Università della Svizzera italiana

Facoltà di scienze informatiche

Introduction to PDEs (LAB) **Academic Year 2015/2016**

Instructor: Prof. Rolf Krause Dr. Drosos Kourounis

Assignment 6 - Finite Element Solution of Poisson's equation in 3D

Due date: Thursday 29 October 2015, 10:30

Solution of Poisson's equation

We seek the discrete solution of Poisson's equation

$$\begin{aligned} -\nabla^2 u(x,y,z) &= f(x,y,z), \quad \in \Omega \\ \frac{\partial u}{\partial n} &= 0, \quad y = 1, z = 1, \\ u(x,y,z) &= u_0(x,y,z), \text{otherwise}, \end{aligned}$$

where $\Omega = [0,1]^3$. We discretize the domain Ω using a quadrilateral $N_x \times N_y \times N_z$ grid of trilinear elements.

1. Find the analytical expression of $f(x,y)$ so that the exact solution of the PDE is

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

2. Verify that the exact solution satisfies the homogeneous Neumann conditions at $y = 1$ and at $z = 1$.
3. Implement the solution in MATLAB following the steps described below.

1. Mesh generation

Modify the mesh generation routine you already have, so that it assumes three input arguments for the cube dimensions L_x, L_y, L_z and three for the grid size N_x, N_y, N_z. It outputs the hexahedral or tetrahedral mesh of the cube.

2. Finite element assembly

Use the MATLAB implementation of the fast assembly provided with the Assignment (in the git repository) `FastAssemblyLhs.m`, `FastAssemblyRhs.m` and replace the slow assembly code you used for Assignment 5 with the new code. The code will not work out of the box, it needs to be adapted to your own mesh structure. Once you finish with the adaptations and the code works,

- Make sure that the results are the same (error norms decrease as previously).
- Create a table containing 4 columns. The first column should be $N = 10, 20, 40, 80$. The rest should be the time needed for the slow assembly, fast assembly and the solution of the linear system. This table will be populated with higher values of the index N in future assignments. So, enjoy ...

`FastAssemblyRhs.m`

```
function b = assembleRhs(b, Mesh)

    assembly_time_start = tic;

    X = Mesh.Points(:,1);
    Y = Mesh.Points(:,2);
    f = source(X, Y);
    I = Mesh.Elements';
    F = f(I);

    Me = makeLocalMass(1, Mesh);
    b = Me*F;
    I = I(:);
    J = ones(Mesh.NumberOfVertices*Mesh.NumberOfElements, 1);

    % here we create a sparse matrix of size n x 1, which is practically
    % a sparse vector. We do it this way because the scattered FE vector
    % contributed by each element will be assembled on a global vector
    % once it becomes a sparse matrix. Then using the method 'find' we
    % extract back our vector held in obj.b

    B = sparse(I, J, b(:), Mesh.NumberOfNodes, 1);
    b=full(B);

    assembly_time = toc(assembly_time_start);
    fprintf('ASSEMBLY RHS TIME: %10.2f\n', assembly_time);
end
```

FastAssemblyLhs.m

```
function [A, M, L] = assembleLhs(A, M, L, Mesh)

    assembly_time_start = tic;

    % Local element mass matrix
    Me= makeLocalMass(1, Mesh);

    % Local element diffusion matrix
    Le= makeLocalLaplacian(1, Mesh);

    % Element matrices unfolded as a column vector column by column
    me    = Me(:);
    le    = Le(:);

    n     = Mesh.NumberOfNodes;
    nel   = Mesh.NumberOfElements;
    nv    = Mesh.NumberOfVertices;

    % We repeat this vectors as many times as the number of elements
    Ms     = repmat(me, nel, 1);
    Ls     = repmat(le, nel, 1);

    i=1:nv;
    j=ones(1, nv);

    % ig = [1..nv, 1..nv, ... 1..nv] nv times
    ig=repmat(i, 1, nv);

    % jg = [1 1 ... 1, 2 2 ... 2, ... , nv nv ... nv] each number
    % repeated nv times
    jg=repmat(1:nv, nv, 1);

    %  (:) produces a column and we need a row of indices
    jg=jg(:)';


    % [1 1 ... 1, 2 2 ... 2, ... , nv nv ... nv ] with each number
    % repeated nv^2 times, these will be the indices to choose the
    % weights for each element matrix (diffusion coefficients), place
    % them in an array and multiply the arrays with the corresponding
    % matrix
    cg  = repmat(1:nel, nv^2, 1);

    Is  = Mesh.Elements(:, ig)';
    Js  = Mesh.Elements(:, jg)';

    Operators.L = sparse(Is(:), Js(:), Ls, n, n);
    Operators.M = sparse(Is(:), Js(:), Ms, n, n);

    assembly_time = toc(assembly_time_start);
    fprintf('ASSEMBLY LHS TIME: %10.2f\n', assembly_time);

end
```