Numerical Methods Laboratory 3
System of linear equations

dr eng. Grzegorz Fotyga
KIMiA, WETI, PG

1. Introduction

In this laboratory you will implement the two iterative methods (Jacobi and Gauss-Seidel) used to solve the systems of linear equations. Next, they will be tested on the data obtained by means of the PageRank implementation from Laboratory 2, developed by Google inc. Consider the system of equations in the following form:

$$\mathbf{M}\mathbf{r} = \mathbf{b}. \tag{1}$$

Vector $\mathbf{r}$, which is the solution of (1) contains the PR values of all pages from the analyzed network. The higher is the PR value, the higher is importance of the corresponding page.

Caution - make sure you have typed $diary('log\_yourID\_lab3')$, right after you had started your Matlab session.

Tasks to be performed:

- **Task A** – Use the function $[Edges] = generate\_network(N, density)$ to generate the random network connections. N is the number of the pages, $density$ is the density of the connections, the table $Edges$ includes all the network connections, where the pages from the first row link to the pages from the second row. Use the parameters values: $N = 10$, $density = 3$.
- **Task B** – Generate matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{I}$ and the vector: $\mathbf{b}$ for the network generated in **Task A** for $d = 0.85$. All matrices should be stored in *sparse* format. Save the obtained data using:
  save taskB_yourID A B I b
- **Task C** (10%) – Solve the system of equations (1) using the direct method ($\mathbf{r} = \mathbf{M}\backslash\mathbf{b}$). Save the resultant vector:
  save taskC_yourID r
- **Task D** (20%) – Measure the computational time of the direct solution of (1) for $d = 0.85$ and *density = 10* using the $tic \dots toc$ function for the 5 cases:
  $N = [500, 1000, 3000, 6000, 12000]$. Generate the plot 'computational time in a function of N' and save it:
  save taskD_yourID.png

2. Iterative algorithms

In this point your task is to implement the two iterative methods: Jacobi and Gauss – Seidel. The formulas for the subsequent iterations are defined as follows:

$$\mathbf{r}^{(k+1)} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{r}^{(k)} + \mathbf{D}^{-1}\mathbf{b}, \tag{2}$$

and

$$\mathbf{r}^{(k+1)} = -(\mathbf{D} + \mathbf{L})^{-1}(\mathbf{U}\mathbf{r}^{(k)}) + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}. \tag{3}$$

for Jacobi and Gauss – Seidel, respectively (the details are provided in the lecture slides).

Caution – The big mistake is to compute explicitly the $(\mathbf{D} + \mathbf{L})^{-1}$ term. A much better choice it to use the forward substitution (similarly as in the second step of LU factorization), which can be computed in Matlab using '\' operator.

In order to assess the error introduced by the approximation $\mathbf{r}^{(k)}$ (in k-th iteration), one can compute the residual vector:

$$\mathbf{res}^{(k)} = \mathbf{M}\mathbf{r}^{(k)} - \mathbf{b}. \tag{4}$$

The algorithm stops, when the 2-norm (Matlab: $norm(res)$) of the residual vector reaches 10^{-10}.

- **Task E** (35%) – implement Jacobi method. Use the functions *tril*, *triu*, *diag*. Make sure that the obtained matrices are correct. In each iteration compute the residual vector and stop the algorithm when it will reach 10^{-10}. Measure the computational time for: $N = [500, 1000, 3000, 6000, 12000]$.

  Generate the three plots: computational time in a function of N, number of iterations in function of N and the residuum norm for each of the iterations (for all values of N). In the last case, use $semilogy(norm\_res)$ function. Save the plots: *save taskE_yourID.png*

- **Task F** (35%) – generate the same results, as in task **E** for Gauss-Seidel method. Save the appropriate results and additionally write a short comment in *zadF_indeks.txt*: which method is more effective, which requires the lower number of iterations to converge?

Caution! Make sure you type 'diary off' before you quit Matlab! Remember to upload your session log, plots, codes and data files to e-nauczanie website.