Università della Svizzera italiana

Facoltà di scienze informatiche

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

Instructor: Prof. Rolf Krause TA: Drosos Kourounis

Assignment 3 - Finite Element Solution of Poisson's equation in 2D

Due date: Thursday 15 October 2015, 10:30

Solution of Poisson's equation

We seek the discrete solution of Poisson's equation

$$\begin{aligned} -\nabla^2 u(x,y) &= f(x,y), \ \in \Omega \\ u &= u_0(x,y), \ \in \partial\Omega \end{aligned}$$

where $\Omega = [0,1]^2$.

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

$$u_0(x,y) = 10x + \tanh(10x - 10)$$

2. Write MATLAB code that solves the PDE implementing the steps following[1]

1. Mesh generation

Write a MATLAB function that generates a $N_x \times N_y$ quadrilateral grid. The input arguments of the function should be `N_x`, `N_y`, `grid_type`. If third argument `grid_type` is equal to `'quadrilaterals'` then you do not split each quadrilateral to 2 triangles as you did in the previous Assignment. The output of the function should be the matrix of the connectivity of the grid (`Elements`) and the matrix of the coordinates of the grid points (`Points`). The matrix `Elements` should by of size $N_e \times N_\mathrm{v}$, where the N_e number of elements and N_v the number of vertices. The matrix `Points`, should be of size $N \times d$, where N the number of points and d the number of space dimensions.

```
function [mesh] = makeGrid(L_x, L_y, N_x, N_y, grid_type)
```

[1]You should include in your submission only the implementations of the functions marked with red and not the whole MATLAB code.

The output mesh is a MATLAB structure that contains all the information that are needed for the mesh generation and mesh description, namely, delta_x, delta_y, N_x, N_y, L_x, L_y, N_e, N_v, N, Elements, Points, PointMarkers. The parameters L_x,L_y represent the length and height of the domain at each direction. Here both are equal to 1, since our domain $\Omega = [0,1]^2$. However, it does not hurt to write the code for the general case.

The array PointMarkers should be 0 at every point except those points that belong to the boundary where they should be 1. We will need it later to enforce Dirichlet boundary conditions to our Discrete system of equations. Use the MATLAB function writeMeshAsVTKFile.m in the MATLAB folder and call it passing the structure mesh and the name of the output file. This will dump the grid as a VTK file that you can visualize using visualization package ViSiT. The declarations follow.

```
function writeMeshAsVTKFile(mesh, vtkfile)
```

2. Finite element assembly

Use the following code sample for the assembly of your discrete operators.

```
function [M,K,b] = assembleDiscreteOperators(mesh)
  N   = mesh.N;
  Ne  = mesh.N_e;
  M   = zeros(N,N); K = zeros(N,N);
  b   = zeros(N,1);
  for e=1:N_e
    Me = makeMe(e, mesh);
    Ke = makeKe(e, mesh);
    fp = makebe(e, mesh);
    I = mesh.Elements(e, :);
    M(I, I) = M(I, I) + Me;
    K(I, I) = K(I, I) + Ke;
    b(I)    = b(I) + Me*fp;
  end % e loop
end
```

Provide implementations of the individual functions needed for forming the element mass matrix M_e, element Laplacian K_e and the element rhs b_e. Use the provided MATLAB routines Ksymbolic.m, Msymbolic.m that can be found in the MATLAB folder. These routines will compute the mass and stiffness matrices only for the particular grid type. The result will be wrong in the general case where we need to use quadrature.

```
function Me = makeMe(e, mesh);
function Ke = makeKe(e, mesh);
function fp = makebe(e, mesh);
```

Keep in mind that f_p should hold the values of the function $f(x,y)$ evaluated at the nodes of the eth element, for example $f_p = [f(x_1^e, y_1^e)\ f(x_2^e, y_2^e)\ f(x_3^e, y_3^e)\ f(x_4^e, y_4^e)]^T$.

3. Solution

Provide a function that solves the linear system $Ku_h = b$, and use it to obtain the solution of the linear system you constructed in the previous step. Provide a function that enforces Dirichlet boundary conditions for all the points that have been marked with 1. For the particular exercise you can stop the loop at the first point of the grid, since we want the solution to satisfy Neumann conditions on the boundary. Dirichlet conditions will destroy the symmetry of the matrix. Can you modify the system such that the symmetry is preserved without screwing up the solution?

```
function x = solve(K, b)
```

4. Convergence study

Compare the solution you obtained with the exact one in the first part $u_0(x, y)$ by computing the L^2 and H^1 norms of the error and plot them using logarithmic scale on both axes as a function of $h = \max(\delta x, \delta y)$ for $N_x = N_y = 5, 10, 20, 40, 80, 160$. Note that the discrete L^2 and H^1 norms are easilly obtained from

$$\|u - u_h\|_{L^2} = \sqrt{(u - u_h)^T M (u - u_h)}$$
$$\|u - u_h\|_{H^1} = \sqrt{(u - u_h)^T M (u - u_h) + (u - u_h)^T K (u - u_h)},$$

where u_h is the vector of the discrete finite element solution you obtained by solving the linear system $K u_h = b$ and u is the vector of the exact values of the solution. The exact solution is given by $u_0(x, y)$ evaluated at the grid points. Delive a single log log plot where you show the reduction of the error with respect to the mesh size h for both the triangular mesh and the quadrilateral mesh. You are free to use sparse matrices to speed up the solution of the related linear systems.

5. Visualization

Use ViSiT to visualize the solution for each different mesh resolution.