Homework no. 4

The files (a_i.txt, b_i.txt, $i=1,\ldots,5$) posted on the lab's web page, contain 5 linear systems with sparse matrix (with *'few'* non-zero elements, $a_{ij} \neq 0$, and size n *'big'*), $Ax = b$, in the following way:

For matrix A (files a_i.txt):

- n system's size,
- $a_{ij} \neq 0$, i, j – non-zero values from the sparse matrix $A \in \mathbb{R}^{n \times n}$, line index and column index of the corresponding element,

Pentru vectorul b (fişierele b_i.txt):

- n ,
- b_i , $i=1,2, \ldots , n$ all the elements of vector $b \in \mathbb{R}^n$.

1. Using the attached files, read the size of the linear system, the vector b, the non-zero elements of matrix A, generate the necessary data structures for economically storing the sparse matrix. For matrix A use the sparse storing method described below. Assume that the non-zero elements are randomly located in the file. Verify that the diagonal elements of the matrix are non-zero.

   Use two methods for memorizing the sparse matrix, one that is described in this text and the second, of your choice. For both these methods, solve the below-described requirements. Other methods for storing sparse matrices can be found here.

   Consider the computation error $\varepsilon = 10^{-p}$ as input.

2. Using the sparse storage for matrix A, approximate the solution of the linear system:

$$Ax=b \qquad (1)$$

   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:

$$\left\| Ax_{GS} - b \right\|_{\infty}$$

where x_{GS} is the approximate solution obtained using Gauss-Seidel method.

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

5. When implementing the Gauss-Seidel method use only one vector x_{GS}.

Bonus 25 pt.: compute the sum of two sparse matrices (a.txt, b.txt, aplusb.txt). Verify that the (computed) sum of matrices from files a.txt and b.txt is the matrix from file aplusb.txt. Two elements that have the same row and column indices (i, j) are considered equal if $|c_{ij}-d_{ij}| < \varepsilon$. Use one method (at your choice) for storing sparse matrices.

Remarks: 1) Don't use classically stored matrices for solving the above problem and don't use a function ***val(i,j)*** that computes the value of a_{ij} the corresponding element in the matrix, for all pairs (i,j).
2) Implementing the sparse storage method described in this file is **mandatory** (not implementing it will be penalised).

3) If in the homework's attached files, there are multiple entries with the same line and column indices:

val_1 , i, j
…
val_2 , i, j
…
val_k , i, j

this situation has the following meaning:

$$a_{ij} = val_1 + val_2 + \cdots + val_k .$$

Sparse matrix storing method

A sparse vector is a vector that has ‘few’ non-zero elements. Such a vector can be stored economically in a data structure that will keep only the non-zero values and the position in the vector of the respective value:

$$\{(val \neq 0, i);\ x_i = val\}.$$

A sparse matrix can be economically stored as a vector of sparsely stored vectors: each line of the matrix is stored in a sparse vector structure.

Iterative methods for solving sparse linear systems

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

$$x^* := A^{-1}b.$$

The iterative methods for solving linear systems were developed for *‘large’* systems (***n*** *‘large’*), with sparse matrix ***A*** (has *‘few’*

nonzero elements $a_{ij} \neq 0$). The iterative methods do not change the matrix A (as it happens in Gaussian elimination or in LU decompositions or in QR factorizations), the non-zero elements of the matrix are employed for approximating the exact solution x^*. For sparse matrices, special storing methods are employed (as the one described in Homework 3).

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

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

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

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

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:

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

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:

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

$$= \frac{\left(b_i - \sum_{\substack{\text{non-zero values on i-th row} \\ \text{except the diagonal element}}} val * x^{(?)}_{column\_index} \right)}{\textbf{i-th row diagonal element value}}, \quad i = 1,2,...,n \tag{3}$$

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 have easy access to the non-zero elements of matrix ***A*** row ***i***, that's why in the sparse storing methods the elements are memorized accordingly.

If the matrix ***A*** is row diagonally dominant, that is:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad \text{for all } i = 1,...,n$$

the sequence $\{x^{(k)}\}$ computed with the Gauss-Seidel algorithm, converges to solution x^*. 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.

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

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

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

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

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

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

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

Iterative method for solving a linear system (with 2 vectors!)

$x^c = x^p = 0$;

$k = 0$;

do

{

$x^p = x^c$;

compute new x^c using x^p (with formula (3));

compute $\Delta x = \| x^c - x^p \|$;

$k = k + 1$;

}

while ($\Delta x \geq \varepsilon$ **and** $k \leq k_{max}$ **and** $\Delta x \leq 10^8$) //($k_{max} = 10000$)

if ($\Delta x < \varepsilon$) $x^c \approx x^*$; // x^c is the approximation of the exact solution

else ,*divergence'*;

Example:

The matrix:

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

can be sparsely/economically stored in the following way:

{ { (2.5, 2) , (102.5, 0)},
{ (3.5, 0), (0.33, 4),(1.05, 2), (104.88, 1)},
{ (100.0, 2)},
{ (1.3, 1) , (101.3, 3)},
{ (1.5, 3), (0.73, 0), (102.23, 4)} }.

Assume that:

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

$x_0^{(1)}$ **(**clasical storage)

$= (b_0 - a_{01}x_1^{(0)} - a_{02}x_2^{(0)} - a_{03}x_3^{(0)} - a_{04}x_4^{(0)}) / a_{00} =$

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

(sparse storage uses only the non-zero elements from the first row)

$= (6.0 - 2.5 * 3.0) / 102.5 = -0.01463414...$

$x_1^{(1)}$ (clasical storage)

$= (b_1 - a_{10}x_0^{(1)} - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - a_{14}x_4^{(0)}) / a_{11} =$

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

(sparse storage uses only the non-zero elements from the second row)

$= (7.0 - 1.05 * 3.0 - 3.5 * (-0.01463414...) - 0.33 * 5.0) / 104.88 = 0.0214647...$

$x_2^{(1)}$ (clasical storage)

$= (b_2 - a_{20}x_0^{(1)} - a_{21}x_1^{(1)} - a_{23}x_3^{(0)} - a_{24}x_4^{(0)}) / a_{22} =$

$= (8.0 - 0.0 * (-0.01463414...) - 0.0 * (0.0214647...) - 0.0 * 4.0 - 0.0 * 5.0) / 100.0$

(sparse storage uses only the non-zero elements from the third row)

$= (8.0) / 100.00 = 0.08$

$x_3^{(1)}$ (clasical storage)

$= (b_3 - a_{30}x_0^{(1)} - a_{31}x_1^{(1)} - a_{32}x_2^{(1)} - a_{34}x_4^{(0)}) / a_{33} =$

$= (9.0 - 0.0 * (-0.01463414...) - 1.3 * (0.0214647...) - 0.0 * 0.08 - 0.0 * 5.0) / 102.23$

(sparse storage uses only the non-zero elements from the fourth row)

$= (9.0 - 1.3 * (0.0214647...)) / 101.30 = 0.0885696...$

$x_4^{(1)}$ (clasical storage)

$= (b_4 - a_{40}x_0^{(1)} - a_{41}x_1^{(1)} - a_{42}x_2^{(1)} - a_{43}x_4^{(1)}) / a_{44} =$

$= (1.0 - 0.73 * (-0.01463414...) - 0.0 * (0.0214647...) - 0.0 * 0.08 - 1.5 * (0.0885696...)) / 102.23$

(sparse storage uses only the non-zero elements from the fifth row)

$= (1.0 - 1.5 * (0.0885696...) - 0.73 * (-0.01463414...)) / 101.30 = 0.0085868...$

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

$$= \frac{\left(b[i] - \sum(\text{val} \neq 0 \text{ from row } i) * x^{(?)}[\text{column index coresponding to non-zero value}]\right)}{\text{the valued of the diagonal element from row } i}$$

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

- (a_1.txt, b_1.txt) has the solution $x_i = 1, \forall i = 0,...,n-1$,
- (a_2.txt, b_2.txt) has the solution $x_i = \frac{2.0}{3.0}, \forall i = 0,...,n-1$
- (a_3.txt, b_3.txt) has the solution $x_i = 0.3 * i \ , \forall i = 0,...,n-1$
- (a_4.txt, b_4.txt) has the solution $x_i = \frac{i}{9.0}, \forall i = 0,...,n-1$
- (a_5.txt, b_5.txt) has the solution $x_i = 3.0 \ , \forall i = 0,...,n-1$. (?!?)