*J. Appl. Math. & Informatics Vol.* **28***(2010), No. 3 - 4, pp. 687 - 696*
Website: http://www.kcam.biz

# A PRECONDITIONER FOR THE NORMAL EQUATIONS

DAVOD KHOJASTEH SALKUYEH

ABSTRACT. In this paper, an algorithm for computing the sparse approximate inverse factor of matrix $A^TA$, where $A$ is an $m \times n$ matrix with $m \geq n$ and $rank(A) = n$, is proposed. The computation of the inverse factor are done without computing the matrix $A^TA$. The computed sparse approximate inverse factor is applied as a preconditioner for solving normal equations in conjunction with the CGNR algorithm. Some numerical experiments on test matrices are presented to show the efficiency of the method. A comparison with some available methods is also included.

AMS Mathematics Subject Classification : 65F10,65F50.
*Key word and phrases* : Inverse factors, Symmetric positive definite, Preconditioning, Incomplete QR, Incomplete $C$-orthogonalization, CGNR, PCGNR.

## 1. Introduction

Consider the normal equations

$$A^TAx = A^Tb, \tag{1}$$

where $A$ is a rectangular sparse matrix of size $m \times n$ with $m \geq n$. Moreover it is assumed that the matrix $A$ is of full rank, i.e., rank$(A) = n$. System (1) is known as the system of normal equations associated with the least-squares problem

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_2.$$

Hence the system (1) and the methods derived from it are often labeled with "NR" (N for "Normal" and R for "Residual"). It can be easily verified that the matrix $C = A^TA$ is a symmetric positive definite (SPD) matrix. Hence among the iterative methods for solving Eq. (1) the conjugate gradient (CG) algorithm is the method of choice and applying implicit CG algorithm to (1) results in the CGNR (also known CGLS) algorithm [11]. Since the convergence of the

Received March 22, 2008. Revised October 18, 2009. Accepted January 28, 2010.

CG algorithm depends of the spectrum of the coefficient matrix, its convergence behavior depends on

$$\lambda(A^TA) = \{\sigma^2 | \sigma \in \sigma(A)\},$$

that is, on the squares of the singular values of $A$. In the particular case that the matrix $A$ is square, the convergence behavior of the CG algorithm is governed by

$$\kappa_2(A^TA) = (\kappa_2(A))^2,$$

where $\kappa_2(A)$ is the spectral condition number of $A$. This relation shows that the convergence can be very slow even for matrices $A$ with moderate condition numbers.

To accelerate the convergence rate of an iterative method, it usually involves a second matrix that transforms the coefficient matrix into one with a more favorable spectrum. The transformation matrix is called a preconditioner. There are several ways for computing a preconditioner for the systems of normal equations. But the ideas behind most of them are the same and are based on incomplete variants of the QR factorization. Let $A = QR$ be the QR factorization of $A$ where $Q$ is $m \times n$ with orthonormal columns, and $R$ is $n \times n$ upper triangular with positive diagonal entries [7]. In this case $A^TA = R^TR$ is the Cholesky factorization of the normal equations matrix $A^TA$. Obviously we have

$$(AR^{-1})^T(AR^{-1}) = I_n.$$

To obtain a preconditioner approximate factorization $A \approx \bar{Q}\bar{R}$ is made where $\bar{R}$ is still upper triangular with positive diagonal entries, but the columns of $\bar{Q}$ may no longer be mutually orthogonal in general. In this case the preconditioned normal equations matrix would be as

$$(A\bar{R}^{-1})^T(A\bar{R}^{-1}) \approx I_n.$$

Note that there is no need for the $\bar{Q}$ factor in the iterative phase of the algorithm and therefore it does not need to be saved. See [4] for more details about the general purpose of the incomplete QR factorization.

Jennings and Ajiz in [9], considered the methods based on Givens rotations and methods based on the Gram-Schmidt precess for computing an incomplete QR factorization of $A$. For preserving sparsity in $\bar{R}$ a drop tolerance is used: fill elements are dropped if they are small according to some criterion. The possibility of the breakdowns is also discussed.

Another way for computing a preconditioner for the normal equations is to compute an incomplete Cholesky factorization of the normal equations matrix $C = A^TA$. In this method an incomplete Cholesky factor of $C$ is obtained explicitly. As noted in [6], there is actually no need to form all of $C$ explicitly; rather, its rows can be computed on at a time, used to perform the corresponding step of the incomplete Cholesky algorithm, and then discarded. As we know, the standard incomplete Cholesky factorization may fail for SPD matrices. Nevertheless, there exist reliable incomplete factorization algorithms that can be applied to a general SPD matrices without breakdowns (see for example [5, 12]).

Benzi and Tuma in [4] propose another method based on $C$-orthogonalization, i.e., orthogonalization with respect to inner product

$$\langle x, y\rangle_C = x^T C y, \quad \forall\, x, y \in \mathbb{R}^n.$$

In this method no entry of $C = A^T A$ needs to be explicitly computed, algorithm works entirely with $A$, the incomplete factorization process cannot break down and intermediate storage requirements are negligible. Numerical results presented in [4] show that the proposed preconditioner significantly reduces the solution time and iterations compared to the unpreconditioned iteration.

Let $C = R^T R$ be the Cholesky factorization of matrix $C$ where $R$ is an upper triangular matrix with positive entries on main diagonal. Let $U$ be a sparse approximation of $R^{-1}$, i.e., $U \approx R^{-1}$. The matrix $U$ is called a sparse approximate inverse factor for $C$. In this case we have

$$(AU)^T (AU)x = U^T A^T AU \approx I_n.$$

Now, the preconditioned normal equations can be written as

$$(AU)^T (AU) = U^T A^T b, \tag{2}$$

and the CGNR algorithm can be used computing its solution. Eq. (2) , will have the same solution as system (1), but may be easier to solve.

In this paper, an iterative method based on a projection technique for solving SPD linear systems of equations, which can be considered as a modification of the Gauss-Seidel method in conjunction with the AIB algorithm (approximate inverse via a bordering technique) [11] is used for computing a sparse approximate inverse factor of $C = A^T A$.

Throughout this paper, we use the following notations:

- $e_{n,k}$ : the $k$th column of the identity matrix of order $n$;
- $E_{n,k} = [e_1, e_2, \ldots, e_k]$, where $e_i$ is the $i$th column of the identity matrix of order $n$;
- $A(:, 1:k)$ : the $m \times k$ submatrix of $A$ containing of its first $k$ columns;
- $A(:, k)$ : the $k$th column of $A$;
- $A_k$ : the $k$th leading principal submatrix of $A$;
- $\langle ., . \rangle$ : the standard inner product in $\mathbb{R}^n$.

This paper is organized as follows. In section 2, an algorithm for computing a sparse approximate solution of an SPD linear system is presented. Section 3 is devoted to compute the sparse approximate inverse factor of $C = A^T A$ via sparse-sparse iterations. Numerical experiments are given in section 4. Section 5 is devoted to some concluding remarks.

## 2. An iterative method for solving SPD linear systems of equations

In this section, we derive an approach for solving SPD linear systems of equations which is provided by a projection method. Consider the following linear

system of equations

$$Bx = v, \tag{3}$$

where $B = (b_{ij})$ is an $n \times n$ SPD matrix and $x, v \in \mathbb{R}^n$. Let $\mathcal{L}$ and $\mathcal{K}$ be two $s$-dimensional subspaces of $\mathbb{R}^n$ and $x$ be an approximate solution of (3). A projection technique onto the subspace $\mathcal{K}$ and orthogonal to $\mathcal{L}$ is a process which finds an approximate solution $x_{new}$ to (3) by imposing the conditions that $x_{new}$ belongs to $x + \mathcal{K}$ and that the new residual vector be orthogonal to $\mathcal{L}$, i.e.,

$$\text{Find } x_{new} \in x + \mathcal{K}, \quad \text{such that } r_{new} := v - Bx_{new} \perp \mathcal{L}.$$

This framework is known as the Petrov-Galerkin conditions. Now, let $\mathcal{L} = \mathcal{K} = \text{span}\{e_i\}$, where $e_i$ is the $i$th column of the identity matrix. Hence the new approximate solution of (3) takes the form $x_{new} = x + \delta$ where $\delta \in \mathcal{K}$, i.e.,

$$x_{new} = x + \alpha e_i,$$

for some $\alpha \in \mathbb{R}$ . Now, we have

$$r_{new} = v - Bx_{new} = r - \alpha Be_i,$$

where the vector $r$ denotes the initial residual vector $r = v - Bx$. Then the Petrov-Galerkin condition $r_{new} \perp \mathcal{L}$ yields

$$\alpha = \frac{e_i^T r}{e_i^T B e_i} = \frac{r_i}{b_{ii}}, \tag{4}$$

where $r_i$ is the $i$th entry of $r$. The next theorem shows that how the appropriate index $i$ is chosen. This theorem also establishes the convergence of the method under some conditions.

**Theorem 1.** *According to the above procedure, we have*

$$\|d\|_B^2 - \|d_{new}\|_B^2 = \frac{r_i^2}{b_{ii}}, \tag{5}$$

*where* $d_{new} = B^{-1}v - x_{new}$, $d = B^{-1}v - x$ *and* $\|z\|_B = \|Bz\|_2 = \langle Bz, z\rangle^{1/2}$ *(B-norm of z) for any vector* $z \in \mathbb{R}^n$. *Moreover, assume that* $i$ *is selected at each projection step such a way that*

$$\frac{r_i^2}{b_{ii}} = \max\{\frac{r_j^2}{b_{jj}}, j = 1, 2, \ldots, n\}. \tag{6}$$

*Then the method converges for any initial guess.*

*Proof.* It can be readily verified that $d_{new} = d - \alpha e_i$ and $Bd = r$. Then

$$\begin{aligned}
\langle Bd_{new}, d_{new}\rangle &= \langle Bd - \alpha Be_i, d - \alpha e_i\rangle \\
&= \langle Bd, d\rangle - 2\alpha r_i + \alpha^2 b_{ii} \\
&= \langle Bd, d\rangle - \alpha r_i \qquad \text{from (4)} \\
&= \langle Bd, d\rangle - \frac{r_i^2}{b_{ii}}.
\end{aligned}$$

This relation results in Eq. (5). To prove the second part of theorem we see that Eq. (5) shows that the reduction in the square of the $B$-norm of the error is equal to $r_i^2/b_{ii} \geq 0$. Note that $b_{ii} > 0$, since $B$ is SPD. Now, if $r = 0$, then there is not anything to prove. Otherwise, by choosing index $i$ via Eq. (6) we have the maximum reduction in the square of the $B$-norm of the error. In fact, in this case we have

$$\|d_{new}\|_B < \|d\|_B,$$

and this proves the convergence of the method. □

Note that an elementary Gauss-Seidel step is a collection of projection steps with $\mathcal{L} = \mathcal{K} = \text{span}\{e_i\}$ cycled for $i = 1, 2, \ldots, n$. Hence the proposed method may be viewed an improvement of the Gauss-Seidel method.

In the next section we exploit the method described in this section for computing the sparse approximate inverse factor of $C = A^T A$.

## 3. Computing the sparse approximate factor of $C = A^T A$ via sparse-sparse iterations

In this section, we combine the method described in the previous section with the AIB algorithm [11] to derive an algorithm for computing a sparse approximate inverse factor of $C = A^T A$. Let $C_k$ be the $k$th leading principal submatrix of $C$. $C_k$ is an SPD matrix and

$$C_k = E_{n,k}^T C E_{n,k} = E_{n,k}^T A^T A E_{n,k} = (AE_{n,k})^T (AE_{n,k}) = A(:, 1:k)^T A(:, 1:k). \tag{7}$$

As we see in continue it does not need to compute $C_k$ explicitly. In the AIB algorithm, the sequence of matrices

$$C_{k+1} = \begin{pmatrix} C_k & v_k \\ v_k^T & \alpha_{k+1} \end{pmatrix}, \quad k = 1, 2, \ldots, n-1,$$

is made in which $v_k \in \mathbb{R}^k$, $\alpha_{k+1} \in \mathbb{R}$ and $C_n = C$. If the inverse factor $U_k$ is available for $C_k$, i.e.,

$$U_k^T C_k U_k = D_k,$$

then the inverse factor $U_{k+1}$ for $C_{k+1}$ will be obtained by writing

$$\begin{pmatrix} U_k^T & 0 \\ -z_k^T & 1 \end{pmatrix} \begin{pmatrix} C_k & v_k \\ v_k^T & \alpha_{k+1} \end{pmatrix} \begin{pmatrix} U_k & -z_k \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} D_k & 0 \\ 0 & \delta_{k+1} \end{pmatrix},$$

in which

$$C_k z_k = v_k, \tag{8}$$

$$\delta_{k+1} = \alpha_{k+1} - z_k^T v_k. \tag{9}$$

Relation (9) can be exploited if system (8) is solved exactly. Otherwise we should use

$$\begin{aligned} \delta_{k+1} &= \alpha_{k+1} - z_k^T v_k - z_k^T (v_k - C_k z_k) \\ &= \alpha_{k+1} - z_k^T (v_k + r_k), \end{aligned} \tag{10}$$

where $r_k$ is the residual obtained in solving (8). Starting from $k = 1$, a procedure for computing the inverse factor of $C$ is obtained. Here, another problem is the computation of $v_k$ and this can be done via

$$v_k = E_{n,k}^T C e_{n,k+1} = E_{n,k}^T A^T A e_{n,k+1} = (AE_{n,k})^T A e_{n,k+1} = A(:, 1:k)^T A(:, k+1). \tag{11}$$

The following theorem shows that $\delta_{k+1}$ is always positive, independently of the accuracy with which the system (8) is solved.

**Theorem 2.** *The scalar $\delta_{k+1}$ computed by Eq. (10) is positive.*

*Proof.* Let $r_k$ be the residual of the approximate solution of (8), i.e.,

$$r_k = v_k - C_k z_k.$$

Hence, we have

$$z_k = C_k^{-1}(v_k - r_k).$$

Now, by simple computations one can see that

$$\delta_{k+1} = \alpha_{k+1} - v_k^T C_k^{-1} v_k + r_k^T C_k^{-1} r_k = s + r_k^T C_k^{-1} r_k,$$

where $s = \alpha_{k+1} - v_k^T C_k^{-1} v_k \in \mathbb{R}$ is the Schur complement of $C_{k+1}$ and is a positive real number (see Theorem 3.9 in [1]). On the other hand $r_k^T C_k^{-1} r_k > 0$, since $C_k$ is an SPD matrix. Therefore, we conclude that $\delta_{k+1}$ is positive. □

This theorem shows that the AIB algorithm for computing an approximate inverse factor of $C = A^T A$ can not break down independently of the accuracy with which the system (8) is solved.

Computing sparse approximate solution for $C_k z_k = v_k, k = 1, 2, \ldots, n-1$, results in a sparse approximate inverse factor for $C = A^T A$. Relation (7) shows that system (8) is itself a system of normal equations (its coefficient matrix is SPD) and therefore for computing a sparse approximate solution for it the method described in the previous section can be exploited. Starting form a zero vector as an initial guess for the solution of $C_k z_k = v_k$ and running $lfil$ iterations of the method results in a solution for this system with at most $lfil$ nonzero entries. Since in each step of the projection technique used for the method only one entry of the current solution is modified and hence at most one entry is added to the current solution. Note that in this case each column of the inverse factor of $C = A^T A$ would have at most $lfil + 1$ nonzero entries.

By the above discussion we now state the following algorithm, **SAIF**[1]**-NR** $(lfil)$, for computing a sparse approximate inverse of $A^T A$ is obtained.

[1]Sparse Approximate Inverse Factor

**Algorithm 1. SAIF-NR($lfil$)**

1. Set $U_1 = [1]$ and $\delta_1 = \|A(:,1)\|_2^2$
2. For $k = 1, \ldots, n-1$ Do: (in parallel)
3. $v_k := A(:, 1:k)^T A(:, k+1)$
4. $z_k := 0$ and $r := v_k$
5. For $j = 1, \ldots, lfil$ and if $\|r\|_\infty > \tau$ Do:
6. Select the index $i$ via (6)
7. $\alpha = r_i / \|A(:,i)\|_2^2$
8. $z_k := z_k + \alpha e_i$
9. $r := r - \alpha A(:, 1:k)^T A(:, i)$
10. EndDo
11. $\delta_{k+1} = \alpha_{k+1} - z_k^T (v_k + r)$
12. Form $U_{k+1}$ and $D_{k+1}$
13. EndDo.
14. $D := D_n^{-\frac{1}{2}}$ and $U := U_n D$

Some observations can be posed here. First of all it can be easily seen that $b_{ii} = \|A(:,i)\|_2^2$ and hence it is used in steps 1 and 7 of the algorithm. An important note is that in step 3 of this algorithm only the entries over the main diagonal entry of column $k+1$ of $C = A^T A$ are computed and after computing the $k$th column of $U$ are discarded. Each column of $U$ is computed independently of other columns. Hence, the algorithm is suitable parallel computers. Finally, this algorithm with a little revision can be used for the normal equations

$$AA^T u = b, \text{ where } x = A^T u.$$

## 4. Numerical examples

All the numerical experiments presented in this section were computed in double precision in Fortran PowerStation version 4.0 on a personal computer Pentium 4, CPU 3.06 GHz, 1.00GB of RAM. For the first set of numerical experiments we used four matrices (WELL1033, ILLC1033, WELL1850, ILLC1850) from the Matrix Market website [10]. These matrices with their generic properties are given in Table 1. For each matrix we report the number $m$ of rows, the number $n$ of columns, and the number $nnz$ of nonzeros. In the last two columns we report iteration counts (under "CGNR") and CPU times (under "Time") for CGNR without preconditioning. Here and in all the other numerical experiments the stopping criterion used was

$$\|A^T(b - Ax_k)\|_2 < 10^{-8} \|A^T(b - Ax_0)\|_2.$$

In all cases we used the initial guess $x_0 = 0$, and the right-hand side $b$ was chosen so that the solution was $x = (1, 1, \ldots, 1)^T$. In all the table timings are in second.

Table 1 : First set of test problems.

| matrix | $m$ | $n$ | $nnz$ | CGNR | Time |
|---|---|---|---|---|---|
| WELL1033 | 1033 | 320 | 4732 | 164 | 0.16 |
| ILLC1033 | 1033 | 320 | 4732 | 830 | 0.77 |
| WELL1850 | 1850 | 712 | 8758 | 411 | 0.75 |
| ILLC1850 | 1850 | 712 | 8758 | 1262 | 2.27 |

Table 2 : Test results for SAIF-NR, RIF, ICNE and IMGS for first set of test matrices.

| | SAIF-NR | | | | | | RIF | | ICNE | | IMGS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| matrix | $lfil$ | size | P-T | Its | I-T | T-T | size | Its | size | Its | size | Its |
| WELL1033 | 5 | 930 | 0.031 | 97 | 0.031 | 0.062 | 911 | 72 | 866 | 108 | 797 | 147 |
| | 6 | 1022 | 0.047 | 94 | 0.031 | 0.078 | | | | | | |
| | 7 | 1120 | 0.054 | 92 | 0.031 | 0.085 | | | | | | |
| ILLC1033 | 4 | 811 | 0.016 | 160 | 0.063 | 0.079 | 825 | 256 | 816 | 286 | 807 | 3588 |
| | 5 | 911 | 0.031 | 148 | 0.047 | 0.078 | | | | | 1848 | 838 |
| | 6 | 1014 | 0.046 | 144 | 0.047 | 0.093 | | | | | 1982 | 484 |
| WELL1850 | 4 | 2451 | 0.031 | 201 | 0.156 | 0.187 | 2835 | 89 | 2595 | 182 | 2526 | 194 |
| | 5 | 2794 | 0.031 | 176 | 0.156 | 0.187 | | | | | 2849 | 150 |
| | 6 | 3089 | 0.031 | 176 | 0.156 | 0.187 | | | | | | |
| ILLC1850 | 5 | 2675 | 0.031 | 271 | 0.219 | 0.250 | 2904 | 248 | 2691 | 774 | 2608 | 1084 |
| | 6 | 2951 | 0.031 | 258 | 0.219 | 0.250 | | | | | 7031 | 155 |
| | 7 | 3208 | 0.031 | 250 | 0.218 | 0.249 | | | | | | |

Table 2 contains the results for the SAIF-NR($lfil$) for different values of $lfil$ together with the results for RIF (Benzi and Tuma's method), ICNE (incomplete Cholesky factorization preconditioner) and IMGS (incomplete QR preconditioner based on the modified Gram-Schmidt) presented in [4]. In this table we report the number of nonzeros in the incomplete factor or inverse factor (size), the time to construct the preconditioner (PT), the number of preconditioned CGNR iterations (Its), the time for the iterative solution phase (IT), and TT (=PT+IT). Since the structure of our computer and the computer used for the numerical results presented in [4] are different we only report SAIF-NR's time.

As we see, for the ILLC1033, the results of SAIF-NR algorithm is better than that of the RIF algorithm, but in general the converse is not correct. As we know the $C$-orthogonalization process is in general sequential but the SAIF-NR algorithm is inherently parallel. More investigation for the numerical experiments presented in Table 2 shows that the results of the SAIF-NR algorithm are often better than that of the ICNE and IMGS algorithms.

The second set of the numerical experiments are devoted to three large matrices RAEFSKY3, VENKAT01 and STAT96V1. These matrices can be extracted from the University of Florida Sparse Matrix Collection [8]. Let $A_1 =$ RAEFSKY3$(:, 1:5000)$, $A_2 =$ VENKAT01$(:, 1:5000)$ and $A_3 =$ STAT96V1$^T$. We use these matrices for our numerical tests. Some generic properties of $A_1$, $A_2$ and $A_3$ are given in Table 3. A † means that convergence was not attained in 10000 iterations for CGNR. Notations in this table and the next one are as before.

Table 3 : Second set of test problems.

| matrix | $m$ | $n$ | $nnz$ | CGNR | Time |
|---|---|---|---|---|---|
| $A_1$ | 21200 | 5000 | 347968 | † | - |
| $A_2$ | 62424 | 5000 | 137568 | † | - |
| $A_3$ | 197472 | 5995 | 588798 | 786 | 77.719 |

Table 4 : Numerical results for the second set of test matrices.

| matrix | $lfil$ | size | P-T | Its | I-T | T-T |
|---|---|---|---|---|---|---|
| $A_1$ | 8 | 31665 | 10.341 | 70 | 0.859 | 11.200 |
| | 9 | 33611 | 11.594 | 66 | 0.828 | 12.422 |
| | 10 | 35333 | 12.828 | 65 | 0.828 | 13.656 |
| $A_2$ | 8 | 38657 | 3.359 | 60 | 0.625 | 3.984 |
| | 9 | 41539 | 3.797 | 58 | 0.609 | 4.406 |
| | 10 | 44642 | 4.234 | 56 | 0.609 | 4.843 |
| $A_3$ | 8 | 22217 | 2.656 | 185 | 5.515 | 8.171 |
| | 9 | 22252 | 2.921 | 181 | 5.422 | 8.343 |
| | 10 | 22257 | 3.250 | 177 | 5.234 | 8.484 |

In Table 4 numerical results of the preconditioned CGNR algorithm with SAIF-NR preconditioner with different values of $lfil$ are given. As we see the proposed preconditioner is effective in reducing both the iteration count and the total solution time. Numerical results presented in this table shows that the parameter $lfil = 10$ usually gives good results.

## 5. Conclusion

We have proposed an algorithm for computing a sparse approximate inverse factor of the coefficient matrix of the normal equations. The computed sparse approximate inverse factor was used as a preconditioner for the normal equations. Numerical experiments show that the proposed preconditioner is robust and free from break down. A comparison with some available methods such as RIF, ICNE and IMGS methods was given. Numerical results show that the new preconditioner is more effective than the ICNE and IMGS methods. But, although in some cases our method gives better results than the RIF method but in general is not competitive with RIF method. Nevertheless, the main advantage of our method over the RIF method is that it is suitable for parallel computers.

## References

1. O. Axelsson, *Iterative solution methods*, Cambridge University Press, Cambridge, 1996.

2. M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. of Computational Physics **182** (2002), 418-477.
3. M. Benzi and M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math. **30** (1999), 305-340.
4. M. Benzi and M. Tuma, *A robust preconditioner with low memory requirements for large sparse least squares problems*, SIAM J. Sci. Comput. **25** (2003), 499-512.
5. M. Benzi and M. Tuma, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl. **10** (2003), 385-400.
6. A. Björck, *Numerical methods for least squares problems*, SIAM, Philadelphia, 1996.
7. B. N. Datta, *Numerical linear algebra and applications*, Pacific Grove, CA: Brooks/Cole Publishing Company, 1995.
8. T. Davis, *University of Florida Sparse Matrix Collection*, http: // www. cise. ufl. edu/~davis/sparse/, 2008.
9. A. Jennings and M. A. Ajiz, *Incomplete methods for soving* $A^T Ax = b$, SIAM J. Sci. Stat. Comput. **5** (1984), 978-987.
10. National Institute of Standards, *Matrix Market*, http://math.nist.gov/MatrixMarket, 2008.
11. Y. Saad, *Iterative Methods for Sparse linear Systems*, PWS press, New York, 1995.
12. M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl. **154/156** (1991), 331-353.

**Davod Khojasteh Salkuyeh** received his B.Sc from Sharif University of Technology, Tehran, Iran and his M.Sc from Ferdowsi University of Mashhad, Mashhad, Iran. He received his Ph.D degree under supervision of professor Faezeh Toutounian at Ferdowsi University of Mashhad in 2003. He is currently an assistant professor of Mathematics at University of Mohaghegh Ardabili, Ardabil, Iran. His research interests are mainly iterative methods for large sparse linear systems of equations and error analysis.

Department of Mathematics, University of Mohaghegh Ardabili,
P. O. Box. 179, Ardabil, Iran.
e-mail: khojaste@uma.ac.ir