ME 280A
Homework 3
Ahmad Zareei

Introduction

In the previous problem set we solved 1-D linear conservation problem as

$$\frac{d}{dx}\left(A_1\frac{du}{dx}\right) = f(x)$$

with finite element method with linear elements. We will use Preconditioner Conjugate Gradient solver to solve the linear system found from finite element method. Since solving the linear system has a high computational cost in solving a problem with FEM, we will see in this problem set, how to use conjugate gradient method to solve the linear system fast and accurate.

Conjugate gradient method is for the numerical solution of systems of linear equations where the matrix is symmetric and positive-definite which is the case in FEM method. The conjugate gradient method is iterative and applicable to sparse systems that are too large to be handled by a direct implementation, for instance guassian solver. We will impelement what is called Cholesky decomposition to solve faster. These Large sparse systems will also often arise when numerically solving partial differential equations or optimization problems.

Objectives

(I) Solving a 1-D steady state conservation equation with discontinuous material parameter A with driving force, using Finite element method with equally sized elements.

(II) Solving the linear system using Preconditioned Conjugate Gradient solver

(III) Understanding the error analysis and comparing the result with exact solution.

(IV) Understanding the effect of using number of elements on number of iterations.

(V) Investigating the effect of tolerance in Preconditioned Conjugate Gradient solver on number of iterations

(VI) Comparing the exact result from Guassian solver in MatLab with PCG solver

(VII) Comparing Potential function for different number of elements

Problem, Procedure and Results

1. Solve the following boundary value problem, with domain $\Omega = (0, L)$, analytically. You should use appropriate boundary conditions and interface conditions to derive your answer:

$$\frac{d}{dx}\left(A_1 \frac{du}{dx}\right) = 256 \sin\left(\frac{3}{4}\pi x\right) \cos\left(16\pi x\right) \tag{1}$$

where $L = 1$ and $u(0) = 0$ and $A_1 \mathrm{d}u/\mathrm{d}x|_{x=L} = 1$ and A_1 is given in ten statements as

$$\begin{aligned}
A_1 &= 2.00 \quad 0.0 \leq x < 0.1 \\
A_1 &= 2.50 \quad 0.1 \leq x < 0.2 \\
A_1 &= 1.25 \quad 0.2 \leq x < 0.3 \\
A_1 &= 0.25 \quad 0.3 \leq x < 0.4 \\
A_1 &= 4.00 \quad 0.4 \leq x < 0.5 \\
A_1 &= 1.75 \quad 0.5 \leq x < 0.6 \\
A_1 &= 0.50 \quad 0.6 \leq x < 0.7 \\
A_1 &= 0.75 \quad 0.7 \leq x < 0.8 \\
A_1 &= 3.25 \quad 0.8 \leq x < 0.9 \\
A_1 &= 1.00 \quad 0.9 \leq x < 1.0
\end{aligned}$$

The analytic solution need to be solved as a cievewise function.

Solution:

This equation is an inhomogenious differential equation, so it will have a particular and a general solution. The general solution is as

$$u_p(x) = C_i x + D_i \tag{2}$$

where C_i and D_i are some constants to be defined in each interval. To find the general solution, we first rewrite the right handside as

$$256 \sin\left(\frac{3}{4}\pi x\right) \cos\left(16\pi x\right) = 128 \sin\left(16.75\pi x\right) - 128 \sin\left(15.25\pi x\right) \tag{3}$$

with integrating the right handisde twice, we will have

$$u_g(x) = -\frac{128}{A_1(16.75\pi)^2} \sin(16.75\pi x) + \frac{128}{A_1(15.25\pi)^2} \sin(15.25\pi x) \tag{4}$$

The solution should be continuous and its first derivative should also be continuous at boundaries. These conditions could be written as

$$A_i \frac{\mathrm{d}u_i}{\mathrm{d}x}|_{x_i} = A_{i+1} \frac{\mathrm{d}u_{i+1}}{\mathrm{d}x}|_{x_{i+1}} \qquad u_i(x_i) = u_{i+1}(x_i) \tag{5}$$

where x_i is at points of $\{0.1, 0.2, ..., 0.9\}$. These boundary conditions will determine C_i and D_i. Starting from the boundary conditoion of $A_1 \mathrm{d}u/\mathrm{d}x$ at $x = 1$, we can find C_{10} and then solving backward, we can easily determine all of the C_is. Afterward, starting from the first point, using the boundary condition of $u(x = 0)$, we determine D_1 and then solving forward, we solve for D_i and therefore we can find the solution in the interval of $[-1, 1]$. The exact solution and its derivative $A_1 \mathrm{d}u/\mathrm{d}x$ are shown in figure 1. As it can be seen, the solution and its derivative are both continuous as we wanted them to be.

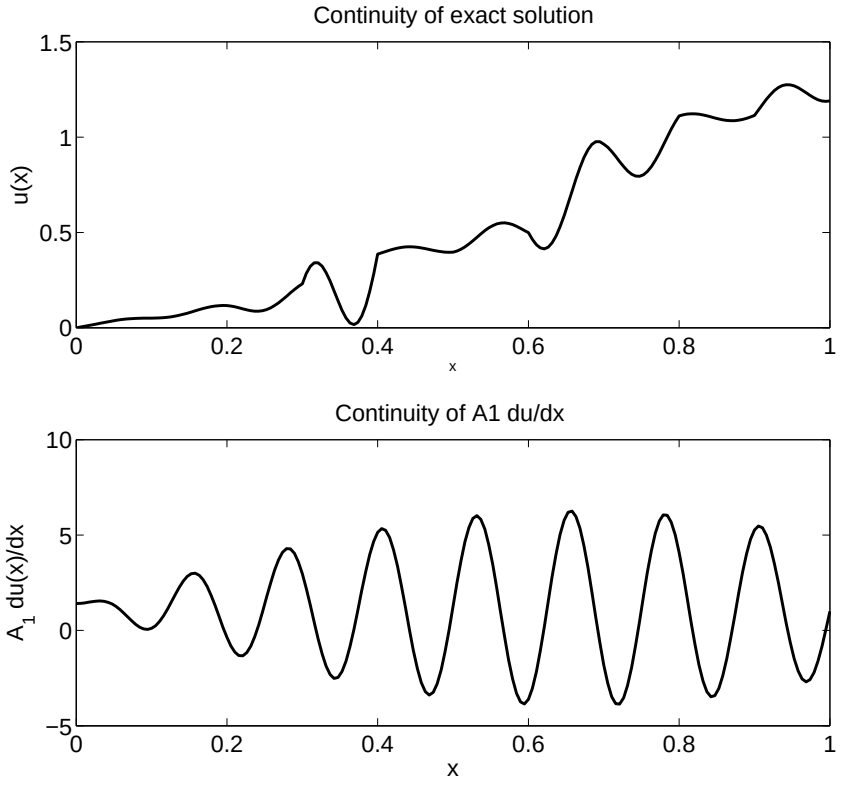

Figure 1: Exact solution of $u\,(x)$ and $\mathrm{d}u\,(x)\,/\mathrm{d}x$ vs x

2. Solve this with the finite element method using linear equal-sized elements. Use 100, 1000 and 10000 elements. You are to write a Preconditioned Conjugate-Gradient solver. Use the diagonal preconditioning given in the notes. The data storage is to be element by element (symmetric) and the matrix vector multiplication is to be done element by element during the iterations. Check your Conjugate Gradient generated results against a regular Gaussian solver, for example the one available in MATLAB.

solution

We did the first steps as previous homeworks, where we did devide the interval of $[0, 1]$ to N equally sized intervals with length of $1/N$. Then as before computed the stiffness $[K]$ and the loading vector $\{L\}$. The parts that has changed is the storage of the stiffness matrix, elements and multipication. In order to solve the problem using a Preconditioned Conjugate-Gradient solver, we first wrote functions to calculate multipication of a matrix with a vector. Since our matrix is sparse, we wrote our multiplyer ourselves. This function is called `my_multiply`. First, we stored the matrix as a $N \times 3$ matrix, where each row shows the contribution of each element to the total stiffness matrix. This matrix is as

$$[K]_{\text{table}} = \begin{bmatrix} K_{11}^{e=1} & K_{12}^{e=1} & K_{22}^{e=1} \\ K_{11}^{e=2} & K_{12}^{e=2} & K_{22}^{e=2} \\ \dots & & \\ K_{11}^{e=N} & K_{12}^{e=N} & K_{22}^{e=N} \end{bmatrix}. \tag{6}$$

For implementing the Dirichlet boundary condition, we need to remove the elements of $K_{11}^{e=1}$ and $K_{12}^{e=1}$ and replace them with 1 and 0 respectively. If in here, we make $K_{12}^{e=1}$, this will make the element of K_{21} in the main stiffness matrix also zero, since $K_{12} = K_{21}$. To make this correct, we need to change the loading vector for the second row (because K_{21} in the main stiffness matrix is now changed). We update the second row of the loading vector as $L_2 = L_2 - K_{12}^{e=1}\, u(x = 0)$. Since $u(0) = 0$, the second row does not change in here. In order to see how to use this matrix, we also need a table of nodes for this matrix. This matrix is called `element`, where it shows the nodes in the element. In our case that it is a simple $1D$ problem, this matrix is as

$$[e]_{\text{table}} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ \dots & \\ N & N+1 \end{bmatrix}. \tag{7}$$

This table, for instance shows how to use the stiffness matrix for the multipication.

For multipication of stiffness matrix into a vector, we wrote the function `my_multiply`. In this function, we go over all the elements, and find its contribution to the resulting vector. Note that, for example if we want to do the multipication of $y = Kx$, the

contribution of element e to result vector of y is as

$$\begin{bmatrix} y_1^e \\ y_2^2 \end{bmatrix} = \begin{bmatrix} K_{11}^{e=1}\ x_1^e & K_{12}^{e=1} x_2^e \\ K_{12}^{e=1}\ x_1^e & K_{22}^{e=1} x_2^e \end{bmatrix} \tag{8}$$

where (y_1^e, y_2^2) are enteries of the vector y regarding the nodes of the element e. The same is true for (x_1^e, x_2^2).

In order to implement the Preconditioned Conjugate gradient solver, we first implemented the following CG method algorithm as

- Step 1: For $i = 1$: Select $\{a\}^1 \rightarrow \{r\}^1 = \{R\} - [K]\{a\}^1 = \{z\}^1$
- step 2: Compute with $\{z\}^1 = \{r\}^1$

$$\lambda^1 = \frac{\{z\}^{T,1}\left(\{R\} - [K]\{a\}^1\right)}{\{z\}^{T,1}[K]\{z\}^1} \tag{9}$$

- Step 3: $\{a\}^2 = \{a\}^1 + \lambda^1\{z\}^1$
- step 4: For $i > 1$, compute $\{r\}^i = \{R\} - [K]\{a\}^i$

$$\theta^i = -\frac{\{r\}^{T,i}[K]\{z\}^{i-1}}{\{z\}^{T,i-1}[K]\{z\}^{i-1}} \tag{10}$$

then

$$\{z\}^i = \{r\}^i + \theta^i\{z\}^{i-1} \tag{11}$$

and

$$\lambda^1 = \frac{\{z\}^{T,i}\left(\{R\} - [K]\{a\}^i\right)}{\{z\}^{T,i}[K]\{z\}^i} = \frac{\{z\}^{T,i}\{r\}^i}{\{z\}^{T,i}[K]\{z\}^i} \tag{12}$$

- Step 4: Compute

$$e^i = \frac{||\{a\}^{i+1} - \{a\}^i||_K}{||\{a\}^i||_K} \tag{13}$$

- Step 5: If $e^i <$ tol then stop, otherwise repeat step 4.

We implemented the mentioned algorithm in the function called `my_conjugate_gradient`. We setted the tolerance of the solution to 10^{-12} for plotting the solutions. This function that we wrote, will get the stiffness matrix K, Loading vector L, initial guess x_0, tolerance of the solution *tol*, the routine name of the multipication used `my_multiply` and the elements connectivity matrix e as inputs and will give the solution x and number of iterations *iter* as it outputs. Results of solutions for different N with both `my_conjugate_gradient` routine and guassian solver of MATLAB are shown in figures 2, 3, 4. Note that, as the preconditioner, we first fount the Preconditioning matrix of T as square root of the diagonal terms, and stored it as a vectore $\{T\}$

$$\{T\} = \begin{Bmatrix} 1/\sqrt{K_{11}} \\ 1/\sqrt{K_{22}} \\ ... \\ 1/\sqrt{K_{NN}} \end{Bmatrix} \tag{14}$$

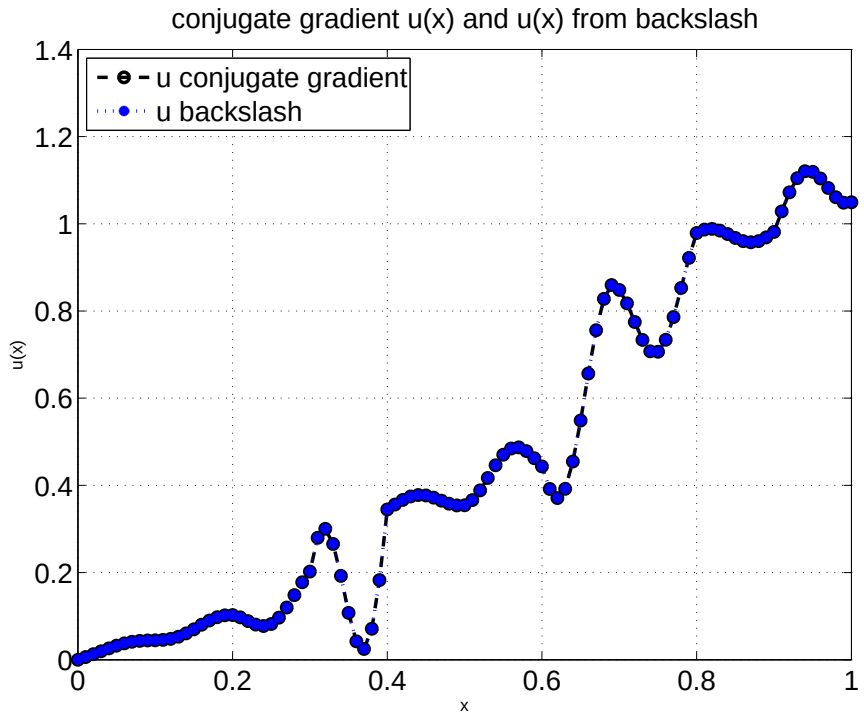

Figure 2: Solutions of preconditioned CG solver and MATLAB Gaussian solver for $N = 100$

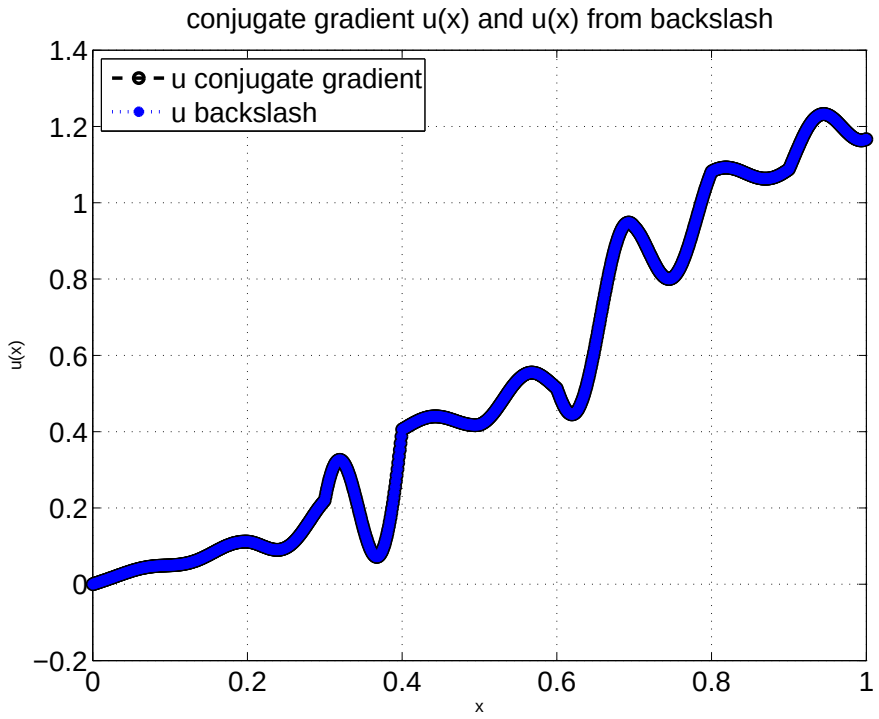

Figure 3: Solutions of preconditioned CG solver and MATLAB Gaussian solver for $N = 1000$

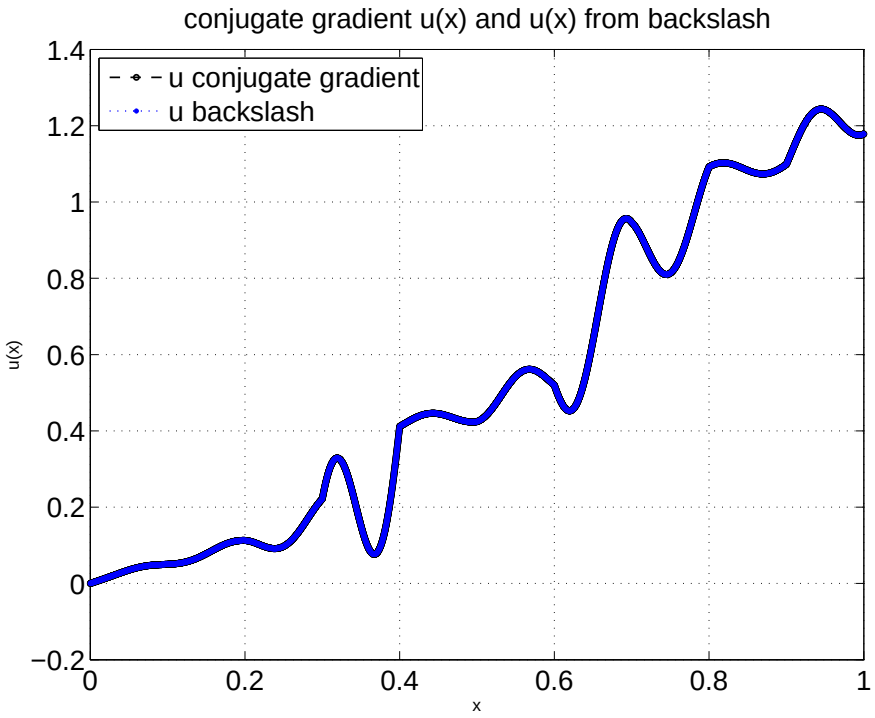

Figure 4: Solutions of preconditioned CG solver and MATLAB Gaussian solver for $N = 10000$

We actually solved for the following stiffness matrix and loading vector as

$$[\bar{K}] = T^T[K]T \qquad \{\bar{L}\} = T^T\{L\} \tag{15}$$

where

$$T^T[K]T \;\; T^{-1}\{a\} = T^T\{L\} \tag{16}$$

where this system of equations is exactly same as what we had before. We will find the $\{\bar{a}\}$ as

$$\{\bar{a}\} = T^{-1}\{a\} \tag{17}$$

where actual $\{a\}$ could be found as

$$\{a\} = T\{\bar{a}\} \tag{18}$$

3. You are to plot the solution (nodal values) for each N.

Solution:

The solutions are already shown in figures 2, 3 and 4.

4. You are to plot the error (defined below) for each N.

$$e^N = \frac{||u - u^N||_{A_1(\Omega)}}{||u||_{A_1(\Omega)}} \tag{19}$$

where

$$||u||_{A_1(\Omega)} = \sqrt{\int_\Omega \frac{du}{dx} A_1 \frac{du}{dx} dx}. \tag{20}$$

Solution:

Error defined in here is same as error that we had in previous homeworks. Plotting the logarithm of the eror versus logarithm the number of the elements is shown in the figure 5.

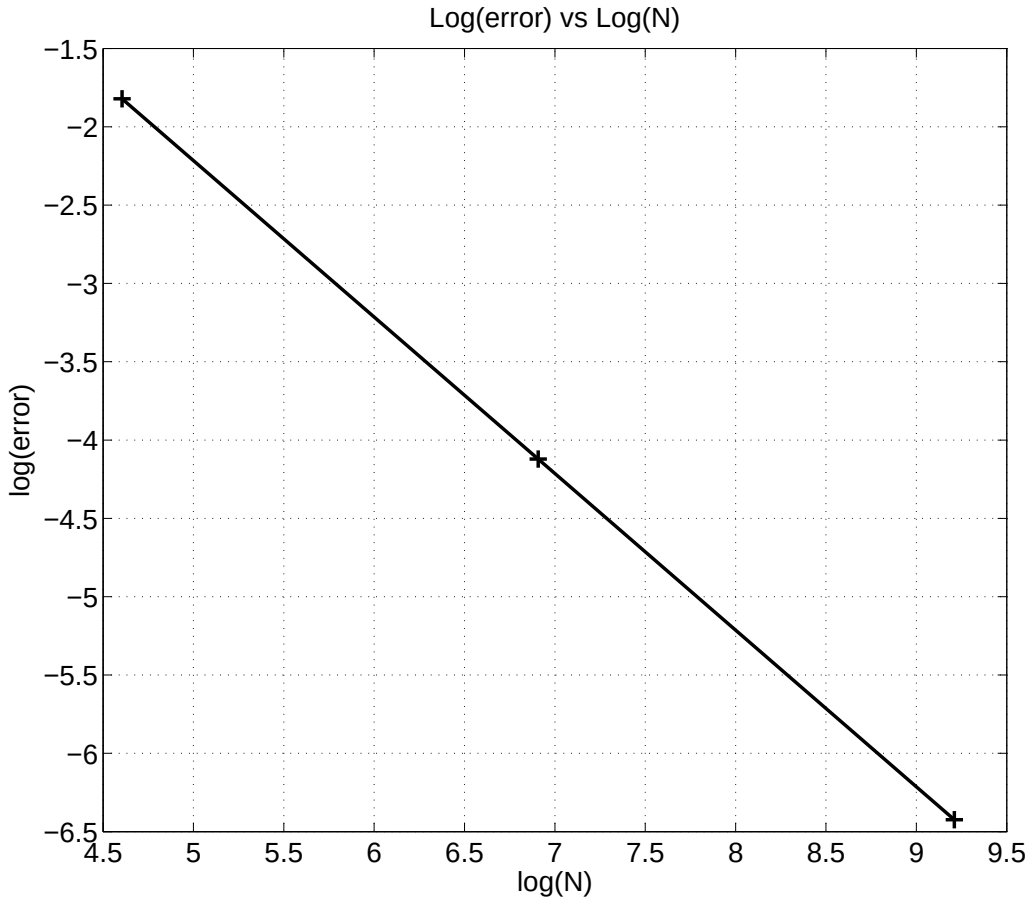

Figure 5: Logarithm of error vs logarithm of number of nodes

We also plotted the logarithm of the error versus the logarithm of element size, as shown in figure 6.

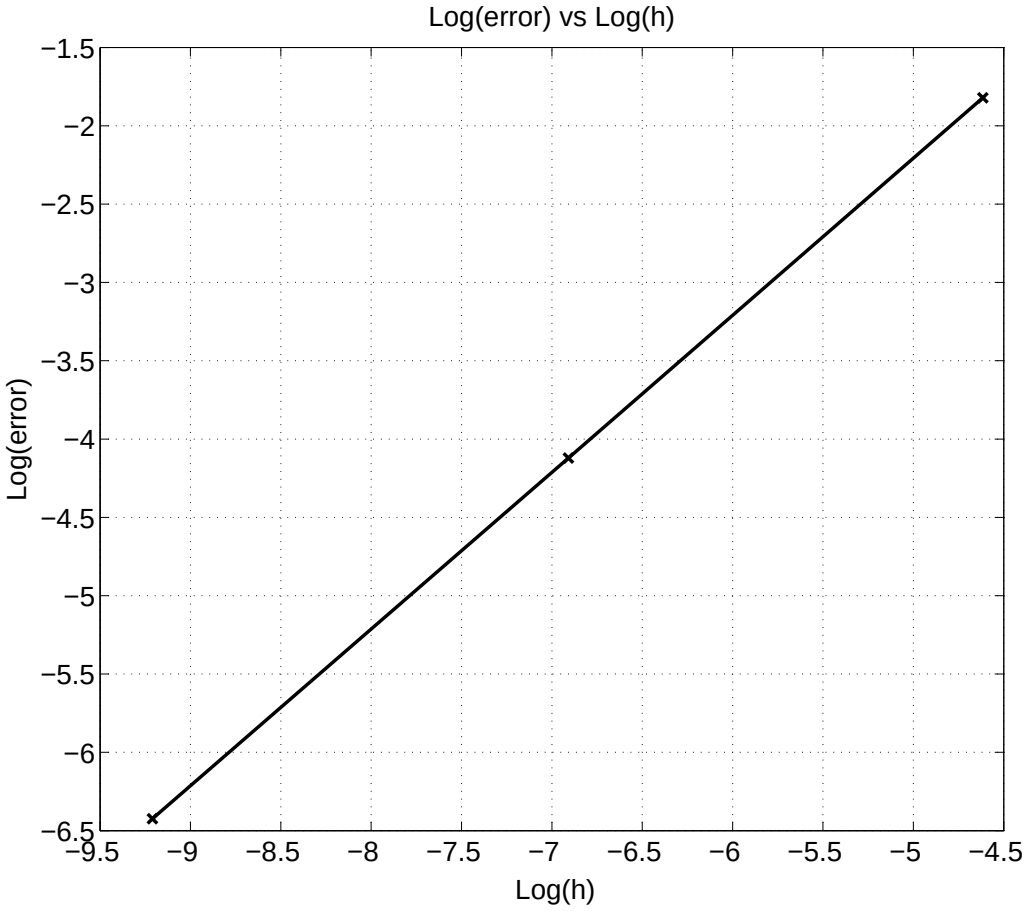

Figure 6: Logarithm of error vs logarithm of number of nodes

5. You are to plot Potential energy $\mathcal{J}(u^N)$ for each N.

   Solution:

   The Potential energy $\mathcal{J}$ is defined as

$$\mathcal{J}(u^N) = ||u||_{A_1(\Omega)} = \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x} A_1 \frac{\mathrm{d}u}{\mathrm{d}x} \mathrm{d}x \tag{21}$$

   We computed the energy norm, for different N, the result is shown in figure 7. To better show this graph, we plotted the potential of the energy versus the logarithm of the size of the elements in figure 8.

6. You are to plot the number of PCG-solver iterations for each N for a stopping tolerance of 0.001. As a further check, see what happens when you vary the tolerance for the solver.

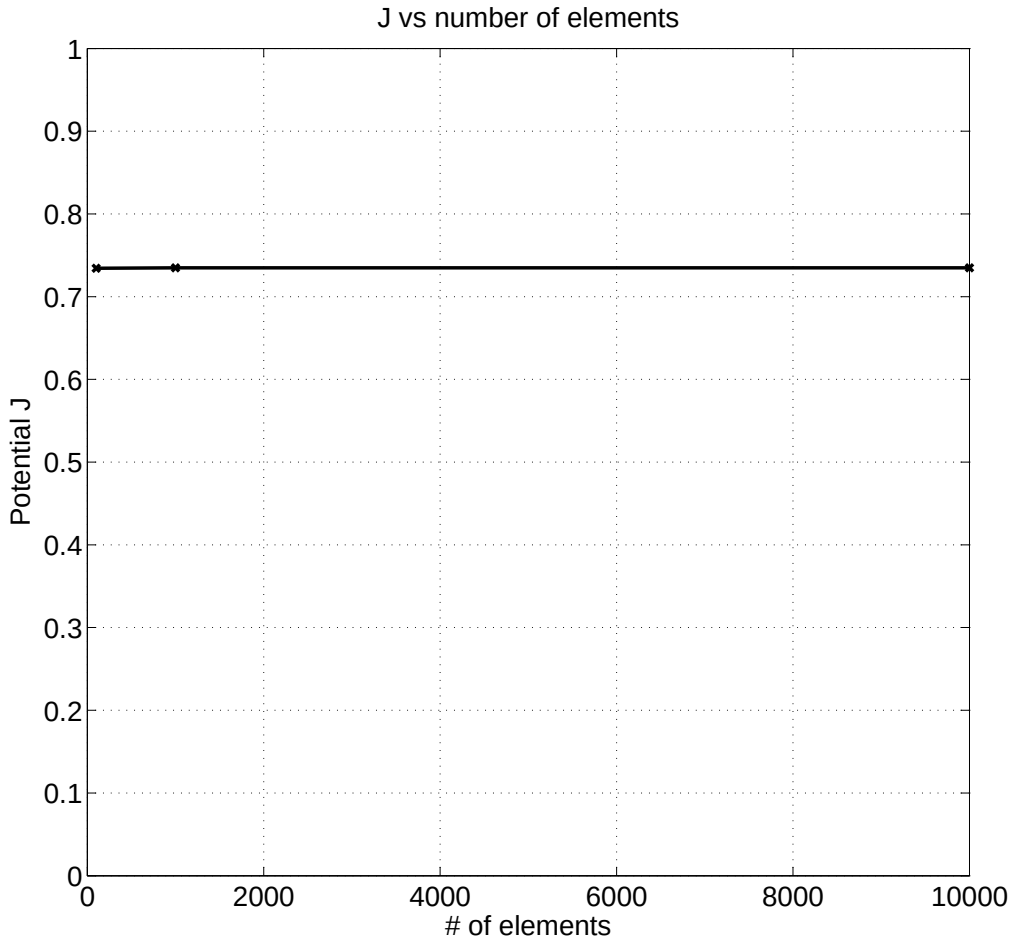

Figure 7: Potential Energy $\mathcal{J}$ versus Number of elements

Solution:

Number of the iterations versus number of the elements for the stopping tolerance of $\tau = 0.001$ is shown in figure 10. To better show the result, we plotted the result in a log-log figure. The result is as shown in figure 11. We also, did the simulation for the error of $\tau = 0.0001$ and $\tau = 0.00001$ and the results are shown in figures 12 and 13. Note that when we set the error to 10^{-12} the number of the iterations were $101, 1001, 100001$ for $N = 100, 1000, 10000$ respectivly.

Conclusion

In conclusion, we found that using preconditioned conjugate gradient solver is efficient for solving 1D finite element method, since the stiffness matrix is positive definite and symmetric. The order of iterations for the convergence is of $O(N)$ for a $N \times N$ matrix.

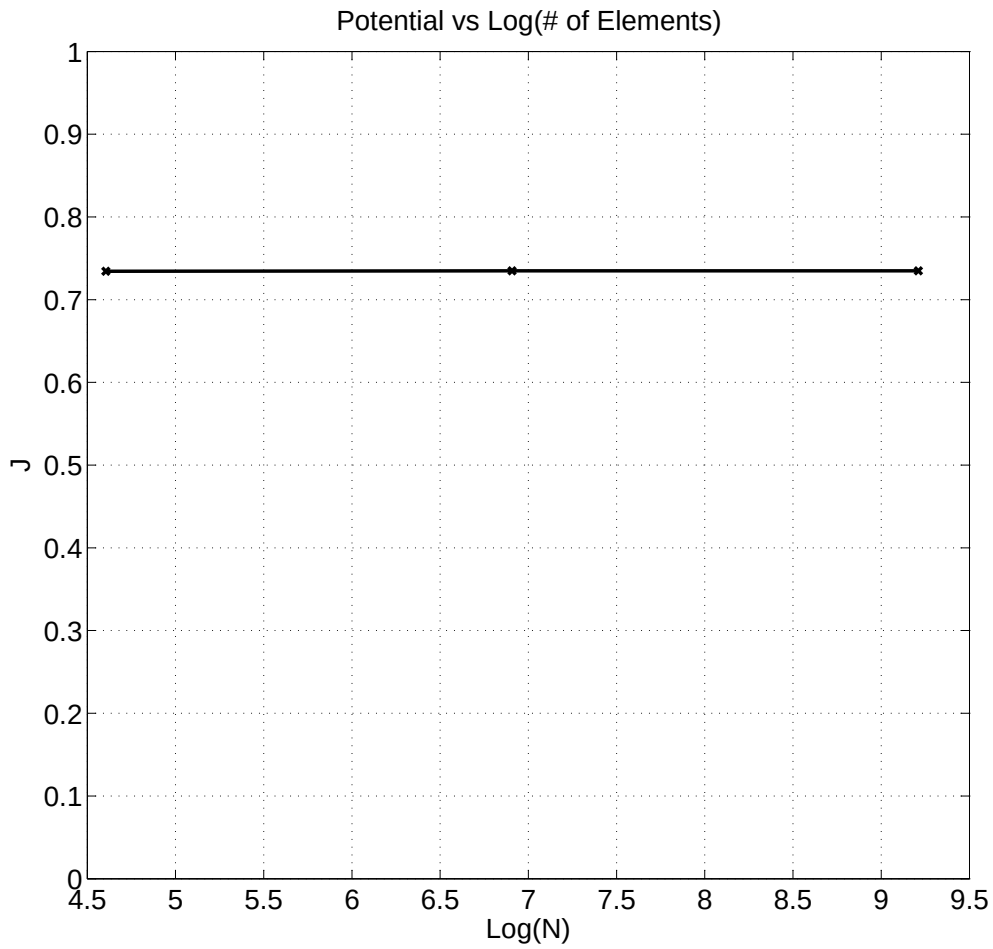

Figure 8: Potential Energy $\mathcal{J}$versus logarithm of Number of elements

We also found that the potential function is constant as we reach to the resutls. So this could be used as a measure of convergence, when we do not have the exact solution.

We found that the number of iterations needed will increase with precision that we chose and also it is of order of N where the stiffness matrix is of size $N \times N$. This is an efficient method, since just multipication of two dense matrices is of order N^3! In PCG, we use the facts that our matrix is sparse, symmetric and positive definite. That will decrease number of iterations to $O(N)$.

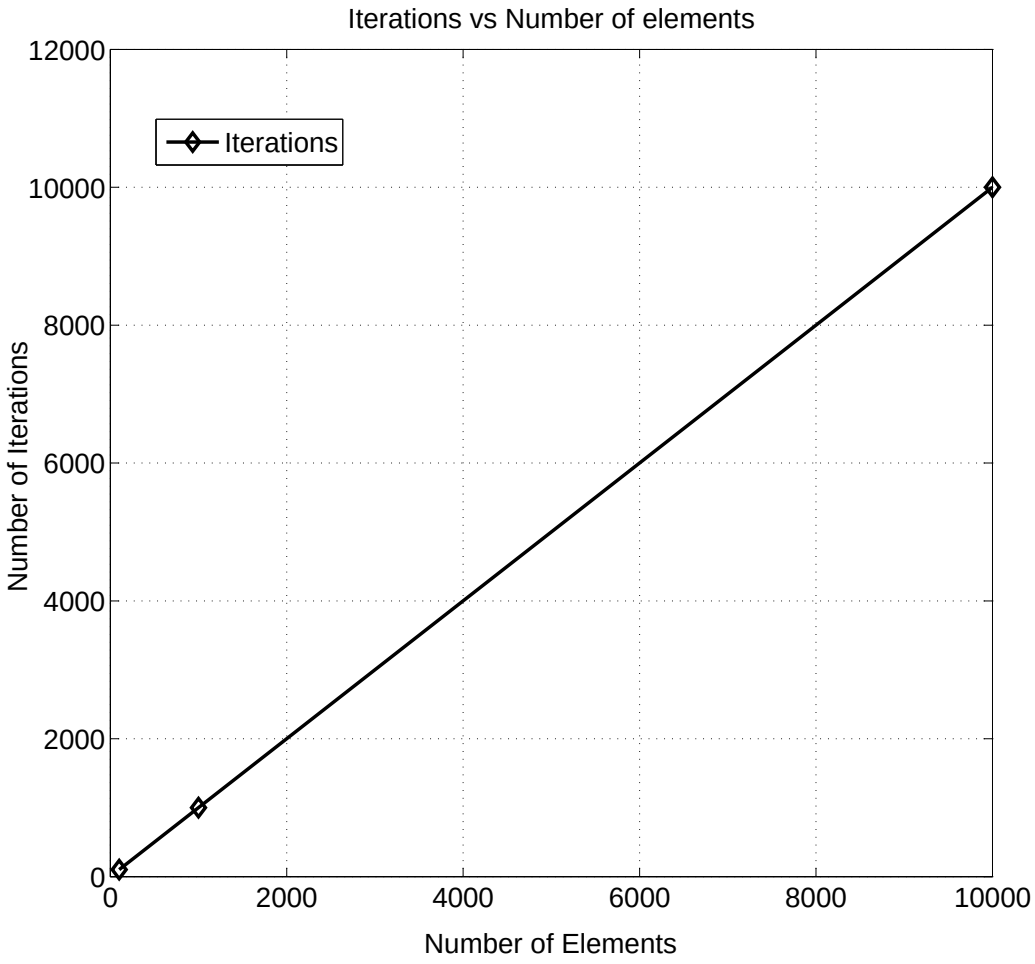

Figure 9: Number of the iterations versus number of the elements for $\tau = 10^{-12}$

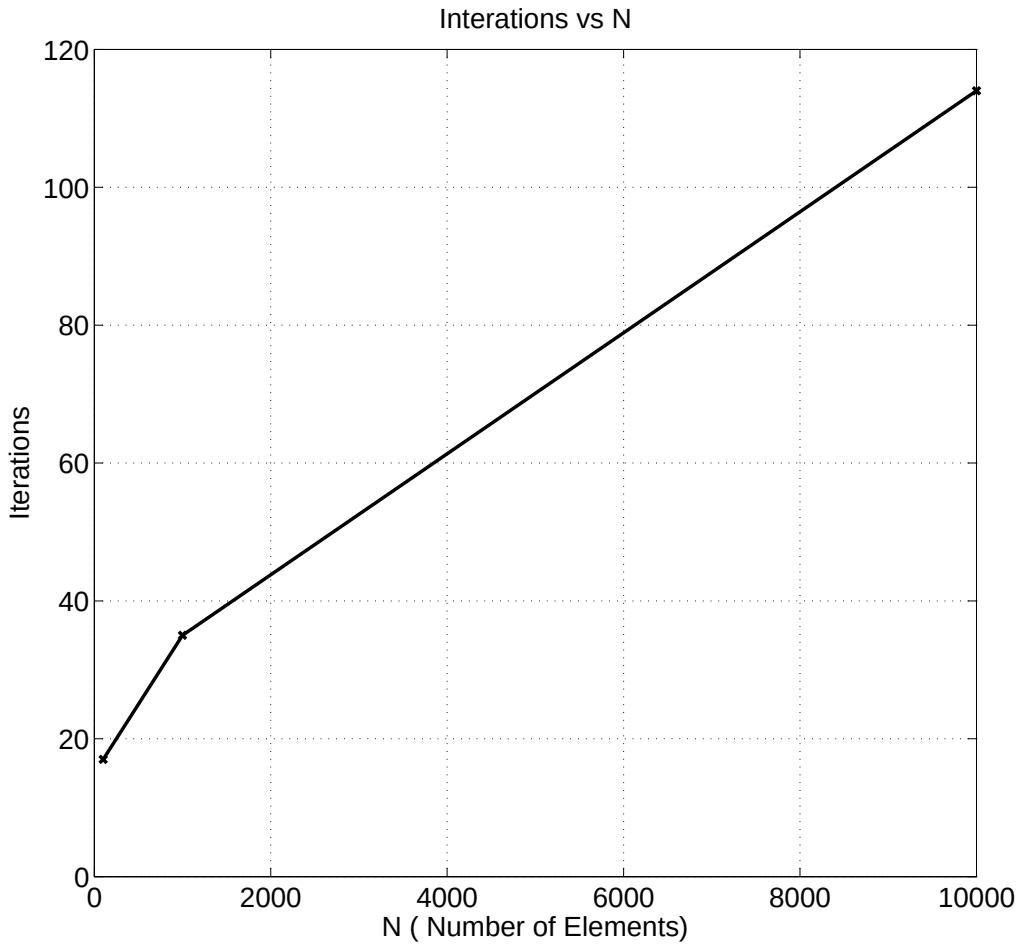

Figure 10: Number of the iterations versus number of the elements for $\tau = 0.001$

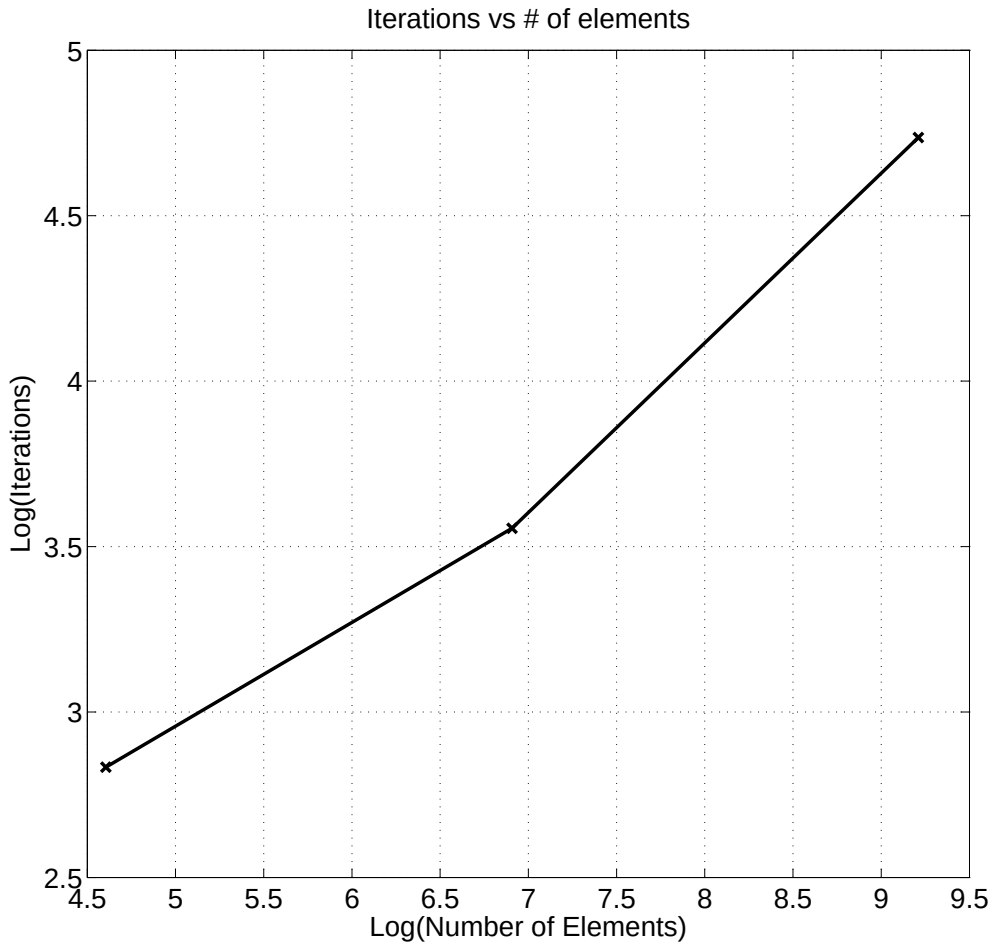

Figure 11: Logarithm of number of the iterations versus logarithm of number of the elements for $\tau = 0.001$

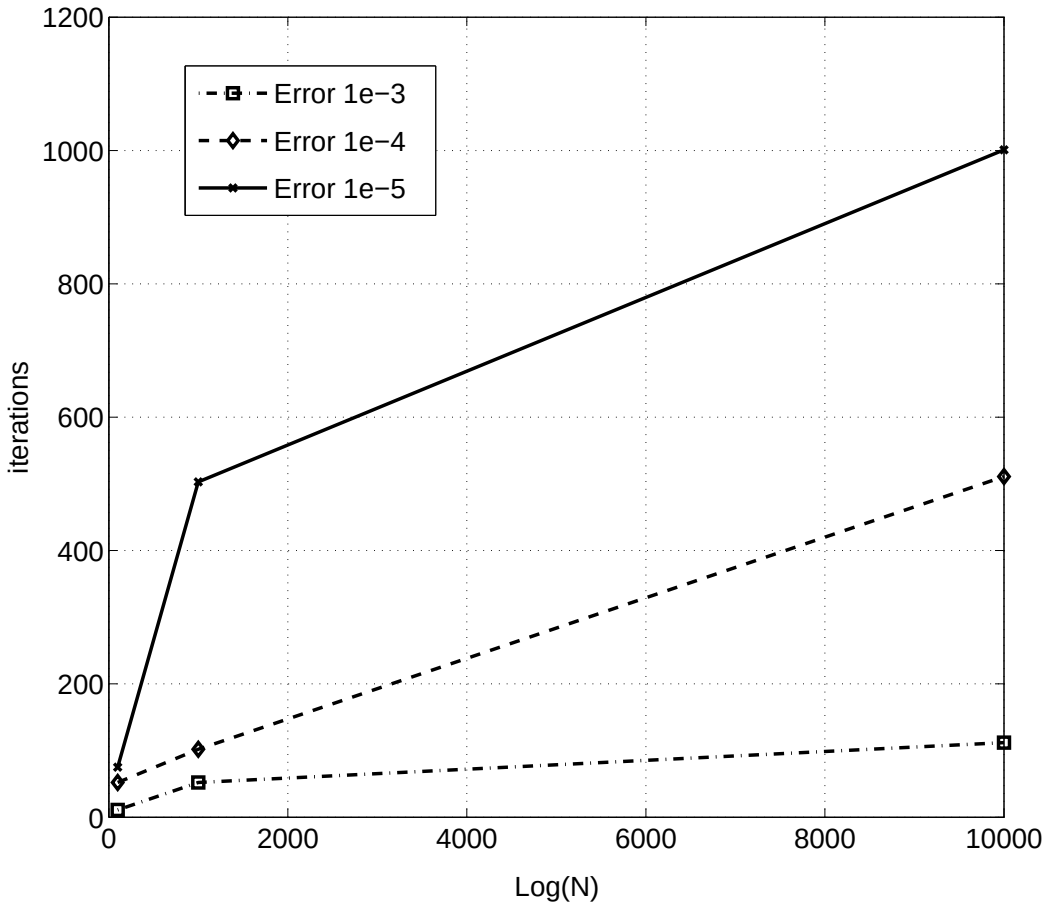

Figure 12: Number of iterations versus number of the elements for different $\tau = 10^{-3}, 10^{-4}, 10^{-5}$

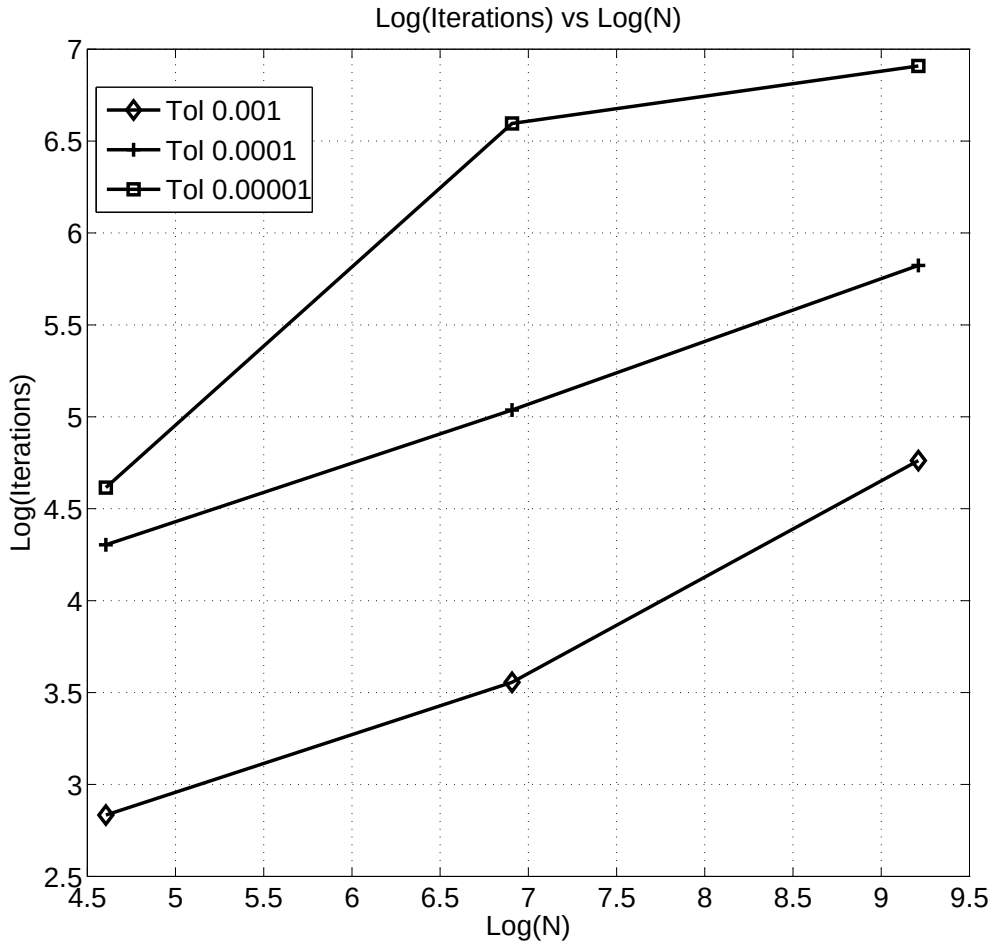

Figure 13: Logarithm of number of iterations versus logarithm of number of the elements for different $\tau = 10^{-3}, 10^{-4}, 10^{-5}$