Abstract

The preconditioned conjugate gradient (PCG) method is an effective means for solving systems of linear equations where the coefficient matrix is symmetric and positive definite. The incomplete LDL^t factorizations are a widely used class of preconditionings, including the SSOR, Dupont-Kendall-Rachford, Generalized SSOR, ICCG(0), and MICCG(0) preconditionings. The efficient implementation of PCG with a preconditioning from this class is discussed.

Efficient Implementation of a Class of Preconditioned Conjugate Methods

Stanley C. Eisenstat

Research Report #185

August 1980

This research was supported in part by ONR Grant N00014-76-C-0277.

1. Introduction

Consider the system of N linear equations

(1) $A\,x = b$,

where the coefficient matrix A is symmetric and positive definite. When A is large and sparse, the preconditioned conjugate gradient (PCG) method is an effective means for solving (1) [2, 4, 5, 9, 13]. Given an initial guess x_0, we generate a sequence $\{x_k\}$ of approximations to the solution x as follows:

(2a) $p_0 = r_0 = b - Ax_0$

(2b) Solve $Mr'_0 = r_0$

FOR k = 0 STEP 1 UNTIL Convergence DO

(2c) $a_k = (r_k, r'_k) \,/\, (p_k, Ap_k)$

(2d) $x_{k+1} = x_k + a_k p_k$

(2e) $r_{k+1} = r_k - a_k A p_k$

(2f) Solve $Mr'_{k+1} = r_{k+1}$

(2g) $b_k = (r_{k+1}, r'_{k+1}) \,/\, (r_k, r'_k)$

(2h) $p_{k+1} = r'_{k+1} + b_k p_k$

The effect of the preconditioning matrix M is to increase the rate of convergence of the basic conjugate gradient method of Hestenes and Stiefel [11]. The number of multiply-adds per iteration is just 5N, plus the number required to form Ap_k, plus the number required to solve $Mr'_k = r_k$.

One widely used class of preconditionings are the incomplete LDL^t factorizations

$$(3) \qquad M = (\tilde{D}+L)\,\tilde{D}^{-1}\,(\tilde{D}+L)^t ,$$

where $A \equiv L+D+L^t$, L is strictly lower triangular, and D and $\tilde{D}$ are positive diagonal. This class includes the SSOR [9], Dupont-Kendall-Rachford [7], Generalized SSOR [1], ICCG(0) [13], and MICCG(0) [10] preconditionings. Letting NZ(A) denote the number of nonzero entries in the matrix A, a straight-forward implementation of PCG with a preconditioning from this class[1] would require 6N+2NZ(A) multiply-adds per iteration.[2]

In this brief note, we show how to reduce the work to 8N+NZ(A) multiply-adds, asymptotically half as many as the straight-forward implementation.[3] We give details in Section 2, and consider some generalizations in Section 3.

2. Implementation

The linear system (1) can be restated in the form

---

[1] Writing M as $(\tilde{D}+L)(I+\tilde{D}^{-1}L^t)$, we solve $Mr_k' = r_k$ by solving the triangular systems $(\tilde{D}+L)t_k = r_k$, $(I+\tilde{D}^{-1}L^t)r_k' = t_k$.

[2] 2N (respectively, N) multiply-adds can be saved by symmetrically scaling the problem to make $\tilde{D} = I$ (respectively, $D = I$).

[3] A similar speedup for pairs of linear iterative methods is given in [6].

(4) $$[(\tilde{D}+L)^{-1} A (\tilde{D}+L)^{-t}] [(\tilde{D}+L)^t x] = [(\tilde{D}+L)^{-1} b]$$

or

(5) $$\hat{A} \hat{x} = \hat{b} .$$

But applying PCG to (1) with $M = (\tilde{D}+L)\tilde{D}^{-1}(\tilde{D}+L^t)$ is equivalent to applying PCG to (5) with $\hat{M} = \tilde{D}^{-1}$ and setting $x = (\tilde{D}+L)^{-t}\hat{x}$.[4] If we update x instead of $\hat{x}$ at each iteration, algorithm (2) becomes:

(6a) $\hat{p}_0 = \hat{r}_0 = \hat{b} - \hat{A}x_0$

(6b) Compute $\hat{r}'_0 = \tilde{D}\hat{r}_0$

FOR k = 0 STEP 1 UNTIL Convergence DO

(6c) $\hat{a}_k = (\hat{r}_k,\hat{r}'_k) / (\hat{p}_k,\hat{A}\hat{p}_k)$

(6d) $x_{k+1} = x_k + \hat{a}_k(\tilde{D}+L)^{-t}\hat{p}_k$

(6e) $\hat{r}_{k+1} = \hat{r}_k - \hat{a}_k\hat{A}\hat{p}_k$

(6g) Compute $\hat{r}'_{k+1} = \tilde{D}\hat{r}_{k+1}$

---

[4] Both are equivalent to applying the basic conjugate gradient method to the preconditioned system

$$\bar{A}\bar{x} \equiv [\tilde{D}^{1/2}(\tilde{D}+L)^{-1} A (\tilde{D}+L)^{-t}\tilde{D}^{1/2}] [\tilde{D}^{-1/2}(\tilde{D}+L)^t x] = [\tilde{D}^{1/2}(\tilde{D}+L)^{-1} b] \equiv \bar{b}$$

(see [4], pp. 58-59).

(6g) $\hat{b}_k = (\hat{r}_{k+1},\hat{r}'_{k+1}) \;/\; (\hat{r}_k,\hat{r}'_k)$

(6h) $\hat{p}_{k+1} = \hat{r}'_{k+1} + \hat{b}_k\hat{p}_k$

$\hat{A}\hat{p}_k$ can be computed efficiently by taking advantage of the following identity:

(7) $$\hat{A}\hat{p}_k = (\tilde{D}+L)^{-1}\,[(\tilde{D}+L) + (\tilde{D}+L)^t - (2\tilde{D}-D)]\,(\tilde{D}+L)^{-t}\,\hat{p}_k$$

$$= (\tilde{D}+L)^{-t}\hat{p}_k \;+\; (\tilde{D}+L)^{-1}\,[\hat{p}_k - K(\tilde{D}+L)^{-t}\hat{p}_k]\ ,$$

where $K \equiv 2\tilde{D}-D$. Thus

(8a) $\hat{t}_k = (\tilde{D}+L)^{-t}\,\hat{p}_k$

(8b) $\hat{A}\hat{p}_k = \hat{t}_k + (\tilde{D}+L)^{-1}\,(\hat{p}_k - K\hat{t}_k)\ ,$

which requires 2N+NZ(A) multiply-adds. $\hat{t}_k$ can also be used to update x_k in (6d), so that the total cost for each PCG iteration is just 8N+NZ(A) multiply-adds,[5] versus 6N+2NZ(A) for the straight-forward implementation.

3. Generalizations

The approach presented in Section 2 extends immediately to preconditionings of the form

---

[5] Again, 3N multiply-adds can be saved by symmetrically scaling the problem so that $\tilde{D} = I$.

(9) $M = (\tilde{D}+L)\ \tilde{S}^{-1}\ (\tilde{D}+L)^t$,

where $\tilde{S}$ is positive diagonal. Moreover, if we take $K \equiv \tilde{D}+\tilde{D}^t-D$ in (7) and (8), then $\tilde{D}$ need not be diagonal or even symmetric. In this case, $\tilde{D}$ would reflect changes to both the diagonal and off-diagonal entries of A in generating an incomplete factorization. If we assume that only the nonzero entries of A are changed, i.e., that $(K)_{ij}$ is nonzero only if $(A)_{ij}$ is nonzero, then the operation count is 7N+NZ(A)+NZ(K).

Another application is to preconditioning nonsymmetric systems. Let

(10) $M = (\tilde{D}+L)\ \tilde{S}^{-1}\ (\tilde{D}+U)$,

be an incomplete LDU factorization of a nonsymmetric matrix A, where $A \equiv L+D+U$, L (respectively, U) is strictly lower (respectively, upper) triangular, and D and $\tilde{S}$ are diagonal. Then a number of authors have proposed solving the linear system Ax = b by solving the normal equations for one of the preconditioned systems

(11a) $\hat{A}_1\hat{x} \equiv [\tilde{S}\ (\tilde{D}+L)^{-1}\ A\ (\tilde{D}+U)^{-1}]\ [(\tilde{D}+U)\ x] = [\tilde{S}\ (\tilde{D}+L)^{-1}\ b] \equiv \hat{b}$

(see [12]) and

(11b) $\hat{A}_2 x \equiv [(\tilde{D}+U)^{-1}\ \tilde{S}\ (\tilde{D}+L)^{-1}\ A]\ x = [(\tilde{D}+U)^{-1}\ \tilde{S}\ (\tilde{D}+L)^{-1}\ b] \equiv \hat{b}$

(see [14, 3]). $\hat{A}_2\hat{p}$ can be computed as

(12) $\hat{A}_2\hat{p} = (\tilde{D}+U)^{-1}\ \tilde{S}\ [\hat{p} + (\tilde{D}+L)^{-1}(D+U-\tilde{D})\hat{p}]$

in 4N+NZ(L)+2NZ(U) multiply-adds, whereas $\hat{A}_1\hat{p}$ can be computed as

(13a) $\hat{t} = (\tilde{D}+U)^{-1}\hat{p}$

(13b) $\hat{A}_1\hat{p} = \tilde{S}\ [\hat{t} + (\tilde{D}+L)^{-1}\ (\hat{p} - (2\tilde{D}-D)\hat{t})]$

in 4N+NZ(L)+NZ(U) multiply-adds. Thus the first approach would be more efficient per iteration, although more iterations might be required to achieve comparable accuracy [14].[6]

REFERENCES

[1] O. Axelsson.
A generalized SSOR method.
BIT 13:443-467 (1972).

[2] O. Axelsson.
On preconditioning and convergence acceleration in sparse matrix problems.
Technical Report 74-10, CERN, May 1974.

[3] Owe Axelsson.
Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations.
Linear Algebra and Its Applications 29:1-16 (1980).

[4] Rati Chandra.
Conjugate Gradient Methods for Partial Differential Equations.
PhD Thesis, Department of Computer Science, Yale University, 1978.

[5] Paul Concus, Gene H. Golub, and Dianne P. O'Leary.
A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations.
In James R. Bunch and Donald J. Rose, Editors, Sparse Matrix Computations, pp. 309-332. Academic Press, 1976.

---

[6] The same would be true if a Generalized Conjugate Residual method such as Orthomin [15, 8] were used to solve (11a) or (11b).

[6] V. Conrad and Y. Wallach.
Alternating methods for sets of linear equations.
Numerische Mathematik 32:105-108 (1979).

[7] Todd Dupont, Richard P. Kendall, and H. H. Rachford Jr.
An approximate factorization procedure for solving self-adjoint elliptic difference equations.
SIAM Journal on Numerical Analysis 5:559-573 (1968).

[8] S. C. Eisenstat, H. Elman, M. H. Schultz, and A. H. Sherman.
Solving approximations to the convection diffusion equation.
In Proceedings of the Fifth Symposium on Reservoir Simulation, pp. 127-132. Society of Petroleum Engineers of AIME, February 1979.

[9] D. J. Evans.
The analysis and application of sparse matrix algorithms in the finite element method.
In J. R. Whiteman, Editor, The Mathematics of Finite Elements and Applications, pp. 427-447. Academic Press, 1973.

[10] Ivar Gustafsson.
A class of first order factorization methods.
BIT 18:142-156 (1978).

[11] Magnus R. Hestenes and Eduard Stiefel.
Methods of conjugate gradients for solving linear systems.
Journal of Research of the National Bureau of Standards 49:409-436 (1952).

[12] David S. Kershaw.
The incomplete Cholesky-conjugate gradient method for the solution of systems of linear equations.
Journal of Computational Physics 26:43-65 (1978).

[13] J. A. Meijerink and H. A. van der Vorst.
An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix.
Mathematics of Computation 31:148-162 (1977).

[14] M. Petravic and G. Kuo-Petravic.
An ILUCG Algorithm Which Minimizes in the Euclidean Norm.
Journal of Computational Physics 32:263-269 (1979).

[15] P. K. W. Vinsome.
Orthomin, an iterative method for solving sparse sets of simultaneous linear equations.
In Proceedings of the Fourth Symposium on Reservoir Simulation, pp. 150-159. Society of Petroleum Engineers of AIME, February 1976.