IMPROVING THE EFFICIENCY OF INCOMPLETE

CHOLESKI PRECONDITIONINGS

MANOLIS PAPADRAKAKIS AND MICHAEL C. DRACOPOULOS

Institute of Structural Analysis and Aseismic Research, National Technical University, Zografou Campus,
GR-15773, Athens, Greece

SUMMARY

Techniques based on the idea of preconditioning have significantly improved the efficiency of classical
iterative methods. In this work two preconditioning approaches based on incomplete Choleski
factorization have been further refined, with the result that both storage requirements and solution times
have been greatly improved. The first is based on the rejection of certain terms of the factorization
process according to their magnitude. An efficient computational handling of the incomplete
factorization is proposed which improves substantially the complicated addressings and the overall
efficiency of the iterative method. The second preconditioning matrix is based on a rejection criterion
relative to the position of the non-zero terms of the coefficient matrix. Such preconditioners have been
proposed in the past but experience with the application of the finite-element method is not favourable
since the factorization very often becomes unstable or the rate of convergence is not satisfactory. The
modifications proposed here give this type of incomplete factorization both robustness and efficiency.

1. INTRODUCTION

Iterative methods are widely used for the solution of linear systems resulting from the finite-
element method where storage requirements are excessive. Besides, the ease of implementation
and the possibility of controlling the accuracy of the solution have made them very attractive
to many applications. Techniques based on the idea of preconditioning have significantly
improved the efficiency of these methods. Under this concept the initial equation

\[ Ax = b \]  

is transformed into

\[ (R^{-1}A)x = R^{-1}b \]  

where the transformation matrix \( R \) is an approximation to \( A \) and is non-singular.

Iterative procedures for solving (2), such as the conjugate gradient the dynamic relaxation
and Chebyshev semi-iterative methods, can be expressed by the three-term recursion formula

\[ x^{(m+1)} = \alpha_m R^{-1}r^{(m)} + (\beta_m + 1)x^{(m)} - \beta_m x^{(m-1)} \]  

and

\[ r^{(m)} = Ax^{(m)} - b \]  

In order to evaluate the expression \( R^{-1}r^{(m)} \) involved in equation (3). \( R \) is usually defined as

\[ R = LL^T \]  

0748-8025/91/080603-10$05.00
© 1991 by John Wiley & Sons, Ltd.

Received October 1989
Revised October 1990
with the preconditioning matrix $L$ being lower triangular. The most widely used triangular matrices $L$ are derived either from the incomplete Choleski factorization of the coefficient matrix\textsuperscript{3-11} or by the SSOR characteristic matrix.\textsuperscript{12-14}

Some of the existing incomplete Choleski preconditioners are strictly oriented to matrices of a special kind, such as diagonal dominant\textsuperscript{9} or M-matrices,\textsuperscript{11} while problems arising from the application of the finite-element method to structural mechanics in general cannot be classified under any category and need more efficient and robust preconditioners.\textsuperscript{4,5,10}

The scope of this paper is to investigate the potentialities of some of these more general methods when applied to structural analysis problems and in particular the partial elimination proposed by Jennings and co-workers.\textsuperscript{4,5} A computational approach is described which eliminates most of the disadvantages of this method such as the complicated addressings and the high demands for auxiliary storage during the formation of the preconditioning matrix. An inconsistency between the theoretical implementation proposed in Reference 4 and in the included computer program is also recorded. Finally, a new incomplete Choleski preconditioning technique is proposed which is simple, effective, stable and general enough to work satisfactorily with structural analysis problems.

2. INCOMPLETE CHOLESKI FACTORIZATION

The widely used type of preconditioners derived from an incomplete Choleski factorization of the coefficient matrix $A$ is defined as

$$A = LL^T - C$$

where $C$ is called the error matrix and is the matrix of elements omitted from the factorization in order to curtail fill-in. For this class of methods, $C$ is defined either by the prescribed positions of the elements to be rejected\textsuperscript{6,7,9,11} or by the computed positions of ‘small’ elements in $L$, which do not satisfy a specified magnitude criterion and therefore are discarded.\textsuperscript{4,5,8} The rejection of certain elements inside the preconditioning matrix often leads to an unstable factorization process. To avoid this, special care has to be taken by some suitable modifications.

The incomplete factorization procedure can be described using a row-by-row approach and referring to $L^T$ rather than $L$, as follows.

For each row, $i := 1, \ldots, N$

$$a^*_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ki}l_{kj}, \quad j := i + 1, \ldots, N$$

$$l_{ii} = \left( a^*_{ii} - \sum_{k=1}^{i-1} l_{ki}l_{ki} \right)^{1/2}$$

$$l_{ij} = a^*_{ij}/l_{ii} \quad \text{if } l_{ij} \text{ is retained }$$

$$l_{ij} = 0 \quad \text{if } l_{ij} \text{ is discarded}$$

(7) \hspace{2cm} (8) \hspace{2cm} (9)

Note that $a^*_{ii}, a^*_{ij}$ can be either the corresponding elements of $A$ or some modified ones according to the preconditioning method employed.

2.1. Rejection by magnitude

The preconditioned technique proposed by Jennings and co-workers,\textsuperscript{4,5} which is based on rejection by magnitude, is robust and presents good convergence characteristics when applied
IMPROVING EFFICIENCY OF INCOMPLETE CHOLESKI PRECONDITIONINGS

...to structural analysis problems. According to this approach, after each element \( a_{ij}^* \) is formed as in equation (7), with \( a_{ij} = a_{ij} \), it is tested for magnitude. The criterion used to decide whether \( a_{ij}^* \) is large or small is to specify a drop-off parameter \( \psi \) and then to consider all elements small which satisfy

\[
a_{ij}^* < \psi \tilde{a}_{ii} \tilde{a}_{jj}
\]

At the time the rejection test is carried out with element \( a_{ij}^* \) the current values of the diagonal elements are

\[
\tilde{a}_{ii} = \left( a_{ii} + \sum_{k=1}^{i-1} c_{ii}^{(k)} - \sum_{k=1}^{i-1} l[k,l_k] \right) + \sum_{k=i+1}^{j-1} c_{jj}^{(k)}
\]

\[
\tilde{a}_{jj} = a_{jj} + \sum_{k=1}^{i-1} c_{jj}^{(k)}
\]

The diagonal modification employed to retain stability when \( a_{ij}^* \) is rejected are computed from

\[
c_{ii}^{(j)} = (\tilde{a}_{ii} \tilde{a}_{jj})^{1/2} | a_{ij}^* |
\]

\[
c_{jj}^{(j)} = (\tilde{a}_{jj} \tilde{a}_{ii})^{1/2} | a_{ij}^* |
\]

In addition, \( a_{ii}^* \) in equation (8) is given by

\[
a_{ii}^* = a_{ii} + \sum_{k=1}^{i-1} c_{ii}^{(k)} + \sum_{k=i+1}^{N} c_{ii}^{(k)}
\]

The choice of \( \psi = 0 \) will retain all elements within the elimination and is therefore equivalent to a complete Choleski factorization, while the case of \( \psi = 1 \) corresponds to a form of diagonal scaling. The diagonal modifications employed produce an error matrix \( C \) that is always positive semi-definite, so the lowest eigenvalue of \( LL^T \) cannot be less than that of \( A \) and stability is assured.

According to the FORTRAN listing included in Reference 4 the current values of the diagonal elements in equation (11) are

\[
\tilde{a}_{ii} = \left( a_{ii} + \sum_{k=1}^{i-1} c_{ii}^{(k)} - \sum_{k=1}^{i-1} l[k,l_k] \right)
\]

\[
\tilde{a}_{jj} = a_{jj} + \sum_{k=1}^{i-1} c_{jj}^{(k)}
\]

while the diagonal modifications are given by

\[
c_{ii}^{(j)} = (\tilde{a}_{jj} \tilde{a}_{ii})^{1/2} | a_{ij}^* |
\]

\[
c_{jj}^{(j)} = (\tilde{a}_{ii} \tilde{a}_{jj})^{1/2} | a_{ij}^* |
\]

instead of equation (12). Element \( a_{ii}^* \) in equation (8) is given by

\[
a_{ii}^* = a_{ii} + \sum_{k=1}^{i-1} c_{ii}^{(k)} + c^*
\]

with \( c^* = 0 \), and only when the last element of the row is rejected is \( c^* \) set equal to the modification described by the first of equations (15). With these diagonal modifications the stability of the method is not ensured, as it will be verified in the numerical applications presented.
2.2. Rejection by position

The incomplete factorization based on rejection by magnitude, although it is an efficient preconditioning method for structural analysis problems, has the disadvantage that the storage size and pattern are not known at the start and the cost in computer space may become high since some auxiliary vectors will be needed for storing the preconditioning matrix. On the other hand, if a suitable rejection parameter is not selected the first time, it is possible to attempt to factorize the coefficient matrix more than once before a proper value for $\psi$ is found that takes advantage of the available storage.

The incomplete Choleski factorization based on rejection by position is based on a sparsity pattern which is to be forced on $L$. Thus, according to this approach, whenever an $l_{ij}$ element does not belong to the specified pattern $(i,j)$ it is set equal to zero and is neither calculated nor stored. A storage pattern which is very attractive, since extra addressings are avoided, is defined by keeping only the factorized elements which correspond to the non-zero entries of $A$.

Incomplete factorizations proposed in the past based on this concept do not work satisfactorily compared with other types of preconditioners. Our experience in structural analysis problems has shown that the modified incomplete factorization proposed by Gustafsson does not work satisfactorily even with the improvements introduced in Reference 10. According to the technique of Reference 9 elements omitted from the factorization are added to the corresponding diagonal elements of $L$. However, since negative entries may move to the diagonal, this may cause problems with the ellipticity of the matrix. It is therefore proposed in Reference 10 to move the negative elements to a positive off-diagonal in the same row if such a term exists. As a final remedy it is suggested in both References 9 and 10 that a diagonal term should be replaced by a positive one whenever it turns out to be negative.6

An alternative formulation is proposed in this work in which the diagonal modifications described in equation (12) are applied whenever an element $l_{ij}$ is rejected. The result is that, irrespective of the enforced sparsity pattern, the error matrix $C$ remains positive semi-definite and the stability of the process is ensured, while the efficiency of the preconditioning matrix is maintained.

3. COMPUTER IMPLEMENTATION

A compact storage scheme is used to store both the coefficient and the preconditioning matrix row-by-row. Non-zero terms are stored in a real array, while the corresponding column numbers are stored in an integer array of equal length. An additional integer array with length equal to the number of equations is used to record the start of each row inside the compact scheme. Thus, the storage requirements for an incomplete procedure by magnitude are $NA + NL$ real (the size of $A$ and $L$ respectively) and $NA + NL + 2* (N + 1)$ integer stores (for the addressings), with $N$ being the number of equations. For the incomplete procedure by position, where the preconditioning and the coefficient matrix share the same auxiliary integer vectors, the storage is $2 * NA$ real and $NA + N + 1$ integer positions. The extra storage required by the conjugate gradient method is $4N$ real positions assuming that the solution vector overwrites the right-hand-side vector.

The computational methodology which is described in Reference 4 needs, for the formation of $L^T$, $NL + 2 * MAXK$ integer stores which are credited as additional storage indispensable during the factorization phase. $MAXK$ is the maximum number of off-diagonal elements in any column of $L^T$, while $NL$ represents the dimension of an integer link array used to scan each column of $L^T$ travelling upwards. This additional storage is one of the most serious
defects of the initial incomplete factorization by magnitude, since in order to gain maximum advantage of the available storage, $4 \cdot N$ real stores should overlap $NL + 2 \cdot \text{MAXK}$ integer stores. This situation is not likely to occur as the size of the problem increases, even when integer variables of reduced wordlength are available in the computer system.

However, it is not necessary to use so much additional storage in order to perform the incomplete factorization. Actually only three integer vectors of length $N$ need to exist at the factorization phase. Let us suppose that they are denoted by $C_1$, $C_2$, $C_3$. $C_3(I)$ stores the minimum row with non-zero entry in column $I$ of $L^T$, or the skyline of each column. At the time row $I$ of $L^T$ is processed $C_1(M)$ is the position of the non-zero element of $L^T$ in row $M$ ($M = C_3(I), ..., I - 1$) whose column number is equal to or greater than $I$. As the element $(I, J)$ of $L^T$ is processed, $C_2(M)$ is the position of the non-zero element of $L^T$ in row $M$ ($M = C_3(J), ..., I - 1$) whose column number is equal to or greater than $J$. Thus, using $C_1$, $C_2$ it is possible to compute the sum in equations (7) and (8) scanning the current columns upwards. Additionally $C_3$ is used to indicate the beginning of this procedure in order to minimize the time needed to compute those sums.

Assuming that the non-zero terms of $L^T$ are stored in the real array $LT$, their corresponding column numbers in the integer array $LTC$ and the start of each row in the integer array $LTADD$, the computation of

$$
\sum_{k=1}^{i-1} l_{ik} l_{kj}
$$

in equation (7) can be described by the following algorithm:

```python
sum := 0.0;
for k := max(c3[i], c3[j]) to i - 1 do
    begin
        m0 := ltadd[k + 1];
        m1 := c1[k];
        m2 := c3[k];
        if (ltc[m1] = i and ltc[m2] = j) then
            begin
                sum := sum + lt[m1] * lt[m2];
                if (m2 + 1 < m0) then c2[k] := m2 + 1
            end;
    end;
A listing of the computer program is reported in Reference 15.

Following the implementation of Reference 4, the computer implementation of partial elimination in this work is oriented mainly to the von Neumann computer architecture. However, the compact data structure used, also known as Yale format, is generally not desirable for vector processors. In this case, the proposed storage pattern for the array indexing information can be incorporated into more vectorizable storage schemes, as for example the Ellpack matrix format.\textsuperscript{16}

4. NUMERICAL EXAMPLES

The preconditioning methods discussed, as well as the SSOR preconditioner, were applied to three structural analysis test problems. The computer used for the comparisons was a SUN 3/50 workstation without math-coprocessor. The basic iterative method employed is the
Figure 1. Example 1. 147 d.o.f. space truss. 4PEL1, UPEL2, +PEL1, +PEL2, (a) remaining terms in $L$; (b) total storage requirements; (c) number of iterations; (d) computing time (sec).
Figure 2. Example 2. 663 d.o.f. plane frame. — PEL1, — PEL2, — PELP, — SSOR, — PEL1+, — PEL2+: (a) remaining terms in $L$; (b) total storage requirements; (c) number of iterations; (d) computing time (sec)
Figure 3. Example 3. 2883 d.o.f. plane frame. --- PEL1, O PEL2, X PELP, SSOR, - - PEL1+, - - - PEL2+: (a) remaining terms in \( L \); (b) total storage requirements; (c) number of iterations; (d) computing time (sec)
conjugate gradient method. For each example the number of iterations, the computing time for solution, the number of non-zero terms in the preconditioning matrix and the total storage requirements including the four conjugate gradient vectors were recorded. In estimating the computer storage it was assumed that both integer and floating point variables occupy the same space in the computer memory. In the presentation of the results PEL1+ and PEL2+ are abbreviations of the theoretical and computational versions of Reference 4, given by equations (12), (13) and Equations (15), (16), respectively. The same implementations, using the proposed data storage scheme for the array indexing information, are abbreviated to PEL1 and PEL2, respectively. The proposed incomplete factorization by position is abbreviated to PELP.

The first example is an extremely ill-conditioned space truss with 147 d.o.f., while the other two are plane frames with 663 and 2883 d.o.f. respectively. Plane frames were chosen because the presence of both sway and axial stiffness produces poor conditioning and hence the convergence rates of classical iterative methods are very slow. The initial estimation for the solution vector was set equal to zero and the iterative procedure was terminated when the norm of the residual vector divided by the corresponding norm of the right-hand-side vector was less than or equal to the required tolerance. A tolerance of $10^{-7}$ was specified for all examples.

As can be seen from the results (Figures 1–3), PEL1 showed robustness in its behaviour with the number of iterations increasing as the number of elements in $L$ decreases. On the other hand, PEL2 cannot be trusted since in example 1 the factorization is unstable for $0.000005 < \psi < 0.0001$. However, for the space truss and for the stable values of $\psi$ PEL2 is better in terms of iterations and computer time than PEL1. Nevertheless, the opposite behaviour is observed in the other two examples.

The significant storage reduction achieved for all examples due to the proposed computer implementation should also be noted while the additional overhead in runtime due to the proposed storage scheme is not important. As can be seen from the storage demands of PEL1+, PEL2+ and for small values of $\psi$, the original implementation proposed in Reference 4 tends to require storage similar to or greater than the direct solution method because of the additional storage overhead required for the complicated addressings. Although, for the small sized examples 1 and 2 the storage gainings, when applying PEL with small values of $\psi$, do not seem to be so significant compared to the direct method, an indication of the storage trends could be seen in example 3, which is a moderately large problem.

On the other hand, the performance of PELP is very promising. This preconditioning approach is independent of a drop-off parameter, is competitive with PEL for a large spectrum of $\psi$ values, while it is far better in terms of storage requirements. Comparing PELP with SSOR it can be seen that, despite the small storage overhead, it is better in terms of number of iterations and computer time. For the second example the modified incomplete Choleski factorization, proposed by Gustafsson in Reference 9, was also tested, as well as an improved version which appeared in Reference 10. The results showed extremely poor convergence rates for these two versions, and as an indication they did not manage to achieve a tolerance of $0.1$ after $N$ iterations.

5. CONCLUSIONS

The incomplete Choleski preconditioners presented contribute to further improving the efficiency of iterative methods when large sets of linear equations are solved. The proposed
computer implementation of the incomplete factorization by magnitude results in a reduction of approximately 25 per cent in the total computer storage requirements for small values of the rejection parameter $\psi$ at the expense of an insignificant overhead in runtime, while on the other hand the algorithm becomes less complicated than the original one of Reference 4. Tests show that for moderate to large problems and for a broad spectrum of $\psi$ values the combination of such a preconditioner with the conjugate gradient method results in an improvement on the total computer time, compared to standard Choleski factorization, while the storage demands could be substantially lower.

The second incomplete Choleski factorization proposed, based on rejection by position, is robust and efficient in contrast to the previously reported incomplete factorizations by position which at least for the structural analysis problems considered may become either unstable or slow convergent. The advantage of this factorization is that the additional storage is prescribed beforehand and thus a possible repeated factorization of the coefficient matrix, if $\psi$ is not properly chosen, is avoided. The case in which the same sparsity pattern of the coefficient matrix is enforced in the preconditioning matrix is very advantageous since practically no additional storage is required nor extra addressings are necessary for storing $L$. On the other hand, the reported computing times are very competitive.

Finally of the two versions of the incomplete factorizations reported in Reference 4 the theoretical one is more robust and should therefore be used.

REFERENCES