A Comparison of Some Preconditioning Techniques for General Sparse Matrices

Benzi, M.
1995

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A Comparison of Some Preconditioning Techniques for General Sparse Matrices[1]

Michele Benzi   Miroslav Tůma

Technical report No. 622

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 66414244   fax: (+422) 8585789
e-mail: uivt@uivt.cas.cz

[1]AMS subject classifications. 65F05, 65F10, 65F25, 65F35, 65F50

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A Comparison of Some Preconditioning Techniques for General Sparse Matrices[1]

Michele Benzi[2] Miroslav Tůma[3]

Technical report No. 622

Abstract

We consider the solution of general sparse systems of linear equations $Ax = b$ by means of the preconditioned conjugate gradient method (PCGNR) applied to the normal equations $A^T Ax = A^T b$. It is known that approaches based on the normal equations can be quite effective for solving problems which are unsymmetric and strongly indefinite. Also, the PCGNR method is an attractive technique for solving large, sparse linear least squares problems.

We obtain robust algebraic preconditioners for the normal equations by means of incomplete orthogonalization schemes applied to A. We focus on schemes based on the Gram-Schmidt process, combined with various sparsity-preserving and stabilization strategies. In addition, we present new methods based on approximating the inverse of the Cholesky factor of $A^T A$ directly. The results of numerical experiments are discussed.

Keywords

Sparse systems of linear equations, sparse matrices, normal equations, preconditioned conjugate gradient, least squares problems, incomplete orthogonal decompositions, approximate inverse preconditioning.

---

[1]AMS subject classifications. 65F05, 65F10, 65F25, 65F35, 65F50

[2]Dipartimento di Matematica, Università degli Studi di Bologna, 40127 Bologna, Italy (e-mail: benzi@dm.unibo.it).

[3]Czech Academy of Sciences, Institute of Computer Science, Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic (e-mail: tuma@uivt.cas.cz)

1 Introduction

In this paper we consider the solution of systems of linear equations

$$Ax = b \tag{1.1}$$

where A is a general sparse matrix, which may be rectangular. For simplicity, we shall assume that A has full column rank. When A is not square, we are interested in solving (1.1) in the least squares sense. An important technique for solving problem (1.1) is the preconditioned conjugate gradient method applied to the normal equations

$$A^T Ax = A^T b \tag{1.2}$$

(hereafter referred to as PCGNR). For a long time, researchers in numerical linear algebra have been skeptical about the normal equations, due to the fact that the spectral condition number of $A^T A$ is the square of that of the original matrix A. Also, explicitly computing $A^T A$ usually entails loss of sparsity and may lead to a severe loss of information (due to cancellation).

In recent times, however, several authors have demonstrated that the preconditioned conjugate gradient method applied to (1.2) can be an effective way of solving problem (1.1), provided that a good preconditioner is available. Fortunately, it is possible to construct a preconditioner for $A^T A$ directly from the original matrix A. Because PCGNR only needs the coefficient matrix of (1.2) in the form of matrix-vector multiplications, there is no need to form the product $A^T A$ explicitly. Numerical experiments have pointed to the fact that PCGNR often exhibits a remarkable degree of robustness, making it a viable alternative to nonsymmetric iterative solvers directly applied to problem (1.1). In particular, it has been shown that approaches based on the normal equations can be quite effective for solving problems which are strongly indefinite (see [10]). Furthermore, the PCGNR method is an attractive technique for solving large, sparse linear least squares problems [7, 11].

We are interested in the construction and testing of robust algebraic preconditioners for the normal equations. Several authors have studied approaches based on the following observation: if $A = QR$ is the QR factorization of A, with R upper triangular with positive diagonal entries and Q orthogonal, the Cholesky factorization of $A^T A$ is given by $A^T A = R^T R$, and $R^{-T} A^T A R^{-1} = I$. This suggests that a sparse approximation to R can be used to precondition the CGNR iteration. A sparse approximation to R may be obtained by explicit formation of $B = A^T A$ followed by an incomplete Cholesky factorization of B. As already mentioned, however, there exist more robust procedures which approximate R directly from A [9, 10, 11]. In [11] the emphasis is on incomplete orthogonalizations obtained by means of Givens rotations, whereas in [10] the preconditioner is constructed using a Gram-Schmidt process; both authors use drop tolerances to preserve sparsity.

In this contribution we restrict ourselves to methods based on the (incomplete) Gram-Schmidt process. We experiment with different sparsity-preserving and safeguarding strategies. In addition, we introduce new methods which approximate the inverse matrix R^{-1} directly. Such incomplete inverse factorization techniques allow the construction of sparse approximate inverses for $A^T A$ and are potentially advantageous on parallel architectures because they avoid the need for (highly sequential) back and forward solves, in contrast with standard incomplete factorization preconditioners ([1], Ch. 8; see also [2]).

An issue of paramount importance in incomplete factorization methods is the *existence* of the incomplete factorization itself. It is well known that if $A^T A$ is an H-matrix, then the incomplete Cholesky factorization of $A^T A$ based on a prescribed sparsity pattern always produces a non-singular preconditioner (in exact arithmetic). On the other hand, a breakdown could take place during an incomplete QR factorization of A even if $A^T A$ is an H-matrix (see [8], where examples are given for an incomplete QR decomposition based on Givens rotations). In practice, however, incomplete orthogonal factorizations based on drop tolerances tend to be very stable, and this is what makes them attractive. In our implementations we included safeguarding mechanisms intended to prevent breakdown of the incomplete factorizations. Our codes always run to completion and produce a nonsingular approximation to R (or R^{-1}).

Concerning sparsity-preserving strategies, we tried a combination of drop tolerances and reordering. Removing computed quantities whose magnitude falls below a preset positive drop tolerance is usually an effective way to preserve sparsity in the approximation to the triangular factor R, but there are some problems for which it is impossible to have both a sparse incomplete factor *and* rapid convergence of the PCGNR iteration. For such stringent problems it may help to resort to the minimum degree algorithm ([6, 7]). This heuristic only requires information about the nonzero structure of A, and is carried out before any floating point computation. After the (column) reordering of A has been determined, an incomplete orthogonal decomposition based on a drop tolerance is performed. A potential problem with this strategy is the fact that the rate of convergence of PCGNR could be adversely affected by the reordering of the unknowns. The risk of this happening is high if the matrix problem originates from the numerical solution of an elliptic PDE and the preconditioner is based on an incomplete factorization with the sparsity pattern of $A^T A$. In this case, reordering the unknowns (i.e., the grid points) according to the minimum degree algorithm can lead to a severe deterioration of the convergence rate as compared with the original (natural) ordering, see [4]. However, it appears from the experiments in [4] that this deterioration is much less serious if the incomplete factorization is obtained with a drop tolerance. In our experience, the use of minimum degree does cause the convergence of PCGNR to somewhat slow down in many cases, but it is useful in order to obtain a more sparse preconditioner, therefore requiring less work per iteration. This remark applies to the preconditioners based on approximating R as well as to the preconditioners based on approximating R^{-1}.

The rest of this paper is organized as follows: in Sections 2 and 3 we describe the incomplete orthogonal and inverse orthogonal factorization schemes (respectively), and in Section 4 we present the results of numerical experiments on a variety of sparse matrix problems, aimed at assessing the relative effectiveness of the various preconditioners. The results are compared with those obtained with a simple preconditioning technique based on the incomplete Cholesky factorization of $A^T A$. Finally, we draw some conclusions in Section 5.

2 Incomplete Orthogonal Factorization Preconditioners

In this section we consider the problem of computing a sparse approximation to R, the upper triangular factor in the QR decomposition of A (we assume that R has positive diagonal elements). To this end we can choose from a variety of available techniques, since there are many algorithms to compute R and within each algorithm there are several ways of obtaining an incomplete factor. Of course, incompleteness can influence R in the various algorithms and implementations very differently.

We describe three versions of incomplete QR decompositions based on the sparse modified Gram-Schmidt (MGS) process, see [3] (for incomplete orthogonalization via Givens rotations, see [11]).

The implementation of all schemes in this and the next section is based on the use of dynamic data structures similar to those adopted in submatrix formulations of sparse unsymmetric Gaussian elimination (see [11]). The coefficient matrix A is stored in the dynamic data structure by columns (CCS format), which is a natural choice for carrying out the (incomplete) MGS process.

A simple way to compute an incomplete QR decomposition is based on dropping elements of the reduced matrix $A^{(i)}$ obtained at step i of the orthogonalization process. We check for elements whose magnitude falls below a prescribed drop tolerance T_d and remove these from the data structure. To improve stability, we increase the diagonal element $a_{ii}^{(i)}$ if the norm of the $i-th$ column of $A^{(i)}$ is less than the prescribed tolerance T_d, as described in Algorithm 2.1. Here μ is a relaxation parameter which was set to 0.1 in our tests. With this safeguarding, the algorithm cannot break down.

The second algorithm in this Section (Algorithm 2.2) is the one described in [9]. Instead of removing "small" elements of updated A it removes "small" nondiagonal elements of the upper triangular factor.

It can be easily shown that Algorithm 2.2 cannot break down —see [9].

Finally, a third algorithm can be obtained by combining the dropping strategies of Algorithms 2.1 and 2.2: small elements are removed both from the factor R and from the set of vectors that remain to be (approximately) orthogonalized.

Algorithm 2.1 Incomplete QR decomposition by diagonally safeguarded MGS

```
set A_1 = (a_1^(1), ..., a_n^(1)) = A ∈ R^{m,n}
for i = 1, ..., n
  if ||a_i^(i)|| < T_d   then
     set a_ii^(i) = a_ii^(i) + μ||(a_ii^(1), ..., a_mi^(1))^T||
  end if
  set r_ii = ||a_i^(i)||
  set q_i = a_i^(i) / r_ii
  for j = i + 1, ..., n
     set α = q_i^T a_j^(i)
     if α ≠ 0 then
        for l = i, ..., m
           if a_lj^(i) ≠ 0 ∨ q_li ≠ 0 then
              set β = a_lj^(i) − α q_li
              if |β| < T_d then
                 a_lj^(i+1) = 0
              else
                 a_lj^(i+1) = β
              end if
           else
              set a_{l,j}^(i+1) = a_{l,j}^(i)
           end if
        end l
     else
        set a_j^(i+1) = a_j^(i)
     end if
  end j
end i
```

Algorithm 2.2 Incomplete QR decomposition by MGS with incomplete updates of R

```
set A_1 = (a_1^(1), ..., a_n^(1)) = A ∈ R^{m,n}
for i = 1, ..., n
  set r_ii = ||a_i^(i)||
  set q_i = a_i^(i) / r_ii
  for j = i + 1, ..., n
     set α = q_i^T a_j^(i)
     if |α| < T_d then
        r_ij = 0
        a_j^(i+1) = a_j^(i)
     else
        r_ij = α
        a_j^(i+1) = a_j^(i) − α q_i
     end if
  end j
end i
```

3 Preconditioning by Incomplete Inverse QR Factorization

In this section we describe some procedures that directly approximate the upper triangular matrix Y such that $(A^TA)^{-1} = Y^TY$. In the language of Statistics, we compute an incomplete *covariance factorization*. This can be computed directly from A applying a procedure which orthogonalizes the columns as in the MGS algorithm and computes Y throughout these steps.

When applied without dropping, this decomposition computes the matrix Y which is equal (up to a diagonal matrix factor) to the transposed inverse of the Cholesky factor of A^TA, which is usually a dense triangular matrix. A sparse approximation may be obtained by removing suitably small fill-in occurring in the course of the computation, or else by a drop-by-position strategy. Here we use a drop tolerance T_d, as in the previous section.

The algorithms to compute an incomplete Y (that is, an approximate inverse of R in the QR factorization of A) update the values of Y in such a way that dynamic data structures must be used. We store Y in this data structure by rows. Elements in these rows are kept in partial order throughout the algorithm. Pointers to elements of the actual column of Y or behind it are kept and updated by a mechanism similar to the one adopted in the numerical factorization procedure in SPARSPAK (see [6]).

The computation of the preconditioner is based on the matrix decomposition $ADY' = Q$, where Q is a matrix with orthonormal columns, D is a positive diagonal matrix, Y' is unit upper triangular and $Y = DY'$ is the inverse of the upper triangular factor in the QR decomposition of A.

We give first the pseudocode for the algorithm with incomplete updates of Y based on the complete sparse MGS and then for the algorithm with complete updates of Y.

Algorithm 3.1 $ADY' = Q$ **decomposition with incomplete updates of** Y'

$set\ A_1 = (a_1^{(1)}, \dots, a_n^{(1)}) = A \in R^{m,n}$
$for\ i = 1, \dots, n$
$\quad set\ d_{ii} = \frac{1}{||a_i^{(i)}||}$
$\quad set\ q_i = a_i^{(i)} d_{ii}$
$\quad for\ j = i+1, \dots, n$
$\quad\quad set\ \alpha = q_i^T a_j^{(i)}$
$\quad\quad if\ |\alpha| \geq T_d\ then$
$\quad\quad\quad a_j^{(i+1)} = a_j^{(i)} - \alpha q_i$
$\quad\quad\quad for\ k = 1, \dots, i$
$\quad\quad\quad\quad if\ y'_{kj} \neq 0 \vee y'_{ki} \neq 0\ then$
$\quad\quad\quad\quad\quad y'_{kj} = y'_{kj} - y'_{ki}\alpha$
$\quad\quad\quad\quad end\ if$
$\quad\quad\quad end\ k$
$\quad\quad end\ if$
$\quad end\ j$
$\quad for\ j = 1, \dots, i-1$
$\quad\quad if\ y'_{ji} \neq 0\ then$
$\quad\quad\quad y'_{ji} = d_{ii} y'_{ji}$
$\quad\quad end\ if$
$\quad end\ j$
$end\ i$

Algorithm 3.2 Incomplete $ADY' = Q$ decomposition diagonally safeguarded

$set\ A_1 = (a_1^{(1)}, \ldots, a_n^{(1)}) = \begin{pmatrix} A \\ 0 \end{pmatrix} \in R^{m,n}$
$for\ i = 1, \ldots, n$
  $if\ ||a_i^{(i)}|| < T_d\ \ then$
    $set\ a_{ii}^{(i)} = a_{ii}^{(i)} + \mu||(a_{ii}^{(1)}, \ldots, a_{mi}^{(1)})^T||$
  $end\ if$
  $set\ d_{ii} = \frac{1}{||a_i^{(i)}||}$
  $set\ q_i = a_i^{(i)} d_{ii}$
  $for\ j = i+1, \ldots, n$
    $set\ \alpha = q_i^T a_j^{(i)}$
    $if\ \alpha \neq 0\ then$
      $for\ l = i, \ldots, m$
        $if\ a_{lj}^{(i)} \neq 0 \vee q_{li} \neq 0\ then$
          $set\ \beta = a_{lj}^{(i)} - \alpha q_{li}$
          $if\ |\beta| < T_d\ then$
            $a_{lj}^{(i+1)} = 0$
          $else$
            $a_{lj}^{(i+1)} = \beta$
          $end\ if$
        $else$
          $set\ a_{l,j}^{(i+1)} = a_{l,j}^{(i)}$
        $end\ if$
      $end\ l$
    $else$
      $set\ a_j^{(i+1)} = a_j^{(i)}$
    $end\ if$
    $for\ k = 1, \ldots, i$
      $if\ y'_{kj} \neq 0 \vee y'_{ki} \neq 0\ then$
        $y'_{kj} = y'_{kj} - y'_{ki}\alpha$
      $end\ if$
    $end\ k$
  $end\ j$
  $for\ j = 1, \ldots, i-1$
    $y'_{ji} = d_{ii} y'_{ji}$
  $end\ j$
$end\ i$

Finally, a third incomplete inverse factorization scheme may be obtained combining the drop strategies adopted in Algorithm 3.1 and Algorithm 3.2.

4 Experimental Results

In this Section we present the results of experiments with the preconditioned CGNR schemes. Most test matrices are from the Harwell-Boeing collection [5]. Matrices WATSONx are from Y. Saad's collection of sparse problems. The test matrices used are representative of problems arising in a variety of applications, such as chemical kinetics, computer system simulation, fluid flow, chemical engineering and others.

The goal of our experiments is to explore the relation between the amount of fill-in allowed in the incomplete factor (or inverse factor) and the rate of convergence of PCGNR. Furthermore, we want to compare the approximate inverse preconditioners described in Section 3 with the more standard implicit preconditioners of Section 2. For completeness, we have also included the results for an incomplete Cholesky factorization (with a drop tolerance) of A^TA. This is the only preconditioner which does not require a dynamic data structure.

Concerning the differences in the computation of the preconditioners themselves, it should be mentioned that in most cases (though not always) it is more expensive to construct an explicit preconditioner than an implicit one. This difference, however, becomes negligible if many linear systems with the same coefficient matrix (or a slightly modified one) and different right-hand sides have to be solved, since in this case the cost of the iterative part dominates the cost of the overall computation. Furthermore, on parallel architectures the approximate inverse preconditioners can take advantage of explicitness.

We present the results of experiments with variants of the incomplete MGS (IMGS) and incomplete inverse MGS (IIMGS) orthogonalization processes, and with the incomplete Cholesky (IC) preconditioner. The results are given in Table 5.1. In all cases we adopted the column ordering of A induced by the minimum degree ordering on the structure of A^TA, for the reasons discussed in the Introduction. The first column in the table contains the matrix name, the number N of unknowns, and the number $NADJ$ of nonzeros in the triangular part of A^TA (after reordering). In case of rectangular matrices, we do not show the number m of matrix rows in the table, since our incomplete algorithms are only affected by the structure of A^TA.

For both groups of Gram-Schmidt algorithms we tried different options for dropping. IMGS1 corresponds to Algorithm 2.1, IMGS3 corresponds to Algorithm 2.2. and IMGS2 combines removing elements from the factor R and from the set of vectors to be approximately orthogonalized in these algorithms. It uses the same drop tolerance for both types of dropping. IIMGS1 is an implementation of Algorithm 3.2, IIMGS3 corresponds to Algorithm 3.1 and IIMGS2 combines both types of dropping.

All the computations were performed in double precision on a SGI Crimson computer. Convergence of the PCGNR iteration was considered achieved when the euclidean norm of the residual was less than 10^{-9}; for all matrices we allowed a maximum number of iterations equal to the number N of unknowns. Notice that without any preconditioning, CGNR fails to converge in N or less iterations for nearly all test problems.

The key role in the comparison is played by the size of fill-in in the approximate factor, which is controlled by the value of the drop tolerance T_d. Note that different matrices and different preconditioners often require totally different values of T_d and it is therefore very difficult to have a good guess for the drop tolerance itself. The only general principle is that severely ill-conditioned problems force the use of small drop tolerances (thus causing high fill-in), or else the iteration will converge very slowly. See [11] for a discussion of this important issue.

For each preconditioning strategy we report the size of fill-in (FILL) and the corresponding number of PCGNR iterations (NIT) for two different values of the drop tolerance. The first (and smaller) one is a value for which the amount of fill in the incomplete factor becomes sensibly smaller than the fill in the complete factor and it still takes only a few PCGNR iterations to fulfill the stopping criterion. The second, larger value corresponds to the point after which the number of iterative steps becomes prohibitive. Notice that in some cases there is not much difference between the two results. Also, it may well happen that a larger value of T_d produces more fill-in than a smaller value; nevertheless, the incomplete factor corresponding to the smaller value of T_d is more accurate, in the sense that it yields convergence in less iterations (see the results for IMGS1 and IMGS2 on matrix FS6801).

For some ill-conditioned matrices we report the results for only one value of T_d. This means that larger values of T_d cause the PCGNR iteration to fail to converge in less than the maximum allowed number N of iterations.

For some extremely ill-conditioned matrices (such as FS5413) we observed convergence only for very small values of T_d, which were ineffective at producing incomplete factors with reduced fill-in. Larger values of T_d produced very slow convergence. Therefore, we did not include those results.

MATRIX N/NADJ	IMGS1 FILL/NIT	IMGS2 FILL/NIT	IMGS3 FILL/NIT	IIMGS1 FILL/NIT	IIMGS2 FILL/NIT	IIMGS3 FILL/NIT	IC FILL/NIT
FS5411	3776/6	1983/6	1183/6	5988/6	881/7	1389/6	1033/6
541/7273	2951/13	850/13	663/14	4552/13	642/13	604/13	697/11
FS5412	7853/13	7604/13	9084/12	61594/13	18904/16	18864/15	17324/2
541/7273	4497/34	4479/33	8193/26	7823/34	13472/34	13523/39	–
FS6801	2492/6	2282/7	1900/5	40491/6	11415/7	11275/7	2056/5
680/3429	2647/23	2647/22	1125/25	38135/23	3545/31	4129/34	1702/10
FS6802	2208/3	2060/5	2028/3	54879/5	5694/6	5683/6	1753/65
680/3429	2207/14	1961/14	1017/16	54074/14	1892/9	2929/9	972/41
FS6803	1895/4	1649/4	1751/2	39288/4	4758/4	5350/3	4013/1
680/3429	1894/8	1621/8	1101/14	39371/8	4265/10	4347/12	986/63
FS7601	2169/3	1577/3	1490/3	4204/3	998/3	963/3	995/3
760/13956	2114/9	1079/9	928/12	4037/9	858/9	860/9	865/9
WELL1033	2531/2	2480/2	2501/2	6182/2	6136/2	6136/2	2500/2
320/2147	2425/12	2238/12	2172/10	5583/12	5201/10	5233/10	2310/11
ILLC1033	2555/2	2554/2	2554/2	6205/2	6205/2	6205/2	2552/3
320/2147	2549/10	2519/10	2415/13	6205/10	5981/17	5976/15	2434/20
GRE115	1805/8	1632/6	1812/2	4319/3	4278/3	4226/3	1672/5
115/691	1655/13	1261/13	1160/11	3282/58	3317/9	3318/8	1365/9
GRE185	5632/4	5632/4	5622/4	11422/4	11363/4	11378/4	5629/3
185/2099	5632/16	5468/16	4894/16	11422/16	10468/19	10955/19	5529/13
GRE343	13274/41	9496/41	10565/14	32108/41	26701/51	27169/52	12064/12
343/2719	6015/98	4494/99	5931/38	31584/98	19835/85	21827/98	6001/50
WATSON1	1621/3	1309/3	1075/3	1655/3	1326/5	1444/3	1005/5
58/1711	1282/9	741/9	575/8	1453/9	1212/11	1207/11	473/10
WATSON2	2212/3	1117/3	799/3	2211/3	1736/4	1721/4	1438/4
66/2211	1530/6	640/8	625/7	1698/6	1573/7	1601/6	758/6
WATSON3	6514/8	4944/7	4287/7	7718/7	6737/5	6525/6	5113/6
124/7750	3558/13	2686/13	3176/14	6543/13	5840/11	5809/17	3923/14
STR400	6834/3	6773/3	6108/5	22961/2	22389/2	22288/2	6771/2
363/4363	5178/18	4814/13	3565/14	19659/13	13059/14	13217/14	4926/10
LNS131	208/55	208/55	208/55	270/55	270/55	270/55	808/1
131/1149	208/85	208/85	206/70	–	–	–	–
WEST0132	746/11	746/11	747/8	2720/8	2720/8	2720/8	831/1
132/692	–	–	573/94	2309/62	1757/99	–	–
WEST0156	510/13	512/12	510/12	2190/12	2190/12	2190/12	663/3
156/496	–	–	506/38	1821/45	1803/48	886/89	–
WEST0167	910/8	910/8	910/8	3737/8	3737/8	3737/8	1026/1
167/856	–	–	843/37	3206/15	–	–	–
WEST0381	18552/5	18517/5	18397/5	47052/7	44846/5	44845/5	18591/1
381/6945	17155/79	16927/78	16288/61	46851/79	39692/35	39610/35	–

Table 5.1: Comparison of iteration counts and fill-in in preconditioners in the PCGNR procedure (CGNR preconditioned by incomplete MGS, incomplete inverse MGS and IC algorithms).

5 Conclusions

The first conclusion drawn from the experiments concerns the comparison of methods within each of the groups IMGSx and IIMGSx. Our results show that while the options 2 and 3 give similar results, the option 1 is often worse, in the sense that fill-in in the incomplete factor is higher. This is especially true for the IIMGS class of methods.

Judging on the ground of robustness, amount of fill in the incomplete factor and effectiveness at reducing the number of PCGNR iterations, the overall winner is preconditioner IMGS3.

An observation which applies to all the preconditioning schemes studied in this paper is that it is difficult to predict in advance what kind of fill-in and convergence behavior will correspond to a given value of the drop tolerance T_d. However, matrices from the same type of application (such as the FS matrices, which arise from the solution of systems of ODEs in chemical kinetics studies) tend to have a similar behavior. Hence, it may be worthwhile to spend some time to find a good value of T_d for one of these matrices and then use it also for other matrices in the same class. Again, this stratagem will not work on extremely ill-conditioned problems.

An interesting and perhaps surprising result of the experiments is the existence of some matrices for which the preconditioners based on the incomplete inverse factorization behave very well. In fact, for some of these matrices the size of the incomplete inverse factor Y (in terms of fill-in) can even be less than the size of the incomplete factor R, and yet produce the same or a similar number of PCGNR iterations. For most test matrices, however, the approximate inverse preconditioners require substantially fuller incomplete factors in order to produce convergence rates comparable with those for the implicit preconditioners. In other words, very sparse approximate inverse preconditioners are usually not as effective at reducing the number of PCGNR iterations as the implicit ones. Explicitness comes to a price: whether this price is worth paying, it will depend on the particular problem and computer architecture at hand.

Finally, it should be observed that in many cases the simple approach based on computing an incomplete Cholesky decomposition of A^TA gives very satisfactory results. On the other hand, our results confirm the well-known fact that in practice, the incomplete Cholesky approach is more prone to suffer from instability problems than the approach based on incomplete orthogonalization. In fact, it is precisely this observation which led researchers to consider the use of incomplete orthogonal factorizations in the first place. In can be seen from the last column in Table 1 that the IC algorithm with comparatively large drop tolerances failed to produce a stable preconditioner in six cases. The most robust of all preconditioners, on the basis of our experiments, is IMGS3 (Algorithm 2.2).

Bibliography

[1] O. Axelsson, *Iterative Solution Methods,* Cambridge University Press, Cambridge (1994).

[2] M. Benzi, C. D. Meyer and M. Tůma, A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method, *NCSU Technical Report #1-062194-02,* Raleigh, NC (1994). To appear in *SIAM J. Sci. Comput.*

[3] Å. Björck, Solving linear least squares problems by Gram-Schmidt orthogonalization, *BIT* **7** (1967), 1–21.

[4] I.S. Duff and G.A. Meurant, The effect of ordering on preconditioned conjugate gradients, *BIT* **29** (1989), 635–657.

[5] I.S. Duff, R.G. Grimes and J.G. Lewis, Sparse matrix test problems, *ACM Trans. Math. Software* **15** (1989), pp. 1–14.

[6] A. George and J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems,* Prentice-Hall, Englewood Cliffs, N.J. (1981).

[7] M. T. Heath, Numerical Methods for Large Sparse Linear Least Squares Problems, *SIAM J. Sci. Stat. Comput.* **5** (1984), 497–513.

[8] D. James, Ph.D. Dissertation, Department of Mathematics, North Carolina State University, Raleigh, NC (1990).

[9] A. Jennings and M. A. Ajiz, Incomplete Methods for Solving $A^T Ax = b$, *SIAM J. Sci. Stat. Comput.* **5** (1984), 978–987.

[10] Y. Saad, Preconditioning Techniques for Nonsymmetric and Indefinite Linear Systems, *J. Comput. Appl. Math.* **24** (1988), 89–105.

[11] Z. Zlatev, *Computational Methods for General Sparse Matrices*, Kluwer Academic Publishers, Dordrecht etc. (1991).