Exercise # 3. Numerical Solution of the Poisson Problem.

Alexandre Rodrigues (2039952)

January 18, 2022

1 Method

I will describe relevant steps of developing this implementation and respective theory. Based on the homework text I implemented each step in Matlab.

1. **Input files from specified mesh**:

   1. The user selects a mesh refinement level (0 to 4);

   2. All 3 files are loaded: `topol`, `bound` and `coord`.

2. **create pattern for the stiffness matrix**:

   1. I created a range vector 1 to Ne, then place it 3 times as a column in a matrix and reshaped the matrix to obtain the column vector $row = [1, 1, 1, 2, 2, 2, 3, \ldots]^T$.

   2. The `col` vector is simply obtained by reshaping the `topol` as a column vector.

   3. Then we can compute the adjacency matrix as `A = sparse(row,col,1)` and the pattern for the stiffness matrix as `H = A' * A`, finally we clean the matrix as we only want the sparse pattern, `H = H * 0`.

3. **Stiffness matrix**: I created the function `[H, delta] = computeStiff(H, topol, coord)` to encapsulate all the following computations. It has the topology and coordinates matrices as inputs, as well as the H matrix with its pattern already defined before. Its outputs are the final H matrix and the `delta` vector with the surface measures of each element. Using a for loop for each element:

   1. Get coordinates of the 3 nodes that define the element, compute the surface measure for that element and save it in the `delta` vector.

$$\Delta = \frac{1}{2}\begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{vmatrix}$$

   2. Compute the b and c coefficients of the basis functions (a is not needed):

$$b_i = y_j - y_m \qquad c_i = x_m - x_j,$$

   others are obtained using anticlockwise indices permutation.

   3. Compute `Hloc` for an element as

$$H_{loc} = \frac{1}{4\Delta}\{b^T b + c^T c\}$$

   where $b = [b_i, b_j, b_m]$ and $c = [c_i, c_j, c_m]$

   4. Assemble the stiffness matrix H from the local matrices using algorithm 3.3.

4. **Right hand size** Using a for loop to iterate in the list of nodes:

   1. Get the node coordinates;

   2. Find elements that have this node as a vertex and get their surface measures from the `delta` vector;

   3. Compute the right hand size vector as $f_i \approxeq (-4 + 2x_i^2 + 2y_i^2)\frac{\sum_e \Delta_e}{3}$

5. **Boundary Conditions** As explained in the homework text we can simply change the diagonal value `H(i,i)`. So I changed the diagonal value of H at the node i of the boundary, `H(i,i)=Rmax`.

6. **Solve the Linear System** Following recommendations I used tolerance as 1×10^{-8} and Matlab's PCG method. Jacobi preconditioner as `M = sparse(diag(diag(H)))`. Choledsky preconditioner as `L = ichol(H)`. Then we can call the PCG method to solve the linear system. I also recorded the solving computational time for comparison. Finally we can show the convergence plots as the semi logarithmic plot of Residual Norm vs Iterations.

7. **Error computation** Using a for loop to visit each node:

   1. Get the coordinates of the node;

   2. Compute the analytical solution as $u(x, y) = x^2 + y^2 - x^2y^2 - 1$;

   3. Sum surface measures of each element that have this node as a vertex;

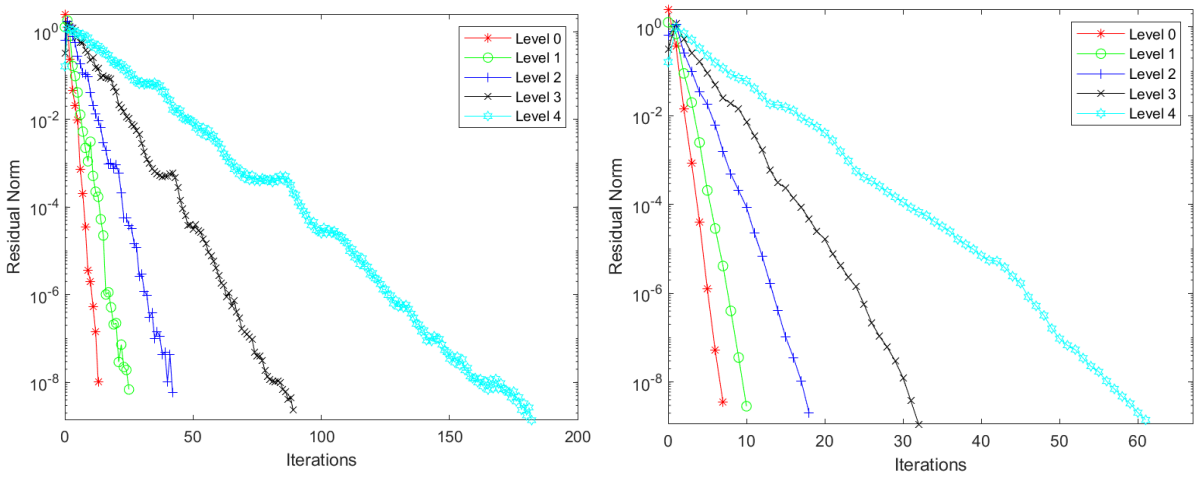
   4. Compute local error as $(u_i - u(x_i, y_i)^2 \frac{\sum_e \Delta_e}{3}$;

   5. Sum all local error and ϵ will be its square root.

2 Results

2.1 Convergence Plots

As required the following images show the convergence plots for all refinement levels, one for each preconditioner.



(a) Jacobi

(b) Choledsky

Figure 1: Convergence plots - Residual norms vs Iterations

There are clear differences in convergence when using the 2 preconditioners. The Choledsky preconditioner significantly improves convergence by reducing the number of iterations needed. There is also a clear slowdown of convergence when increasing the level of refinement.

2.2 Error

Level	**Error ϵ**	**Error Ratio**
0	6.912×10^{-2}	N/A
1	1.631×10^{-2}	0.236
2	3.984×10^{-3}	0.244
3	9.883×10^{-4}	0.248
4	2.465×10^{-4}	0.249

Table 1: FEM Convergence table

Each level of refinment drecreases l by a factor of 2, $l_{i+1} = \frac{1}{2} l_i$. The error is proportional to l^2 so $\epsilon_{i+1} = \frac{1}{4}\epsilon_i$. This corresponds to our experimental values and ratio.