A method for constructing diagonally dominant preconditioners based on Jacobi rotations

Article *in* Applied Mathematics and Computation · March 2006

DOI: 10.1016/j.amc.2003.10.070 · Source: DBLP

2 authors:

Jin-Yun Yuan
Universidade Federal do Paraná
112 PUBLICATIONS **825** CITATIONS
SEE PROFILE

Plamen Y. Yalamov
Rousse University
49 PUBLICATIONS **192** CITATIONS
SEE PROFILE

A Method for Constructing Diagonally Dominant Preconditioners based on Jacobi Rotations

Jin Yun Yuan* Plamen Y. Yalamov†

Abstract

A method is presented to make a given matrix strictly diagonally dominant as much as possible based on Jacobi rotations in this paper. The strictly diagonally dominant rows are used to build a preconditioner for some iterative method. Roundoff error analysis of the method is also given. The numerical tests illustrate that the method works very well even for very ill-conditioned linear systems.

AMS subject classification:65F10

Key words:strictly diagonally dominant matrix, preconditioned iterative method, preconditioner, Jacobi rotation.

*Departamento de Matemática, Universidade Federal do Paraná, Centro Politécnico, CP: 19.081, CEP: 81531-990, Curitiba, Brazil, e-mail: jin@mat.ufpr.br. The work was partially supported by Grant 301039/93-8 from CNPq, Brazil.

†Center of Applied Mathematics and Informatics, University of Rousse, 7017 Rousse, Bulgaria, e-mail: yalamov@ami.ru.acad.bg. The work was finished during the visit of this author to the Federal University of Paraná. The research was partially supported by Grant MM-434/94 from the Bulgarian Ministry of Education and Science.

1 Introduction

It is well-known that strictly diagonally dominant matrices are usually well conditioned, and that the solution of linear systems with such matrices is stable (for example, see [4, p. 120]). Many iterative methods, such as the Jacobi method, the Gauss-Seidel and the SOR method, are convergent for diagonally dominant systems. Unfortunately, in practice we often have of course non-diagonally dominant matrices. Hence, iterative methods can perform badly when applied to general matrices.

We call a matrix $A \in \mathcal{R}^{n\times n}$ strictly diagonally dominant, if

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}|, \ i = 1, \ldots, n.$$

For every nonsingular matrix A it follows immediately from the Singular Value Decomposition (SVD) that there exist nonsingular matrices P and Q such that PAQ is strictly diagonally dominant. A different proof of this fact is given in [8]. The author also showed that there exists a nonsingular matrix P such that PA is strictly diagonally dominant. A tridiagonal matrix P is constructed such that PA is strictly diagonally dominant for 3-cyclic matrices as an example in [8].

It is very clear that after finding the preconditioners P and Q such that PAQ is strictly diagonally dominant, we can apply iterative methods for solving

$$PAQy = Pb,$$

and

$$x = Qy$$

instead of solving

$$Ax = b.$$

Therefore, the key problem is to find P and Q cheaply. The simplest way for obtaining such strictly diagonally dominant matrix is the SVD, where $P = U$ and $Q = V^T$. But the price of the SVD is prohibitively high for solving linear systems. Unfortunately, at the moment there is no other cheap way to construct such preconditioners P and Q for general matrices A.

In the present work we consider some cheaper approach for the desired preconditioners P and Q. We propose a method which transforms the matrix A to either a strictly diagonally dominant matrix, or a matrix in which some of the rows are strictly diagonally dominant which can be used to build a preconditioner for some iterative method (see [1]).

The outline of the paper is as follows. The method is established in Section 2. In the last section, some numerical examples and some comments are given.

2 The method of diagonal dominance

The SVD reduces matrix A to a diagonal matrix by orthogonal transformations. The LU and the QR factorizations (also other incomplete or modified factorizations) reduce the matrix A to some triangular matrix. We must eliminate many nonzero entries with these approaches which will cause fill-in problems. Fortunately, Yuan's result in [8] shows that it is not necessary to reduce the matrix A to a triangular or diagonal matrix for solving linear systems $Ax = b$ by iterative methods. We can keep more nonzero elements for the dense case, and as much as needed for the sparse case. We just reduce the matrix A to some strictly diagonally dominant (but not triangular or diagonal) matrix by orthogonal transformations. With the new strongly diagonally dominant matrix we can solve the linear system by some iterative

methods,such as the Gauss-Seidel method and the SOR method.

The idea of the method is as follows. One of the approaches to obtain the SVD of a given matrix is to apply Jacobi iterations (see [4, p.457]). Such iterations reduce the magnitude of the off-diagonal entries until all the off-diagonal entries are small enough.

The SVD is the limit case when the matrix obtained is diagonally dominant to the maximal possible extent. The idea is to stop process when some, or all, of the rows of the matrix are diagonally dominant but not when we have all elements very small. The method can be given as follows:

Choose m (the number of Jacobi iterations to be applied), and $\delta > 0$ (the level of diagonal dominance);

for $k = 1, \ldots, m$

Find $a_{i_0 j_0}$ such that $|a_{i_0 j_0}| = \max_{i \neq j} |a_{ij}|$;

$i_1 = \min(i_0, j_0)$; $j_1 = \max(i_0, j_0)$;

Compute $Q_k = \begin{pmatrix} a_{i_1 i_1} & a_{i_1 j_1} \\ a_{j_1 i_1} & a_{j_1 j_1} \end{pmatrix} = U_k \Sigma_k V_k^T$ (the SVD of matrix Q_k);

Apply the transformation U_k^T to rows i_1 and j_1 of matrix A: $A = (U^{(k)})^T A$, where the matrix $U^{(k)}$ differs from the unity matrix in the following positions: $U^{(k)}_{i_1 i_1} = (U_k)_{11}, U^{(k)}_{i_1 j_1} = (U_k)_{12}, U^{(k)}_{j_1 i_1} = (U_k)_{21}, U^{(k)}_{j_1 j_1} = (U_k)_{22}$;

Apply the transformation V_k to columns i_1 and j_1 of matrix A: $A = AV^{(k)}$, where the matrix $V^{(k)}$ differs from the unity matrix in the following positions: $V^{(k)}_{i_1 i_1} = (V_k)_{11}, V^{(k)}_{i_1 j_1} = (V_k)_{12}, V^{(k)}_{j_1 i_1} = (V_k)_{21}, V^{(k)}_{j_1 j_1} = (V_k)_{22}$;

Compute $b = (U^{(k)})^T b$;

end

Compute l (the number of rows in which $|a_{ii}| - \sum_{j \neq i} |a_{ij}| \geq \delta$);

Generate a preconditioner M: $M(1:l, 1:l) = A(1:l, 1:l)$, $M(i,i) = A(l,l), i = l+1, \ldots, n$;

Solve $Ay = b$ by some iterative method with a preconditioner M;

Compute $x = V^{(1)} \ldots V^{(m)} y$.

We choose the left upper corner of A as a preconditioner because the diagonal entries of the first rows converge to the largest singular values of A. Hence, the diagonal dominance is very strong in the first rows. The rest of the diagonal entries of M, i. e. $M(i,i), i = l+1, \ldots, n$, are chosen to be equal to $A(l,l)$ because this entry approximates the l-th singular value of A. In this way M is a well conditioned approximation of A.

In practice the number δ needs not be large because in most cases it measures the diagonal dominance in row l. The previous rows can have larger diagonal dominance. From our numerical experience we recommend the value $\delta \in [10^{-2}, 10^{-6}]$ (in double precision arithmetic with machine roundoff unit $\approx 10^{-16}$).

The preconditioner can be also chosen in other ways. For example, instead of $A(1:l, 1:l)$ we can take the tridiagonal part of $A(1:l, 1:l)$, and $M(i,i) = A(l,l), i = l+1, \ldots, n$, again.

3 Numerical experiments

In this section, we consider three examples. The experiments are done by Matlab. For simplicity, we introduce some notations in this section as follows:

FE: the relative error in ∞-norm, i.e.,

$$FE = \frac{\|x - \tilde{x}\|_\infty}{\|x\|_\infty},$$

IT: the number of iterations of the iterative method which was applied to solve the new systems,

Cnd: the condition number of the original matrix A with respect to $2-$norm,

Cndm: the condition number of the preconditioned matrix $M^{-1}A$ with respect to $2-$norm.

For comparison purpose, we choose b such that $Ax = b$ has the exact solution $x = (1, \ldots, 1)^T$ for all test problems.

We first apply the new method to Hilbert matrices H_n of different size. These matrices are well-known to be severely ill conditioned. For example, Gaussian elimination with partial pivoting breaks down when $n \geq 12$. We apply the bi-conjugate gradient method to solve the preconditioned system, and choose a tridiagonal preconditioner as it is described in Section 2. Even the matrices are not good test matrices, we can use the results to verify the improvement of the condition number of the new method.

The results for $\delta = 1e - 6$ and different choices of the matrix size n are given in Table 1 with $m = n^2$. We see that the number of iterations is not large when the matrix becomes larger. The error is quite satisfactory taking into account the behavior of the Gaussian elimination. It is not necessary to take $m = n^2$ for this example by our numerical tests.

n	10	20	30	40	50
FE	3.30e-4	4.20e-4	5.03e-5	8.29e-6	1.86e-5
IT	11	8	6	7	5

Table 1: The error and the number of iterations for the Bi-CG method with H_n when $\delta = 1e-6$

δ	1e-1	1e-2	1e-3	1e-4	1e-5	1e-6
FE	1.86e-5	1.86e-5	1.86e-5	1.86e-5	1.86e-5	1.86e-5
IT	23	18	15	11	9	5

Table 2: The error and the number of iterations for $n = 50$ and different choices of δ with H_n

In Table 2 the case when $n = 50$ is shown for different choices of δ for $m = n^2$. We see that the number of iterations reduces essentially when δ is small. This is because the preconditioner includes entries from more rows of the matrix $A^{(1)}$ for smaller δ. In this way we get a better preconditioner, and the number of iterations is less. Very small values for δ are not recommended because then the preconditioner becomes very ill conditioned.

We then apply the method to matrix $A_n = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} a & \text{if } i \neq j \\ a+1 & \text{if } i = j \text{ and } i \text{ is odd,} \\ a-1 & \text{if } i = j \text{ and } i \text{ is even.} \end{cases} .$$

The numerical results with $m \leq 3n$ for $a = 1e7$ are given in Table 3. We also test different δ for $n = 60$ and $a = 1e8$. the results for all $\delta \in (1e-1, 1e-6)$ are IT $= 1$, $m = 180$, $FE = 5.0715e-6$, Cnd $= 2.453e25$ and Cndm $= 4.9062e9$, where the stop criterion is $\|r_k\|/\|b\| < 1e-12$.

The method is also applied for nonsymmetric ill-conditioned matrices

n	10	20	30	40	50
Cnd	9.43e15	3.20e16	5.88e16	4.85e16	1.19e17
Cndm	9.62e7	1.89e8	3.03e8	3.93e8	4.76e8
FE	2.52e-8	8.35e-8	1.65e-7	2.74e-7	3.16e-7
IT	1	1	1	1	1
m	25	55	85	110	130

Table 3: The error and the number of iterations for the Bi-CG method with A_n when $\delta = 1e-6$

$Y_n = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} 1/(i+j-1) & \text{if } i \leq j, \\ (i-j)^k/(i+j-1), \quad k \geq 5 & \text{if } i > j \end{cases}.$$

For example, the condition number of A is 7.2173e14 when $k = 7$ and $n = 20$. The stop criterion of the Bi-CG method is $\|r_k\|/\|b\| \leq \epsilon$. The Bi-CG method without preconditioning obtains the solution with relative error $1.9778e-6$ under the tolerance $\epsilon = 1e-11$ in 96 iterations. The Bi-CG method does not converge to the solution if $\epsilon < 1e-11$. In order to compare with other preconditioners, we consider only the LU preconditioner and incomplete LU preconditioner for the Bi-CG method here. We also compare the Bi-CG method with our preconditioner to the Gaussian Elimination (GE) in Table 4. We use Matlab code to find the LU preconditioner and ILU preconditioner here with drop $1e-6$. The LU Bi-CG (Bi-CG with a complete LU preconditioner) and ILU Bi-CG (Bi-CG with an incomplete preconditioner) do not converge to the solution after 200 iterations because the preconditioners are very ill-conditioned. The numerical results of our method are given in Table 4. Note that our method is competitive with the Gaussian elimination

for our tested ill-conditioned systems.

We finally tested Riemann system $Ax = b$ where the matrix A is 100×100 Riemann matrix. In this case, the condition number of A is 480.5192. The Gaussian elimination obtains the solution with relative error $4.441e - 16$. The Bi-CG method without preconditioner obtains the approximate solution with tolerance $1e - 12$ at 177 steps. Preconditioned Bi-CG method with LU preconditioner given by Matlab does not obtain the solution after 200 steps because the preconditioner is very ill-conditioned. The Bi-CG method with our preconditioner obtains the solution with tolerance $1e - 12$ at 53 steps where $\delta = 1e - 6$ and $m = 10$. By our experiments, the value of m might not be big for well-conditioned systems.

In fact, our numerical tests illustrated that the value of m can be reduced to cn but not necessary cn^2 where $1 \leq c < n$ is an integer. This means the multiplication of finding the preconditioner is not very big. In general, $m \leq 5n$ is enough for well-conditioned matrices, and m is between $15n$ and $25n$ for ill-conditioned matrices.

All these results illustrated that our method is suitable to find good preconditioner for very ill-conditioned matrices. We do not present numerical tests for sparse matrices because the implementation of our method is much more complicated than the dense case. This is a topic for our further research.

Acknowledgement: The first author likes to give his sincere thanks to Professor Gene H. Golub and Professor Van der Vorst for their very helpful discussions and constructive suggestions on this topic during the Winter School in Hong Kong in 1995. The authors also thank Professor Golub and Professor Plemmons for introducing them to the references [2], [3] and [5]. We would like to thank and Professor Howard and Elman referees for their very helpful suggestions concerning the numerical tests, specially for compar-

n	k	Cnd	Cndm	FE	IT	m	GE
12	6	2.3827e10	2.6023e4	1.062e-8	5	120	2.757e-8
12	6	2.3827e10	7.5857e4	1.374e-8	7	100	
12	6	2.3827e10	2.8434e4	1.539e-8	9	90	
20	7	7.2173e14	3.3646e10	1.647e-6	86	120	1.740e-6
20	7	7.2173e14	1.6210e9	1.225e-6	40	180	
20	7	7.2173e14	2.9267e8	2.927e-5	29	220	
20	7	7.2173e14	1.6592e8	1.480e-6	28	300	
20	7	7.2173e14	2.5634e7	1.479e-6	20	350	
20	7	7.2173e14	1.0920e7	1.479e-6	16	400	

Table 4: The error and the number of iterations for the Bi-CG method with Y_n when $\delta = 1e-6$

ison with the Gaussian elimination. Their comments and suggetions helped us to improve the quality of the paper.

References

[1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, 1996.

[2] A. Berman and R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM, Philadelphia, 1994.

[3] M. Fiedler and V. Ptak, On matrices with nonpositive off-diagonal elements and positive principal minors, Czech. Math. J. 12(1962) 382-400

[4] G. H. Golub and C. F. Van Loan, Matrix Computations, John Hopkins University Press, Baltimore, 1996.

[5] M. Neumann, A note on generalizations of strict diagonal dominance for real matrices, Linear Algebra and its Applications, 26(1979) 3-14.

[6] V. V. Voevodin, Computational Processes in Linear Algebra, Nauka, Moscow, 1977. (In Russian)

[7] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

[8] J.Y. Yuan, Preconditioned Diagonal Dominant Matrices, Applied Mathematics and Computation, 114(2000) 255-262.