Exercise # 3. Numerical Solution of the Poisson Problem.

Alexandre Rodrigues (2039952)

January 14, 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**: The user selects a mesh refinement level (0 to 4), all 3 files are loaded: `topol`, `bound` and `coord`.

2. **create pattern for the stiffness matrix**: 1st I created a range vector 1 to Ne, then place it 3 times in a matrix and convert the matrix to obtain the `row` vector $row = [1, 1, 1, 2, 2, 2, 3, \ldots]$. The `col` vector is simply obtained by reshaping the `topol` as a column vector. Then we can compute the adjancy matrix as `A = sparse(row,col,1)` and finaly the pattern for the stifness matrix as `H = A' * A`.

3. **Stifness matrix**: I created the function `[H, delta] = computeStiff(H, topol, coord)` to encapsulate all the following computations. It has the topology and coordinates matrixes as input 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: 3.1 - get coordinates of the 3 nodes that define the element, compute the surface measure for that element and save in the `delta` vector. 3.2 - compute b,c (a is not needed): ...s 3.3 - compute Hloc as ... (Algorithm 3.3)

4. **Right hand size** Using a for loop: 4.1 - get coords of the node; 4.2 - find ¡elements that have that node as a vertex to get their surface measures ... 4.3 - compute the rhs as $fxy * surfs/3$

5. **Boundary Conditions** As explained in the homework text we can simply change the diagonal value Hii. When susbitituting Hii=Rmax I got a nonpositive pivot H matrix resulting in a error when computing the Choledsky factorization. The solution was to multiply Hii by Rmax instead of replacing it. `H(i,i) = Rmax*H(i,i)`.

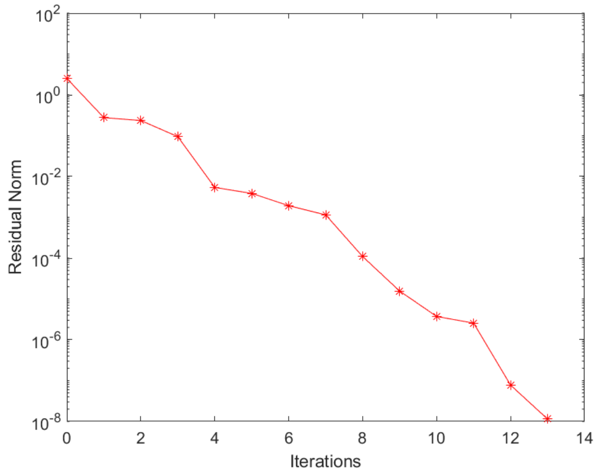
6. **Solve the Linear System** Using tolerance as 1e-8 and Matlab's PCG method.... Jacobi preconditioner as `M = sparse(diag(diag(H)))`. Choledsky preconditioner as `L = ichol(H)`. Then we can call the PCG method to solve the linear system. I also recorded the solving computational time for comparison.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

7. **Error computation** Using a for loop to visit each node: 7.1 - get the coordinates of the node; 7.2 - compute the analytical solution as $u_a = x^2 + y^2 - x^2y^2 - 1$. 7.3 - sum surface measures of each element that have this node as a vertex; 7.4 - compute local error as ... 7.5 - sum all local error and ϵ will be its square root.

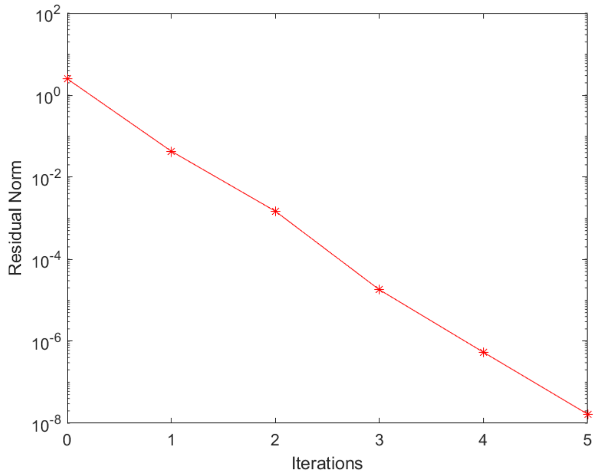
2 Results

2.1 Convergence Plots

...

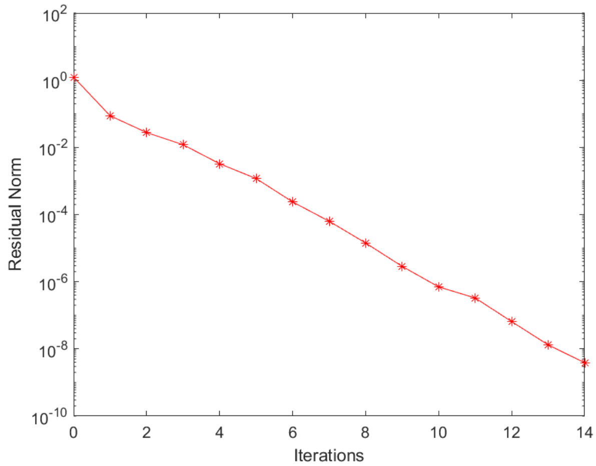


(a) Jacobi

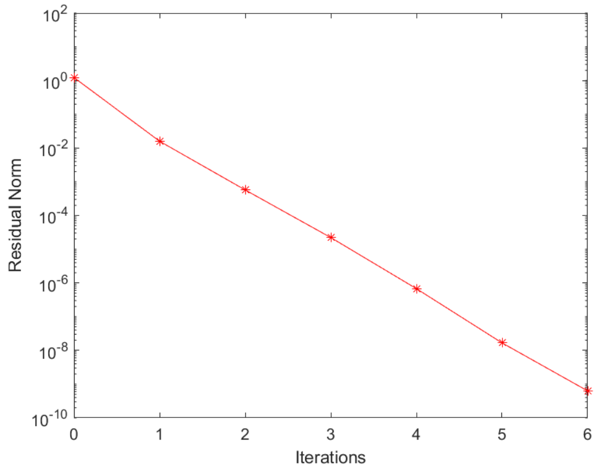


(b) Choledsky

Figure 1: Convergence plots for level 0

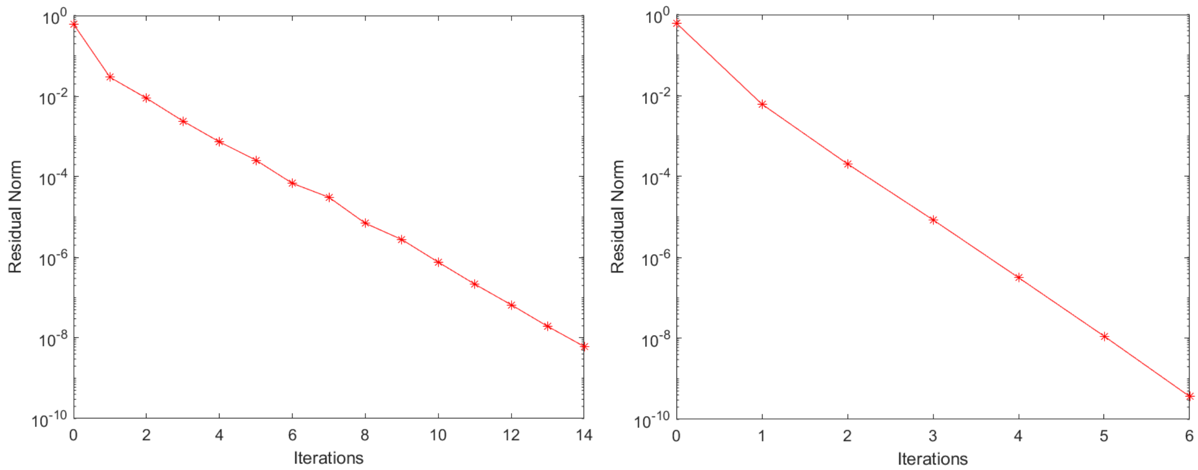


(a) Jacobi



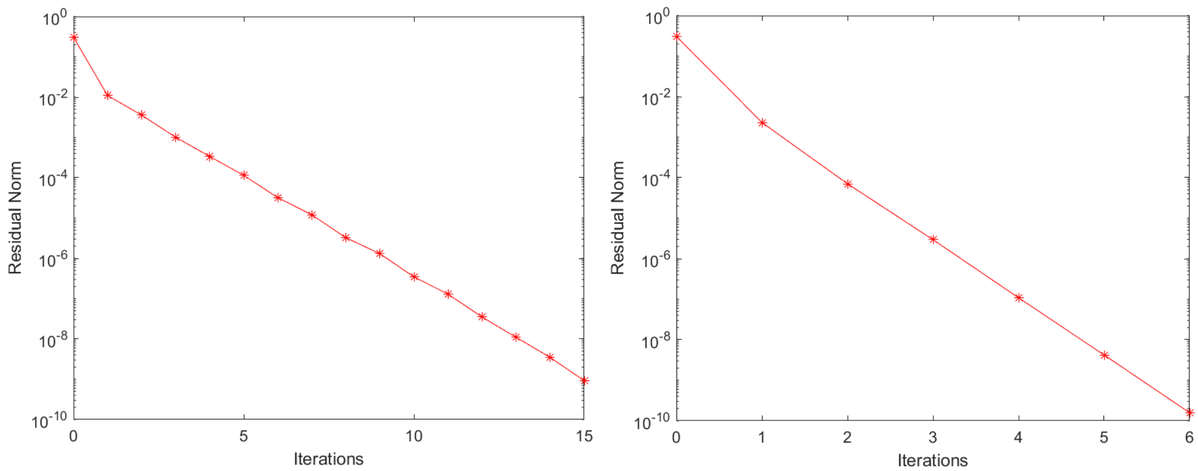
(b) Choledsky

Figure 2: Convergence plots for level 1



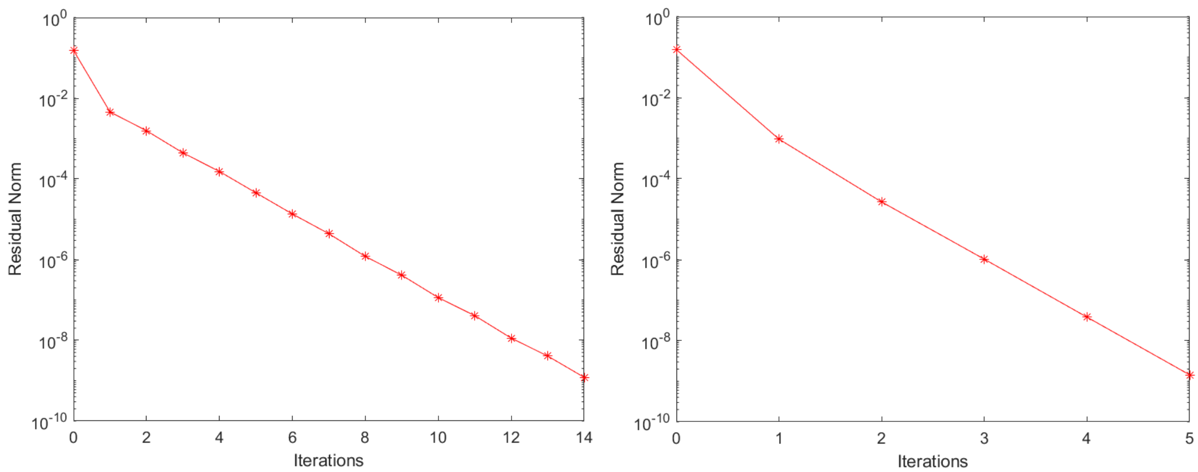
(a) Jacobi (b) Choledsky

Figure 3: Convergence plots for level 2



(a) Jacobi (b) Choledsky

Figure 4: Convergence plots for level 3



(a) Jacobi (b) Choledsky

Figure 5: Convergence plots for level 4

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 are no significant differences regarding the residual norms and between levels of refinement.

2.2 Epsilon

Level	ϵ	**Error Ratio**
0	0.9931	N/A
1	1.0508	1.0581
2	1.0629	1.0115
3	1.0657	1.0026
4	1.0664	1.0007

Table 1: FEM Convergence table

The ϵ value slightly increases on each step of refinement....??