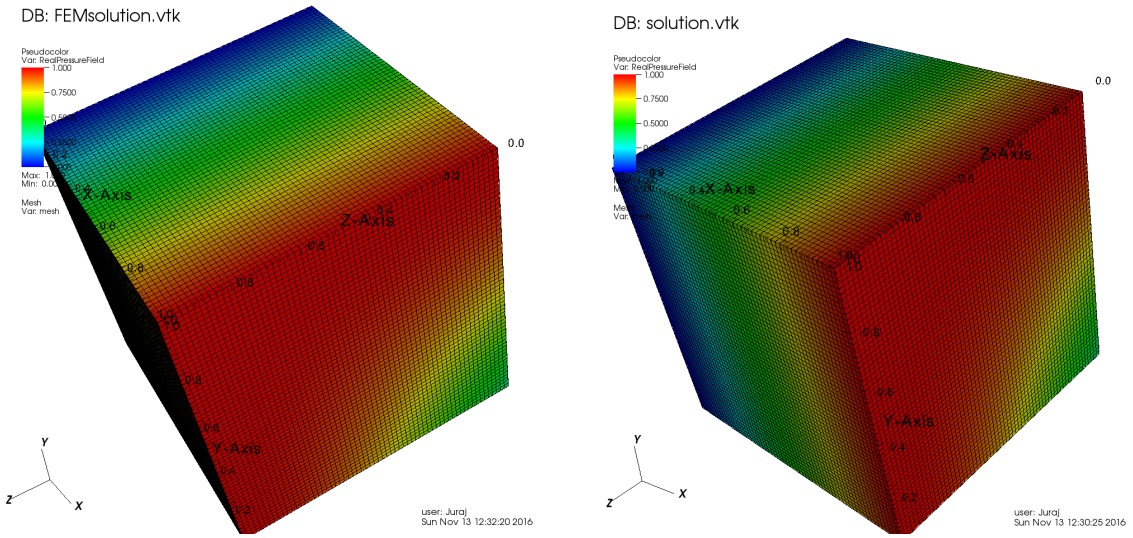
```
% solve
u = K\b;

% visualize solution
writeMeshToVTKFile(mesh, x, 'solution')
```

2.5. Visualization of the Solution



(a) Error norms *(b)* Solution time

Figure 1. Discretizations `10,20,40,80`



(a) FEM solution for $N = 60$ *(b)* Exact solution

Figure 2. Solution of u