Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

121

Zahari Zlatev
Jerzy Wasniewski
Kjeld Schaumburg

Y12M

Solution of Large and Sparse Systems of Linear Algebraic Equations

Springer-Verlag
Berlin Heidelberg New York

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

121

Zahari Zlatev
Jerzy Wasniewski
Kjeld Schaumburg

Y12M

Solution of Large and Sparse Systems of Linear Algebraic Equations
Documentation of Subroutines

Springer-Verlag
Berlin Heidelberg New York 1981

Authors

Zahari Zlatev
Computer Science Department, Mathematical Institute
University of Aarhus, Ny Munkegade, DK 8000 Aarhus C

Jerzy Wasniewski
The Regional Computing Centre at the University of Copenhagen
Vermundsgade 5, DK 2100 Copenhagen

Kjeld Schaumburg
Department of Chemical Physics, University of Copenhagen
The H.C. Oersted Institute, Universitetsparken 5,
DK 2100 Copenhagen

AMS Subject Classifications (1979): 15 A 04, 15 A 06, 15 A 23, 65-04,
65 F 05, 65 F 10, 68 E 05
CR Subject Classifications (1981): 5.14, 4.6

ISBN 3-540-10874-2 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-10874-2 Springer-Verlag New York Heidelberg Berlin

CIP-Kurztitelaufnahme der Deutschen Bibliothek
Zlatev, Zahari:
Y12M [YM] solution of large and sparse systems of linear algebraic equations:
documentation of subroutines / Zahari Zlatev; Jerzy Wasniewski; Kjeld Schaumburg.
– Berlin; Heidelberg; New York: Springer, 1981.
(Lecture notes in computer science ; 121)
ISBN 3-540-10874-2 (Berlin, Heidelberg, New York);
ISBN 0-387-10874-2 (New York, Heidelberg, Berlin)
NE: Wasniewski, Jerzy:; Schaumburg, Kjeld:; GT

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

Preface

The Y12M is a package of Fortran subroutines for the solution of large and sparse systems of linear algebraic equations developed at the Regional Computing Centre at the University of Copenhagen (RECKU). Gaussian elimination and pivotal interchanges are used to factorize the matrix of the system into two triangular matrices L and U. An attempt to control the magnitude of the non-zero elements in order to avoid overflows or underflows and to detect singularities is carried out during the process of factorization. Iterative refinement of the first solution may be performed. It is verified (by a large set of numerical examples) that iterative refinement combined with a large drop-tolerance and a large stability factor is often very successful when the matrix of the system is sparse. Not only is the accuracy improved but the factorization time is also considerably reduced so that the total computing time for the solution of the system with iterative refinement is less than that without iterative refinement (in some examples the total computing time was reduced by more than three times). The storage needed can often be reduced also.

Note that if the matrix of the system is dense then the total computing time for the iterative solution of the system is always larger (because extra time must be used to perform the iterations needed to improve the accuracy of the first solution). Note too that the use of iterative refinement with dense matrices leads to an increase of the storage by a factor approximately equal to 2 (because a copy of the matrix of the system must be kept).

The factorization found by the use of large values of the drop-tolerance is sometimes referred to as incomplete. The matrix LU obtained in this way can be considered as a preconditioned matrix. Preconditioned matrices are often used when systems with symmetric and positive definite matrices are solved by the conjugate gradients method. When the matrices are

general the conjugate gradients method can not be used (at least in its classical form). In our package iterative refinement is used instead of the conjugate gradients method. The experiments show that this approach is normally extremely efficient. This is especially true for expensive problems, i.e. problems whose matrices are such that many fill-ins are produced in the process of the LU decomposition. It should be mentioned that efficency is most required just for such problems.

It is necessary to emphasize that a reliable error estimation will normally be obtained when the iterative refinement process is convergent. No error estimation can be found when the system is solved directly. However, one can expect that the required accuracy will be achieved, especially when double precision is used, if the condition number of the coefficient matrix is not extremely large. A subroutine which evaluates the condition number of a matrix is available and may optionally be called. If this subroutine is called (which can be done when the LU decomposition is calculated), then a reliable measure of the sensitivity of the results to the round-off errors will be obtained. It must be emphasized that the use of the subroutine for evaluation of the condition number is relatively cheap; its computational cost is equal to the computational cost for two back substitutions.

There exist problems for which the application of iterative refinement is not very efficient with regard to the storage and computing time used (e.g. when many systems with the same coefficient matrix are to be solved). Therefore the iterative refinement process should in our opinion be only an option in the package for the solution of large and sparse systems of linear algebraic equations. Using some machine dependent facilities (paging, multibanking etc.) one can modify the iterative refinement option so that it will never use more storage than the direct solution option. The price which should be paid for this is a modest increase of the computing time. A modified version of the iterative refinement option which uses multibanking has been developed for Univac 1100 series computers at RECKU (the Regional Computing Centre at the University of Copenhagen).

All subroutines of the package have been run on three different computers: a Univac 1100/82 computer at the Regional Computing Centre at the University of Copenhagen

(RECKU), an IBM 3033 computer at the Northern Europe University Computing Centre (NEUCC) and a CDC Cyber 173 computer at the Regional Computing Centre at the University of Aarhus (RECAU). Some results from these runs are reported in this book.

If reference is made to a paper or a book on some page, then its title, the name(s) of the author(s) and the place of publication appear as a footnote on the same page. Thus the superscript in any reference indicates the number of the footnote where a detailed information about the reference is given. All references are also listed at the end of this book. The INDEX can be used to check the page where a reference to any paper or book is made.

The codes of the subroutines with full documentation are in RECKU Library. They are available at the usual costs (for the magnetic tape, machine time, shipment, etc.). The requests should be addressed to J. Wasniewski. Advances in theory and the experience of users may prompt alterations in these codes. Readers and/or users are invited to write to the authors concerning any changes they may advocate.

CONTENTS

1. Introduction to Y12M

1.1. Scope of the Y12M

Y12M is concerned with the calculation of the solution of systems of linear algebraic equations whose matrices are large and sparse.

Two matrices A and B are of the same structure
if $a_{ij} \neq 0$ implies $b_{ij} \neq 0$ and $b_{ij} \neq 0$ implies $a_{ij} \neq 0$.

The following notation is useful in the classification: matrices denoted by the same letters and subscripts are the same, matrices denoted by the same letters and different subscripts are of the same structure (but different), matrices denoted by different letters are of different structure.

We consider the solution of the following problems:

(i) Only one system with a single right-hand side, $Ax = b$, is to be solved.

(ii) A sequence of systems with the same matrices is to be solved. The matrices of this sequence are

$$A_1, A_1, \ldots, A_1.$$

The case where one system with many right-hand sides is to be solved can easily be presented as a sequence of systems with the same matrices.

(iii) A sequence of systems whose matrices are different but of the same structure is to be solved. This means that the matrices of this sequence are

$$A_1, A_2, ..., A_p.$$

(iv) A sequence of systems whose matrices are of the same structure and some of them (but not all of them) are different is to be solved. This means that the matrices of this sequence are

$$A_1, A_1, ...,A_1; A_2, A_2, ..., A_2; ... ; A_p, A_p, ..., A_p.$$

(v) A sequence of systems whose matrices are of a different structure is to be solved. The matrices of this sequence are

$$A, B, C, ..., Z.$$

There are recommendations for the use of the package in each of the above five cases.

1.2. Background to the Problem

1.2.1. Storage Operations

Given a matrix A with Z non-zero elements. These elements are stored (ordered by rows) in the first Z positions of array A. Their column numbers are stored in the same positions of array SNR. Arrays A and SNR form the row ordered list. The row numbers of the non-zero elements (ordered by columns) are stored in the first Z positions of array RNR. Array RNR

forms the column ordered list. This storage algorithm was first proposed by Gustavson[1)2)] . During the computation the non-zero elements that are not necessary in subsequent stages are removed from the lists. Unfortunately, some new non-zero elements (fill-ins) are created and placed at the beginning or at the end of the row (column) if there are free locations. Otherwise, a new copy of the row (column) at the end of the row (column) ordered list is made, thus freeing the locations originally reserved for the row (column). Obviously there is a limit to the number of new copies that can be made without exceeding the capacity of the arrays, and therefore occasional "garbage" collections are necessary. More details about the storage scheme used in our subroutines can be found in Zlatev, Schaumburg and Wasniewski[3)] and Zlatev[4)] .

1.2.2. Mathematical Method

Gaussian elimination is used with interchanges. The system

$$PAQ(Q^Tx) = Pb$$

(where P and Q are permutation matrices) is replaced by

$$LU(Q^Tx) = Pb$$

1. Gustavson, F.G. - "Some Basic Techniques for Solving Sparse Systems of Linear Equations", In: "Sparse Matrices and Their Applications" (D.J. Rose and R.A. Willoughby, eds.), pp. 41-52, Plenum Press, New York, 1972.

2. Gustavson, F.G. - "Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition", ACM Trans. Math. Software, 4, pp. 250-269, 1978.

3. Zlatev, Z., Schaumburg, K. and Wasniewski, J. - "Implementation of an Iterative Refinement Option in a Code for Large and Sparse Systems". Computers and Chemistry, 4, pp. 87-99, 1980.

4. Zlatev, Z. - "Use of Iterative Refinement in the Solution of Sparse Linear Systems", Report 1/79, Institute of Mathematics and Statistics, The Royal Veterinary and Agricultural University, Copenhagen, Denmark, 1979 (to appear in SIAM J. Numer. Anal.).

(where L is a unit lower triangular matrix and U is an upper triangular matrix). Then the latter system is solved by forward and back substitution and (normally) an approximation to the solution vector x is computed. If iterative refinement is to be used then we denote the above solution vector by x_1 and perform the following successive calculations:

$$r_{k-1} = b\text{-}Ax_{k-1},$$

$$d_{k-1} = QU^{-1}L^{-1}Pr_{k-1},$$

$$x_k = x_{k-1} + d_{k-1},$$

where k = 2, 3, 4, ...

(various stopping criteria must be used in order to terminate the process if the required accuracy is achieved or if the process is not convergent). Normally the accuracy will be improved if iterative refinement is applied in the calculations of the solution of linear systems.

More details about the theory of the Gaussian elimination can be found e.g. in Forsythe and Moler[5], Stewart[6] and Wilkinson[7][8].

5. Forsythe, G.E. and Moler, C.B. - "Computer Solution of Linear Algebraic Equations", Prentice-Hall, Englewood Cliffs, N.J., 1967.

6. Stewart, G.W. - "Introduction to Matrix Computations", Academic Press, New York, 1973.

7. Wilkinson, J.H. - "Rounding Errors in Algebraic Processes", Prentice-Hall, Englewood Cliffs, N.J., 1963.

8. Wilkinson, J.H. - "The Algebraic Eigenvalue Problem", Oxford University Press, London, 1965.

1.3. Recommendations on the Use of the Routines

1.3.1. Choice of Pivotal Strategy

Interchanges are normally used in order to preserve the sparsity of the original matrix (i.e. to minimize the number of non-zero elements created during the factorization) and to ensure numerically stable computations (i.e. to attempt to prevent the occurrence of large errors in the solution vector).

The choice of the pivotal strategy depends on matrix A. We have three pivotal strategies, each of which may be used by appropriate initialization of the parameter IFLAG(3) before the call of the package. For general matrices A, IFLAG(3) = 1 must be used. In this case the number of rows that will be investigated at each stage of the elimination in order to determine the pivotal element must be given as well. This number must be initialized in IFLAG(2) and should not exceed three. A generalized Markowitz strategy is used[9] , i.e. among the elements of the selected IFLAG(2) rows with least numbers of non-zero elements, the element which satisfies the stability requirement and for which the product of the other non-zero elements in its row and the other non-zero elements in its column is a minimum will be chosen as a pivotal element. Moreover, if there are several such elements, the largest in absolute value will be chosen (more details can be found in Zlatev[9] and Zlatev, Schaumburg and Wasniewski[10]).

If the use of pivotal elements on the main diagonal is sufficient (and this is the case if e.g. the matrix is symmetric and definite or diagonally dominant) then the package can take advantage of this property. IFLAG(3) should be initialized with the value 2 before the call of

9. Zlatev, Z. - "On Some Pivotal Strategies in Gaussian Elimination by Sparse Technique", SIAM J. Numer. Anal., 17, pp. 18-30, 1980.

10. Zlatev, Z., Schaumburg, K. and Wasniewski, J. - "Implementation of an Iterative Refinement Option in a Code for Large and Sparse Systems". Computers and Chemistry, 4, pp. 87-99, 1980.

Y12M.

In rare cases no pivoting is necessary. The sparsity pattern of the matrix and the numerical stability of the computation will be preserved if no pivotal interchanges are made. In this case IFLAG(3) should be initialized with the value zero before the call of the Y12M. This choice applies when matrix A is symmetric and definite or diagonally dominant and moreover, nearly all non-zero elements are located not far from the main diagonal. When the iterative refinement (IR) option is used then computations without pivoting are possible for quite general matrices. This has successfully been demonstrated by Schaumburg *et al.*[11] .

Note that the use of the two special strategies may reduce the computing time considerably. In our experiments with positive definite matrices and with the use of the above special strategies, the computing time needed to solve the systems by our package is comparable to the computing time needed to solve the same systems with codes especially written to be used with positive definite matrices, see e.g. Zlatev, Wasniewski and Schaumburg[12] .

1.3.2. Robustness

In our opinion the code for the solution of large and sparse systems of linear equations should attempt to detect any of the following situations: (a) the elements of the matrix grow too quickly (then the subsequent computations are not justified, and even overflow may take place), (b) very small elements appear in the computation (so that underflows are possible), (c) the matrix is singular (or nearly singular, so that the machine accuracy will not be sufficient to compute an acceptable solution).

11. Schaumburg, K., Wasniewski, J. and Zlatev, Z. - "The Use of Sparse Matrix Technique in the Numerical Integration of Stiff Systems of Linear Ordinary Differential Equations". Computers and Chemistry, 4, pp. 1-12, 1980.

12. Zlatev, Z., Wasniewski, J. and Schaumburg, K. - "Comparison of Two Algorithms for Solving Large Linear Systems". Report No 80/9, Regional Computing Centre at the University of Copenhagen, Vermundsgade 5, DK-2100 Copenhagen, Denmark, 1980.

Below we describe how these problems are handled by the routines.

(a). Denote $a = \max|a_{ij}|$ where a_{ij}, i,j = 1,2,...N, are the elements of the original matrix A. Denote $b_k = \max|a_{ij}^{(s)}|$ where k = 1,2,...., N-1; s = 1,2,...,k; i,j = s,s+1,...,N and $a_{ij}^{(s)}$ are elements of matrix A which are to be transformed by the Gaussian transformations at stage s of the elimination. Let u be the stability factor (where $u \geq 1$ and u should be initialized in AFLAG(1) before the call of the package). It can then be shown that $b_k/a \leq (1+u)^k$ (for k = 1,2,...,N-1) and even examples where $b_k/a = (1+u)^k$ may be constructed. Moreover, during the computation we compute $\bar{L}\bar{U} = PAQ+E$ (instead of LU = PAQ) and for the elements e_{ij} of the pertubation matrix E it is true that $|e_{ij}| \leq 3.01 b_{N-1}\varepsilon$ (where ε is the machine accuracy)[13]. Therefore it is clear that if b_k (stored by the subroutine Y12MC in AFLAG(7)) grows very quickly then the computed $\bar{L}\bar{U}$ will be inaccurate and in an extreme case overflow may occur. We monitor the largest computed elements and the growth factor b_k/a and if this factor is larger than a number prescribed by the user (this number should be initialized in AFLAG(3), we recommend 10^{16}) the package will stop the computation and give the error indication that the growth factor is too large. Note that smaller values of the stability factor AFLAG(1) tend to decrease the growth factor.

(b). Sometimes very small elements appear during the computation. The drop-tolerance T (stored in AFLAG(2)) may successfully be used in this situation. If during the computation $|a_{ij}^{(k)}| < T$ then the subroutine Y12MC will remove this element from the lists. In our experiments even the use of very small values of the drop-tolerance (say $T = 10^{-30}$) has proven very efficient in the cases where underflows were registered by the use of T = 0. Note that some compilers stop the computation and give error diagnostics if underflows appear. We recommend the use of $T = 10^{-12}$. It must be emphasized here that larger values for the drop-tolerance may be used with iterative refinement.

(c). Let the rank of matrix A be $k < N$. Then all elements in submatrix A_{k+1}, obtained after k stages of Gaussian elimination, should be equal to zero, but in general they are not because

13. Reid, J.K. - "A Note on the Stability of Gaussian Elimination", J. Inst. Math. Appl., 8, pp. 374-375, 1971.

of round-off errors (the elements of matrix A_{k+1} are all elements whose row and column numbers are larger than or equal to k+1). Nevertheless, all the elements of A_{k+1} and the successive submatrices are normally small and it is possible (and in our examples it was so) that they will be removed and a row and/or a column without non-zero elements will be found (in this case IFAIL = 7 or 8 on exit will normally indicate that the matrix is numerically singular, except in the cases where the drop-tolerance is too large).

Another means of detecting singularity is the check for the minimal pivotal element (stored on exit in AFLAG(8)). One might check the whole sequence of pivotal elements (stored in array PIVOT, renamed Y in subroutine Y12MF). A very small pivotal element will indicate that the matrix is nearly singular (if AFLAG(1) is not too large). Note that if the absolute value of the current pivotal element is smaller than AFLAG(4) ★ AFLAG(6) then the routine stops the computation. A very small number must be initialized in AFLAG(4) before the entry (we recommend AFLAG(4) = 10^{-12}; the largest in absolute value element of the original matrix is stored by subroutine Y12MB in AFLAG(6)).

1.3.3. Storage of Matrix L

The principle of storing the non-zero elements of the unit lower triangular matrix L is automatically taken from the solution of systems with dense matrices (where this operation does not require extra storage and extra time). In the case of systems with sparse matrices the storage of L requires about 40% larger arrays A and SNR. Extra computing time is often needed too. Therefore L must be stored only if the problem justifies it (i.e. we have problems of type (ii) or (iv))[14]. Moreover, note that some very large problems of type (i), (iii) or (v) may be solved without the use of secondary storage (as for example discs) only if the non-zero elements of matrix L are removed. If the non-zero elements of L are not required then IFLAG(5) = 1 must be initialized (the length NN of arrays A and SNR may be reduced in this case and NN = 2★Z will often be enough). If the non-zero elements are to be stored then

14. See Section 1.1 "Scope of Y12M"