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

Lecture 2. Preconditioning methods

Nizhni Novgorod

2014

OBJECTIVES

The objective of this lecture is to review approaches to reduce the condition number of a matrix based on *preconditioning*. Preconditioning is important for iterative methods as it improves their convergence rate.

ABSTRACT

Preconditioning is an explicit or implicit transformation of a linear system $Ax=b$ which makes this system more suitable for solution using iterative methods as it improves their convergence rate. This lecture describes a number of preconditioning methods, from simpler (Jacobi and Seidel methods) to more effective ones such as *ILU(0)* and *ILU(p)* that are based on the incomplete *LU*-factorization. It gives experimental results for a model problem. The lecture also demonstrates matrix condition improvement as a result of various preconditioning methods.

GUIDELINES

One of the key properties of the 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 its 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.

Preconditioning is an explicit or implicit transformation of a linear system $Ax=b$ which makes this system more suitable for solution using an iterative method. For example, explicit preconditioning involves multiplying the matrix A lines by the reciprocals of the matrix diagonal to obtain a single main diagonal.

Another example of preconditioning is multiplying the initial system by some matrix, M^{-1}, i.e. transition to a system like $M^{-1}Ax = M^{-1}b$. The matrix M is called a *preconditioner matrix* or *preconditioner*. This preconditioning method may be called implicit as no explicit modification of the linear system takes place and the preconditioner matrix is used within the iterative method to correct certain algorithm steps.

An informal requirement to the preconditioner matrix may be put as follows: *M* must be close to *A* (for $M^{-1}A$ to be close to the unity matrix *E*). Please note that the selection of $M=A$ gives an immediate solution $Ex = A^{-1}b$, however, it has no practical sense as it requires computation of A^{-1}.

Product matrix computation $M^{-1}A$ generally leads to the matrix pattern modification (which is unacceptable for sparse systems), so the use of preconditioner requires another approach. Additional operations are introduced into the method scheme to allow for preconditioner influence. For example, the initial system residual $r = b - Ax$ has an obvious relation $M\bar{r} = r$ to the preconditioned system residual $\bar{r} = M^{-1}b - M^{-1}Ax$. The same is true for multiplication of a matrix by a vector.

Having summarized all prerequisites of effective preconditioning, we will see that the matrix *M*:

- must be close to *A*;
- must be easy to compute;
- must be easily invertible, i. e. allow for fast solution of systems like $Mz = r$.

As these three criteria are clearly controversial (as illustrated by the case when *M*=*A*), it is not easy to find the preconditioning method that will satisfy all of them. This lecture describes a number of preconditioning methods, from elementary ones like the Jacobi and Seidel methods to more efficient ones such as *ILU(0)* and *ILU(p)* based on incomplete *LU*-factorization.

The first part of the lecture deals with elementary preconditioners. First of all, there is a relation between the basic iterative methods (Jacobi, Seidel, SOR and SSOR) and the preconcitioner. Any basic iterative method is nothing else but the simple iteration method applied to a system with the preconditioner *M*. Here, *M* is based on the iteration matrix of the respective method.

These preconditioners satisfy two of the three requirements to them: they are easy to compute (as they are either submatrices or products of the matrix *A* submatrices) and invertible (as they are diagonal, triangular and a product of triangular matrices, respectively). However, they are not entirely in line with the third requirement, that is, they are not close to *A*, therefore, spectral parameters of the matrix $M^{-1}A$ are only slightly improved.

The second part of the lecture deals with preconditioners resulting from incomplete *LU*-factorization. Let us describe the matrix *A* as *A*=*LU*– *R*, where the matrices *L* and *U* are the lower and upper triangular matrices, respectively, and the residual *R* satisfies a number of additional conditions (e. g. specific *R* elements are equal to zero). In such a case, approximate representation of *A*≈*LU* is called *incomplete LU-factorization of the matrix A* (or *ILU-factorization*). If *R*=0, this will be 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 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 portraits. The numeric parts includes computation of specific values for all non-zero elements corresponding to the portraits.

Two incomplete factorization methods are considered, $ILU(0)$ and $ILU(p)$.

The idea of $ILU(0)$-factorization is to eliminate all new non-zero elements that appear in the course of factorization, 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 idea of $ILU(p)$-factorization 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). The method is based on the concept of *p filling level*. In this case, p regulates factor construction precision 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 $ILU(0)$.

The conclusion contains experimental results for a model problem. The lecture also demonstrates matrix condition improvement as a result of application of various preconditioning methods.

RECOMMENDATIONS FOR STUDENTS

A description of preconditioning methods may be found in numerous publications as such methods considerably improve the convergence rate of iterative methods used for solving linear systems. A concise summary of these methods (including pseudocode algorithms) is given in [1]. A detailed description of the approach itself including determination of methods, their theoretical justification and examples of use can be found in [4]. A brief description of preconditioning (as applicable to the conjugate gradient method) is given in [2, 3].

REFERENCES

1. J. Dongarra et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.
2. Gene H. Golub, Charles F. Van Loan. Matrix Computations. The John Hopkins University Press, 1996.
3. James W. Demmel. Applied Numerical Linear Algebra. SIAM, 1997.
4. Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

PRACTICE

1. Construct the symmetric Gauss-Seidel preconditioner. Estimate the matrix A condition number reduction using this preconditioner.
2. Construct the $ILU(p)$-preconditioner. Estimate the matrix A condition number reduction using this preconditioner with various degrees of p filling.
3. Implement a parallel version of the $ILU(p)$-preconditioner. Check your program scalability.

TEST

1. Which property complicates the use of the matrix M as a preconditioner?
   a. + Matrix M is ill-invertible
   b. Matrix M is ill-conditioned
   c. Matrix M has complex eigenvalues
2. Which of the following properties complicates the use of the matrix M as an explicit preconditioner (i. e. complicates transition to the $M^{-1}A$ system)?
   a. Matrix $M^{-1}A$ will be sparse
   b. Matrix $M^{-1}A$ will be well-conditioned
   c. + Matrix $M^{-1}A$ will be dense
3. The use of the preconditioner M in an iterative method involves
   a. Computation of the matrix M^{-1} and transition to the problem with the matrix $M^{-1}A$.
   b. +Solving the system with the matrix M within each method iteration
   c. Solving the system with the matrix M within the first method iteration
4. The Jacobi preconditioner (in the $A = R+D+L$ assumption) is computed as
   a. + D
   b. $(D+L)$
   c. $(D+R)$

5. The Seidel preconditioner (in the $A = R+D+L$ assumption) is computed as
    a. D
    b. + $(D+L)$
    c. $(D+R)$
6. Matrices L and U of the $ILU(0)$-preconditioner have
    a. +The same portrait as the matrix A
    b. The same portrait as the matrix A^{-1}.
    c. The same portrait as the matrix A^T.
7. The procedure of ILU(0)-preconditioner computation is based on
    a. +Gaussian elimination method
    b. Gauss-Seidel iterative method
    c. Cyclic reduction method
8. $ILU(p)$-preconditioning method
    a. Does not allow filling of the matrices L and U if compared to the matrix A
    b. +Allows filling of the matrices L and U if compared to the matrix A
    c. Filling of the L and U matrices is not controlled by the method.