Homework no. 4

The files (a_i.txt, b_i.txt, $i=1,\dots,5$) posted on the lab's web page, contain five linear systems with sparse triangular matrix, $\boldsymbol{Ax} = \boldsymbol{f}$, in the following way:

- $\boldsymbol{n}$ system's size, $\boldsymbol{p}, \boldsymbol{q}$ – indices where the secondary nonzero diagonals begin
- $\boldsymbol{n}$ values for the main diagonal of the matrix (vector $\boldsymbol{a}$), followed by $\boldsymbol{n-1}$ $\boldsymbol{(n-p+1)}$ values of vector $\boldsymbol{b}$ and then the $\boldsymbol{n-1}$ $\boldsymbol{(n-q-1)}$ elements of vector $\boldsymbol{c}$. The three vectors are separated by empty lines.
- $\boldsymbol{f_i}$, $\boldsymbol{i=1,2, \dots , n}$ the elements of vector $\boldsymbol{f} \in \mathbb{R}^{\boldsymbol{n}}$.

For this homework we consider traingular matrices of the following form:

$$A = \begin{pmatrix} a_1 & b_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ c_1 & a_2 & b_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & c_2 & a_3 & b_3 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & c_{n-2} & a_{n-1} & b_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & c_{n-1} & a_n \end{pmatrix}$$

1. Using the attached files, read the size of the linear system, the vector $\boldsymbol{f}$, the non-zero elements of matrix $\boldsymbol{A}$, generate the necessary data structures for economically storing the sparse matrix (use the sparse storing method described in Homework 3). Assume that all the elements on the main diagonal of the matrix are non-zero. Verify that these main diagonal elements of the matrix are non-zero.
   Consider the computation error $\boldsymbol{\varepsilon = 10^{-p}}$ as input.

2. Using the sparse storage for matrix $\boldsymbol{A}$, approximate the solution of the linear system:

$$\boldsymbol{Ax=f} \qquad (1)$$

employing the Gauss-Seidel method. Display (also) the number of iterations performed until convergence.

3. Verify the correctness of the computed solution by displaying the norm:

$$\left\| \boldsymbol{A}\boldsymbol{x}_{GS} - \boldsymbol{f} \right\|_{\infty}$$

where $\boldsymbol{x}_{GS}$ is the approximate solution obtained using Gauss-Seidel method.

4. In all computations that involve the elements of matrix $\boldsymbol{A}$, use the sparse storage structures (do not declare a classical matrix).

5. When implementing the Gauss-Seidel method use only one vector $\boldsymbol{x}_{GS}$.

Bonus 20 pt.: solve a linear sparse system of equations with the Gauss-Seidel for general tridiagonal matrices ($\boldsymbol{p}, \boldsymbol{q}$ are not restricted).

Iterative methods for solving sparse linear systems

Assume known that **det** $\boldsymbol{A} \neq \boldsymbol{0}$. Denote the exact solution of system (1) by $\boldsymbol{x}^*$:

$$\boldsymbol{x}^* := \boldsymbol{A}^{-1}\boldsymbol{f}.$$

The iterative methods for solving linear systems were developed for *'large'* systems ($\boldsymbol{n}$ *'large'*), with sparse matrix $\boldsymbol{A}$ (has *'few'* nonzero elements $\boldsymbol{a}_{ij} \neq 0$). The iterative methods do not change the matrix $\boldsymbol{A}$ (as it happens in Gaussian elimination or in $\boldsymbol{LU}$ decompositions or in $\boldsymbol{QR}$ factorizations), the non-zero elements of the matrix are employed for approximating the exact solution $\boldsymbol{x}^*$. For sparse matrices, special storing methods are employed (as the one described in Homework 3).

For approximating the solution $\boldsymbol{x}^*$, a sequence of real vectors $\left\{\boldsymbol{x}^{(k)}\right\} \subset \mathbb{R}^n$ is computed. If certain conditions are fulfilled, this sequence converges to the exact solution $\boldsymbol{x}^*$ of system (1):

$$\boldsymbol{x}^{(k)} \to \boldsymbol{x}^* \ , \ \text{for } \boldsymbol{k} \to \infty$$

The first vector of the sequence, $\boldsymbol{x}^{(0)}$, is, usually, set to 0:

$$x_i^{(0)} = 0 \ , \ \ i = 1,...,n \tag{2}$$

When the sequence converges, the limit is x^* the exact solution of system (1).

Gauss-Seidel Method

In order to be able to apply this method all the diagonal elements of matrix A must be non-zero:

$$a_{ii} \neq 0 \ , \ \ i=1,\dots,n$$

When reading the matrix from one of the attached files, verify that all the diagonal elements of the matrix are non-zero ($|a_{ii}| > \varepsilon, \ \forall i$). If there exists a zero, diagonal element, the system cannot be solved using successive over-relaxation iterative method.

The sequence of vectors generated by the Gauss-Seidel iterative method, is computed with the following relation:

$$x_i^{(k+1)} = \frac{\left(f_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)}{a_{ii}} \ , \quad i = 1, 2, ..., n. \tag{3}$$

The above formula must be adapted to the new way of storing the sparse matrix A. In the above sums for computing the ***i-th*** component of a vector, one needs only the non-zero elements a_{ij} from line i. For fast computing ***i-th*** component $x_i^{(k+1)}$ we need to rewrite formula (3) using only the vectors ***a, b,*** and ***c***.

For all iterative methods that solve linear systems, the convergence or divergence of sequence $\{x^{(k)}\}$ doesn't depend on the choice of the first vector $x^{(0)}$ in the sequence. There exist non-singular linear systems for which the sequence computed with Gauss-Seidel method is not convergent (does not compute a sequence that converge to the solution of the linear system).

In order to get an approximation for solution $\boldsymbol{x^*}$ one must compute an element $\boldsymbol{x^{(k)}}$ of the sequence of vectors, for $\boldsymbol{k}$ sufficiently large. If the difference between two consecutive elements of the sequence $\boldsymbol{\{x^{(k)}\}}$ becomes sufficiently *‚small'*, then the last computed vector is *‚near'* the exact solution:

$$\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \| \leq \boldsymbol{\varepsilon} \Rightarrow \| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{*} \| \leq \boldsymbol{c\varepsilon}\ ,\ \boldsymbol{c} \in \mathbb{R}_{+} \quad \rightarrow\ \boldsymbol{x}^{(k+1)} \approx \boldsymbol{x}^{*} \tag{2}$$

It is not necessary to store all the vectors of the sequence $\boldsymbol{\{x^{(k)}\}}$, we only need the last two computed elements, and for approximating the solution we need the vector that satisfies $\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \| \leq \boldsymbol{\varepsilon}$. In your program, you can use only two vectors:

$$\boldsymbol{x^c} \text{ for vector } \boldsymbol{x}^{(k+1)} \text{ and } \boldsymbol{x^p} \text{ for vector } \boldsymbol{x}^{(k)}.$$

The Gauss-Seidel method can be implemented by only using one vector for all the computations.

$$\boldsymbol{x_{GS} = x^c = x^p}.$$

When using one vector, computing formula (3) and the norm computation $\|\boldsymbol{x^{(k+1)}-x^{(k)}}\|=\|\boldsymbol{x^c-x^p}\|$ must be done in the same *for* loop.

Iterative method for solving a linear system

```
x^c = x^p=0;
k=0;
do
   {
      x^p = x^c;
      compute new x^c using x^p (with formula (3));
      compute Δx = || x^c - x^p||;
      k=k+1;
   }
while (Δx ≥ ε  and  k ≤ k_max  and  Δx ≤ 10^8) //(k_max = 10000)
 if (Δx < ε) x^c ≈ x^*; // x^c is the approximation of the exact solution
 else ‚divergence';
```

Example:

System matrix:

$$A = \begin{pmatrix} \mathbf{102.5} & \mathbf{2.5} & \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} \\ \mathbf{3.5} & \mathbf{104.88} & \mathbf{1.05} & \mathbf{0.0} & \mathbf{0.0} \\ \mathbf{0.0} & \mathbf{1.3} & \mathbf{100.0} & \mathbf{0.33} & \mathbf{0.0} \\ \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.73} & \mathbf{101.3} & \mathbf{0.0} \\ \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} & \mathbf{1.5} & \mathbf{102.23} \end{pmatrix}$$

Assume that:

$$x^{(0)} = \begin{pmatrix} \mathbf{1.0} \\ \mathbf{2.0} \\ \mathbf{3.0} \\ \mathbf{4.0} \\ \mathbf{5.0} \end{pmatrix} \quad , \qquad f = \begin{pmatrix} \mathbf{6.0} \\ \mathbf{7.0} \\ \mathbf{8.0} \\ \mathbf{9.0} \\ \mathbf{1.0} \end{pmatrix}$$

$x_1^{(1)}$ (classical storing)

$$= (f_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - a_{14}x_4^{(0)} - a_{15}x_5^{(0)}) / a_{11} =$$

$$= (6.0 - 0.0 * 2.0 - 2.5 * 3.0 - 0.0 * 4.0 - 0.0 * 5.0) / 102.5$$

(sparse storing uses only the non-zero elements of row 1)

$$= (6.0 - 2.5 * 2.0) / 102.5 = -0.01463414...$$

$x_2^{(1)}$ (classical storing)

$$= (f_2 - a_{21}x_1^{(1)} - a_{23}x_3^{(0)} - a_{24}x_4^{(0)} - a_{25}x_5^{(0)}) / a_{22} =$$

$$= (7.0 - 3.5 * (-0.01463414...) - 1.05 * 3.0 - 0.0 * 4.0 - 0.33 * 5.0) / 104.88$$

(sparse storing uses only the non-zero elements of row 2)

$$= (7.0 - 3.5 * 1.0 - 1.05 * (-0.01463414...)) / 104.88 = 0.033536$$

$x_3^{(1)}$ (classical storing)

$$= (f_3 - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} - a_{34}x_4^{(0)} - a_{35}x_5^{(0)}) / a_{33} =$$

$$= (8.0 - 0.0 * (-0.01463414...) - 1.3 * 0.033536 - 0.33 * 4.0 - 0.0 * 5.0) / 100.0$$

(sparse storing uses only the non-zero elements of row 3)

$$= (8.0 - 1.3 * 0.033536 - 0.33 * 4.0) / 100.00 = 0.066364$$

$x^{(k+1)}[i]$ (sparse storing uses only the non-zero elements of the i-th row)

$$= \frac{\left(f[i] - a[i][i-1] * x^{(k+1)}[i-1] - a[i][i+1] * x^{(k)}[i+1] \right)}{a[i][i]}$$

The linear systems that are stored in the files posted on the lab's web page have the following solutions:

- (a_1.txt, f_1.txt) has the solution $x_i = 1, \forall i = 0,...,n-1$,
- (a_2.txt, f_2.txt) has the solution $x_i = 1.0 / 3.0, \forall i = 0,...,n-1$
- (a_3.txt, f_3.txt) has the solution $x_i = 2.0 * (i+1) / 5.0$, $\forall i = 0,...,n-1$
- (a_4.txt, f_4.txt) has the solution $x_i = 2000 / (i+1), \forall i = 0,...,n-1$
- (a_5.txt, f_5.txt) has the solution $x_i = 2.0$, $\forall i = 0,...,n-1$. (?!?)