The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program

of the Lobachevsky State University of Nizhni Novgorod

among the world's research and education centers

Strategic initiative

“Achieving leading positions in the field of supercomputer technology

and high-performance computing”

ITERATIVE METHODS FOR SOLVING LINEAR SYSTEMS

Practice 3. Preconditioning using incomplete LU-factorization

Nizhni Novgorod

2014

OBJECTIVES

The purpose of this laboratory work is implementation of preconditioner construction methods based on incomplete *LU*-factorization.

ABSTRACT

The subject of this laboratory work is preconditioning methods based on incomplete *LU*-factorization. We shall introduce the notion of incomplete *LU*-factorization (or *ILU*-factorization) and compare it to complete *LU*-factorization. To improve the method effectiveness, we shall introduce the idea of filling level that enables construction of the *ILU*(*p*) algorithm to control the factorization filling degree. In the course of work, students will have to implement this method of preconditioner construction for nondegenerate general sparse matrices. The objectives of this laboratory work will also include analysis of dependence of the matrix condition number (after preconditioning) on the selected filling level.

GUIDELINES

One of the key properties of a matrix A in a linear system is its *condition number* μ_A. For example, the condition number of a symmetric positive definite matrix is a relation of its maximum eigenvalue to the minimum one. This relation is more complex for general matrices.

The condition number is critical for convergence of iterative methods to solve linear systems. If $\mu_A >> 1$, the convergence rate will be low. If $\mu_A \approx 1$, the convergence rate will be high. Therefore, to improve matrix condition, the use of special methods, i. e. matrix *preconditioning* will be required.

The main idea of preconditioning is transformation of a linear system with the ill-conditioned matrix A to another linear system which is well-conditioned and results from multiplication of the matrix A by a matrix M^{-1} *(*where M is *preconditioner)*. A preconditioner can be constructed in a number of ways; in this work we will use incomplete factorization of the matrix A.

As a rule, solving linear systems using direct methods involves factorization, i.e. factorization of the initial matrix into factors L and U, where L is the lower triangular matrix and U is the upper triangular one. In this case, one may speak of a complete factorization or complete *LU*-factorization of the matrix A. In fact, complete factorization means solving the linear system using a direct method. However, this operation is a) complex b) difficult to implement for sparse matrices. Incomplete *LU*-factorization (ILU – *Incomplete LU*) lets obtain matrices L and U close to those obtained as a result of complete factorization. In such a case, incomplete *LU*-factorization takes much less time.

The basis for the implementation procedure of the incomplete LU-factorization is the Gaussian elimination method. In this case, for the purposes of sparse matrix LU-factorization the algorithm is divided into two parts, symbolic and numeric. The symbolic part involves construction of L and U patterns. The numeric part includes computation of concrete values for all non-zero elements corresponding to the patterns.

The idea of $ILU(0)$-factorization is to eliminate all new non-zero elements that appear in the course of factirization, from the factor. For the purpose of this algorithm, the initial matrix A pattern is used as the patterns of L' and U'. The lower A triangle serves as the pattern of L' while the upper one is used as the pattern of U'. In the numeric part, coefficients of matrices L' and U' are computed in such a way to make the matrix $L'U'$ coincide with A for all non-zero A elements. It should be noted that due to incomplete factorization multiplying L' by U' may lead to additional non-zero elements.

The first part of this work has three stages. The first stage consists in the $ILU(0)$ algorithm implementation. Then the algorithm is checked for correctness. In the end, the main function of the program is developed to enable experimentation and evaluate the obtained preconditioners. Experimental results show a high preconditioner construction algorithm running speed and a certain improvement of matrix conditioning after the constructed preconditioner is used.

The second part of the laboratory work involves implementation of another, more effective (than ILU(0)) preconditioning algorithm. The idea of algorithm improvement is that the patterns of matrices L' and U' obtained as a result of incomplete LU-factorization are closer to the patterns of L and U (i. e., a certain degree of factor filling compared to *ILU(0)* is allowed). Let us study one of these algorithms, the *ILU(p)* factorization method based on the idea of *filling level* p. In this case, p regulates accuracy of the factor pattern construction and may be interpreted as the number of additional diagonals where filling is possible. So if p is equal to the matrix size, the algorithm will result in the complete LU-factorization. If p is equal to zero, the algorithm transforms to $ILU(0)$.

This laboratory work reviews both the initial $ILU(p)$ algorithm whose implementation is rather slow, and its modification enabling an effective computational process. A software implementation of the modified algorithm is demonstrated. *ILU(0)*-preconditioning is compared to *ILU(p)*-preconditioning, condition number reduction resulting from p increase is shown.

RECOMMENDATIONS FOR STUDENTS

Basic preconditioner construction algorithms are given in [1]. These issues can be studied in more detail in [2].

REFERENCES

1. J. Dongarra et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.
2. Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

PRACTICE

1. Implement a parallel *ILU*(*p*) algorithm version and analyze its scalability.
2. Implement a block modification of the *ILU*(*p*) algorithm to increase the algorithm effectiveness for large matrices.

TEST

1. What is the complexity of the Gaussian elimination method as applied to a triangular matrix?
    a. O(n)
    b. +O(n^2)
    c. O(n^3)
2. What pattern does the preconditioner matrix resulting from the *ILU*(0) algorithm have?
    a. +The same as the initial matrix
    b. Diagonal matrix pattern
    c. Dense matrix pattern
3. How many non-zero elements are there in the matrix resulting from *ILU*(0) algorithm?
    a. Fewer than in the initial matrix
    b. +The same as in the initial matrix
    c. More than in the initial matrix
4. If you compare the number of non-zero elements in preconditioner matrices resulting from *ILU*(*i*) and *ILU*(*i*+1), respectively, you will see that
    a. *ILU*(*i*) > *ILU*(*i*+1)
    b. *ILU*(*i*) < *ILU*(*i*+1)
    c. + *ILU*(*i*) <= *ILU*(*i*+1)
    d. *ILU*(*i*) >= *ILU*(*i*+1)
5. What *p* value in the *ILU*(*p*) algorithm will guarantee that the preconditioner coincides with the inverse matrix?

a. 0

b. $\frac{n}{2}$

c. $+n$

6. Can the *ILU*(0) algorithm result in an inverse matrix?

    a. +Yes

    b. No.

7. What is the smallest *p* value for the *ILU*(*p*) algorithm to result in an inverse matrix?

    a. 1

    b. +elimination tree height

    c. n

8. Upon preconditioning, the matrix condition number must:

    a. Increase

    b. +Decrease

    c. Remain the same

9. If preconditioned, the numerical iterative methods have, as a rule

    a. +an improved convergence rate

    b. a reduced convergence rate

    c. more accurate convergence

10. The use of *ILU*(1) as compared to *ILU*(0) can

    a. +improve the convergence rate

    b. reduce the convergence rate