APPROXIMATE INVERSE PRECONDITIONING FOR THE CONJUGATE GRADIENT METHOD ON A VECTOR COMPUTER

Michele Benzi
CERFACS, Toulouse, France (benzi@cerfacs.fr)

Miroslav Tůma
Institute of Computer Science, Czech Academy of Sciences, Prague, CZ (tuma@uivt.cas.cz)

1. Introduction. In this note we consider the solution of large, sparse linear systems of the form

$$Ax = b \tag{1}$$

where A is a $n \times n$, symmetric and positive definite matrix, by the preconditioned conjugate gradient method (PCG) (see, e.g., [4]). It is well-known that one of the most powerful and widely used preconditioning techniques, the incomplete Cholesky factorization (IC), is difficult to implement on vector and parallel machines, due to the recursive nature of the triangular solves involved. This observation has motivated, in recent years, an intense research activity in the area of *approximate inverse* preconditioning. With this approach, a sparse matrix $G \approx A^{-1}$ is explicitly constructed and used as a preconditioner for (1). This is sometimes referred to as *explicit* preconditioning, in contrast with *implicit* techniques, like incomplete Cholesky, which require the solution of (triangular) linear systems at each PCG step. Applying an explicit preconditioner G within a step of the conjugate gradient method reduces to computing a matrix-vector product with G, an operation which offers considerably more scope for vectorization and parallelism than the triangular solves. For efficiency reasons, the approximate inverse G should be easily computed and applied. Typically, the cost of applying the preconditioner at each iteration of the conjugate gradient method should be of the same order as the cost of a matrix-vector multiply involving A. For a sparse A, this implies that G should also be sparse with a density of nonzeros roughly the same as that of A.

Because the inverse of a sparse, irreducible matrix is structurally full, it is not obvious that a sparse approximation to A^{-1} can be found. However, many sparse matrices which arise in practical problems have an inverse whose entries decay rapidly away from the main diagonal (see [3]), making the approximation of A^{-1} with a sparse matrix feasible. Nevertheless, explicit preconditioners are sometimes less effective than implicit ones at reducing the number of iterations, in the sense that there are problems for which they require a higher number of nonzeros to achieve the same rate of convergence insured by implicit preconditioners.

Furthermore, explicit preconditioners are usually more expensive to compute than implicit ones, although this difference may become negligible if several linear systems with the same coefficient matrix and different right-hand sides have to be solved. In this case the time for computing the preconditioner is often only a fraction of the time required for the overall computation. Clearly, the main question to be answered is whether the advantages of explicitness are sufficient to compensate the above-mentioned shortcomings. The answer is problem- and architecture- dependent. In this note we show that we can answer the question in the affirmative for a meaningful test problem and architecture. Here we consider only symmetric and positive definite matrices, but approximate inverse techniques can be used for nonsymmetric systems as well. We refer the reader to [1] and [2] for detailed discussions of approximate inverse preconditioning and extensive references.

2. Computing a sparse approximate inverse. There exist several alternative methods of computing a sparse approximate inverse of a given matrix A, see [2]. In this note we restrict ourselves to factorized sparse approximate inverses obtained by the incomplete A-conjugate Gram-Schmidt process. A detailed study of this method, including theoretical properties and the results of numerical experiments in a scalar computing environment, can be found in [1] and [2]. The construction of the preconditioner is based on an algorithm which computes a set of vectors $\{z_i\}_{i=1}^n$ which are A-conjugate, i.e. such that $z_i^T A z_j = 0$ if and only if $i \neq j$. If

$$Z = [z_1,\ z_2, \ldots, z_n]$$

is the matrix whose ith column is z_i, we have

$$Z^T A Z = D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$$

where $p_i = z_i^T A z_i > 0$. It follows that

$$A^{-1} = Z D^{-1} Z^T \tag{2}$$

and a factorization of A^{-1} is obtained. Hence, the inverse of A is known if a basis of A-conjugate vectors is known. Note that there are infinitely many such bases. A matrix Z whose columns are A-conjugate can be explicitly computed by means of a conjugation process applied to the standard basis vectors $e_1, \ldots, e_n$. This is simply the Gram-Schmidt process with the inner product defined by $\langle x, y\rangle_A := x^T A y$. The resulting matrix Z is unit upper triangular; indeed, $Z = L^{-T}$ where $A = LDL^T$ is the root-free Cholesky factorization of A. Hence, the process produces an inverse triangular decomposition of A or, equivalently, a triangular decomposition (of the $L^T DL$ type) of A^{-1}. The p_i's returned by the algorithm are the pivots in the LDL^T factorization of A. Denoting by a_i^T the ith row of A, the basic A-conjugation procedure can be written as follows.

THE A-CONJUGATION ALGORITHM

(1) Let $z_i^{(0)} = e_i \quad (1 \le i \le n)$

(2) **for** $i = 1, 2, \ldots, n$

for $j = i, i+1, \ldots, n$

$p_j^{(i-1)} := a_i^T z_j^{(i-1)}$

end

if $i = n$ **go to** (3)

for $j = i+1, \ldots, n$

$$z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}}\right) z_i^{(i-1)}$$

end

end

(3) Let $z_i := z_i^{(i-1)}$ and $p_i := p_i^{(i-1)}, \quad$ for $1 \le i \le n$.

Return $Z = [z_1, \; z_2, \ldots, z_n]$ and $D = \mathrm{diag}(p_1, p_2, \ldots, p_n)$.

Once Z and D are available, the solution of (1) can be computed as

$$x^* = A^{-1}b = ZD^{-1}Z^Tb = \sum_{i=1}^{n} \left(\frac{z_i^T b}{p_i}\right) z_i.$$

For a dense matrix this method requires about twice as much work as Cholesky. For a sparse matrix the cost can be substantially reduced, but the method is still impractical because the resulting Z tends to be dense. On the other hand, sparsity in Z can be preserved by removing "small" fill in the z-vectors. To this end, we use a drop tolerance $T > 0$ to decide which fill-in is to be discarded: new nonzeros in the z-vectors are dropped if smaller than T in absolute value. Concerning the stability of the incomplete process, it was shown in [1] that the incomplete A-conjugation process (incomplete inverse factorization) cannot break down, in exact arithmetic, if A is an H-matrix. For more general matrices it is necessary to safeguard the computation in order to avoid breakdowns. This can be done by shifting small pivots —we refer to [1] for details. The incomplete A-conjugation algorithm computes a sparse unit upper triangular matrix $\bar{Z} \approx Z$, and a nonsingular (positive) diagonal matrix $\bar{D} \approx D$ such that

$$G := \bar{Z}\bar{D}^{-1}\bar{Z}^T \approx A^{-1}$$

is a factorized sparse approximate inverse of A which can be used as an explicit preconditioner for the conjugate gradient method for the solution of (1). Applying the preconditioner requires two matrix-vector products with $\bar{Z}$ and its transpose (the diagonal factor $\bar{D}$ can be absorbed into

$\bar{Z}$). The factorized form of the approximate inverse has several advantages. Clearly, it is easy to check that the preconditioner is a symmetric and positive definite matrix, as needed in the PCG process, since this only requires that the diagonal entries in $\bar{D}$ be positive. Furthermore, factorized forms of sparse matrices contain more information for the same storage than if a single product was stored.

3. Numerical results. In this section we present the results of numerical experiments carried out on a Cray C-90 vector processor located at Meteo France, in Toulouse. The test problem considered is a matrix of order $N = 32,010$ with $NNZ = 159,466$ nonzero entries, arising from a finite difference discretization of Poisson's equation with Dirichlet boundary conditions on a highly non-uniform mesh in 2D. The problem is extracted from an explicitly time-dependent simulation of rare-gas atoms exposed to intense laser pulses. It is worth emphasizing that this particular application requires the solution of thousands of linear systems with the same coefficient matrix and different right-hand sides. In this situation, preconditioners which are expensive to compute but are highly efficient to apply are well worth considering, since the cost for setting up the preconditioner can be amortized over repeated solutions. Numerical experiments with matrices from other application areas (finite element modelling of structures, fluid flow problems, power system admittance matrices) are reported in [1], where it was found that the approximate inverse preconditioner produces convergence rates similar to those obtained with incomplete Cholesky preconditioning.

In order to take advantage of vectorization, all matrix-vector products have been implemented with the *jagged diagonal scheme* for storing general sparse matrices, see [6], [5]. In addition, no advantage was taken of symmetry to save storage for A, as this is known to cause a degradation of the performance on supercomputers, see [5]. For the dot products and the vector updates (saxpys) in the CG algorithm, standard BLAS were used. We concentrated on efficient implementation of the PCG process because the construction of the preconditioners offers relatively little opportunity for optimization.

The coefficient matrix was rescaled by dividing its elements by the absolute value of the largest nonzero entry. With this scaling, the drop tolerance can be taken to be a number between 0 and 1. No other scaling was used. The right-hand side was computed from the solution vector x^* of all ones. The initial guess for the PCG iteration was the zero vector. The stopping criterion used was $||r_k||_2 < 10^{-9}$, where r_k is the (unpreconditioned) updated residual after k iterations. Codes were written in standard Fortran 77 and compiled with the optimization option *-Zv*.

The results of the experiments are reported in Table 1. We included in the comparison the standard CG algorithm (without preconditioning), CG with Jacobi (or diagonal) preconditioning and CG preconditioned with a no-fill incomplete Cholesky factorization (denoted by IC(0)). Furthermore, we compared drop tolerance-based incomplete Cholesky and approximate inverse preconditioners, denoted by IC and AINV respectively, for different choices of the drop tolerance.

This was adjusted as to produce preconditioners with a similar relative density of nonzeros, which is defined as the number of nonzeros in the preconditioner divided by the number of nonzeros in A (denoted in the table by $Fill$). The table gives the time (in seconds) required for setting up the preconditioner (under $P - time$), the number of PCG steps (under $PCG - it$), and the time for the iteration (under $PCG - time$).

PRECOND	Fill	P-time	PCG-it	PCG-time
None	–	–	3359	9.56
Diagonal	–	0.09	804	2.55
IC(0)	1.	0.31	209	14.94
IC	0.48	0.31	411	19.94
AINV	0.48	3.37	851	3.82
IC	0.68	0.34	366	21.35
AINV	0.65	3.43	603	2.92
IC	0.80	0.49	313	19.52
AINV	0.81	3.53	527	3.08
IC	1.10	0.55	183	13.21
AINV	1.20	3.67	422	2.81
IC	1.60	0.66	122	8.95
AINV	1.60	3.78	326	2.67
IC	1.80	0.73	102	7.55
AINV	1.80	3.83	305	2.68
IC	2.10	0.87	87	6.53
AINV	2.10	3.95	296	2.93
IC	2.70	1.16	68	5.29
AINV	2.70	4.31	259	3.14
IC	3.10	1.39	65	5.10
AINV	3.10	4.47	243	4.37

Table 1: Test results for the conjugate gradient method with various preconditioners (Fill = relative density of preconditioner, P-time = time for computing the preconditioner, PCG-it = number of PCG iterations, PCG-time = time for PCG iteration).

We first observe that for this particular problem diagonal preconditioning performs extremely well, and it should be used in practice. However, for most problems of interest, especially strongly unsymmetric and indefinite ones, diagonal preconditioning is usually not sufficient to achieve convergence. In these cases more powerful methods must be used, like ILU and approximate inverse techniques (see [2]). It is therefore important to compare the behavior of the approximate inverse preconditioner with the incomplete Cholesky one.

It can be seen from the experiments that the AINV preconditioner gives faster execution for all values of the relative density considered, in spite of a much lower rate of convergence.

For a density of about 1.5, it is virtually as good as diagonal preconditioning. This confirms that AINV has excellent vectorization properties, unlike IC. We note that as the relative density approaches 3, the difference between IC and AINV becomes smaller, thanks to the fact that IC exhibits a much better convergence rate. The number of iterations continues to drop rapidly for PCG preconditioned with IC, whereas the improvement is much smaller for AINV. This behavior is typical of sparse approximate inverse preconditioners. Once a certain threshold is reached (density ≈ 2), the execution time for AINV starts actually increasing because the reduction in the number of iterations is not sufficient to compensate for the increased work to be done at each PCG step. In fairness to AINV, it must be mentioned that the difference in the rate of convergence obtained with IC and AINV preconditioners of equal relative density is usually not as pronounced as in this case (cf. the results in [1]). This is a hard problem for the sparse approximate inverse preconditioner.

It is clear that the IC preconditioner can be computed faster than the AINV preconditioner. However, this is not important for this particular application, where a long sequence of linear systems with the same matrix and different right-hand sides have to be solved. It is also worth mentioning that the approximate inverse considered in this paper is substantially less expensive to compute than other explicit preconditioners found in the literature [2].

In this note we have only considered the effect of vectorization in a uniprocessor implementation. In a forthcoming paper, we will present the results of a multiprocessor implementation. Parallelization of the sparse matrix-vector multiplies is relatively simple, whereas it is difficult to exploit concurrency in the triangular solves required by IC. We will compare AINV with IC implemented with level scheduling (see [6]) and also with other parallel preconditioners.

Acknowledgments. The work of the first author was performed in the framework of an HCM Project on Iterative Methods in Scientific Computing (Contract No. CHRX-CT9300420). The second author was supported in part by NSF grant INT-9218024 and by GA CR grant 205/96/0921. We would like to thank Dr. Carsten Ullrich of CERFACS for providing us with the test problem.

REFERENCES

[1] M. Benzi, C. D. Meyer and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17:1135–1149, 1996.

[2] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. To appear in *SIAM J. Sci. Comput.*

[3] S. Demko, W. F. Moss and P. W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 43:491–499, 1984.

[4] J. J. Dongarra, I. S. Duff, D. C. Sorensen and H. A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM, Philadelphia, 1991.

[5] M. A. Heroux, P. Vu and C. Yang. A parallel preconditioned conjugate gradient package for solving sparse linear systems on a Cray Y-MP. *Appl. Num. Math.*, 8:93–115, 1991.

[6] Y. Saad. Krylov subspace methods on supercomputers. *SIAM J. Sci. Stat. Comput.*, 10:1200–1232, 1989.