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 vector Poisson's equation in 3D

Due date: Thursday 29 October 2015, 10:30

We seek the discrete solution of the vector Poisson's equation

$$\begin{aligned} -\nabla^2 \mathbf{u}(x,y,z) &= \mathbf{f}(x,y,z), \quad \in \Omega \\ \mathbf{n}\cdot\nabla\mathbf{u} &= 0, \quad y=1, z=1, \\ \mathbf{u}(x,y,z) &= \mathbf{u}_0(x,y,z), \text{otherwise}, \end{aligned}$$

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 $\mathbf{f}(x,y)$ so that the exact solution of the PDE is

   $$u_{0,x} = u_{0,y} = u_{0,z} = x\, e^{-(y-1)^2(z-1)^2}.$$

   Note that **f** is a vector. Please explain what are the values of each comment.

2. Solve the problem by assuming an ordering of the variables per vector component, more precisely assume that the solution vector is

   $$\begin{pmatrix} U_x^T & U_y^T & U_z^T \end{pmatrix}^T \tag{1}$$

   where U_x contains all the unknowns corresponding to the x component, etc.

   - Solve the related linear system using the backslash operator and the CG algorithm with Cholesky preconditioning. Use the object-oriented MATLAB code that was given to you.
   - Write on a small table the timings using numbers with one decimal digit only.
   - Plot the solution using visit and give the norms of the errors for $N = 10, 20, 40$.

3. Next reorder the variables so that each node contains all the components. Your solution vector obtains the form:

$$\begin{pmatrix} u_{1_x} & u_{1_y} & u_{1_z} & \cdots & u_{n_x} & u_{n_y} & u_{n_z} \end{pmatrix}^T \tag{2}$$

For the permutation (reordering) of the original matrix and vector familiarize your self with the MATLAB function `repmat`[1]. Compute the permutation array and then write a function that accepts the matrix to be permuted `A`, the vector (rhs) to be permuted and the permutation array `p`, and returns the permuted matrix and vector.

- Use `spy()` to see the result of the permutation. And provide in your report pictures of both the original matrix and the permuted one.
- Solve the permuted system measuring the times for the backslash and CG with Cholesky preconditioning. Note that for using Cholesky, you should apply the boundary conditions maintaining the symmetry of the matrix. Complete the table you created previously with the running times of backslash and CG for the permuted system. How do they compate with those of the original unpermuted system?
- Permute back the solution to the original ordering and compare to the solution you obtained at the previous step. Are they the same? What are the error norms?

Hint: *Suppose that you have a 6x6 matrix A. The matrix A is identical with the permuted matrix A(p,p) where p is the permutation array* `p=[1 2 3 4 5 6]`. *So yes, in order to permute a matrix in MATLAB you only need to write* `B=A(p,p)`, *where* `p` *is the permutation array. How do you permute a vector? In order ta familiarize yourself with all of that you can use the symbolic package. The command* `A=sym('A', [6,6])` *will create a symbolic matrix 6 by 6. You can permute it in any way you like by creating a permutation array p. For example to permute the 3 first rows and columns to the 4th,5th and 6th correspondingly, you just need to introduce a permutation array* `p=[4 5 6 1 2 3]; B=A(p,p)`.

4. Finally assemble the matrix from the very beginning so that its ordering is identical to that of the previous step, meaning that
   - you need to assemble vector mass and stiffness element matrices,
   - you need to provide the spy plot of the local vector stiffness matrix and the global stiffness matrix,
   - you need to compare the solution with the solution of the previous step.
   - attach the code for the vector assembly, the code for generating the permutations of the matrix and the vectors and the inverse permutation array.

For this assignment you should use the object-oriented code that was provided to you. Think how the code should be structured introducing new methods in the class that exploit the existing methdos where needed and introduce entirely new methods for the vector assembly.

[1] Experiment with a 6 x 6 matrix to understand permuting a matrix and permuting it back. If the permutation routine works correctly then permuting the matrix back using the inverse permutation should result to the original matrix before the permutation was applied. Suppose p is the permutation vector. Then the inverse permutation $pinv$ satisfies $p(pinv(i)) = pinv(p(i)) = i$