J. Appl. Math. & Informatics Vol. **26***(2008), No. 1 - 2, pp. 213 - 222*
Website: http://www.kcam.biz

A PRECONDITIONER FOR THE LSQR ALGORITHM

SAEED KARIMI, DAVOD KHOJASTEH SALKUYEH* AND FAEZEH TOUTOUNIAN

ABSTRACT. Iterative methods are often suitable for solving least squares problems $\min\|Ax - b\|_2$, where $A \in \mathbb{R}^{m\times n}$ is large and sparse. The well known LSQR algorithm is among the iterative methods for solving these problems. A good preconditioner is often needed to speedup the LSQR convergence. In this paper we present the numerical experiments of applying a well known preconditioner for the LSQR algorithm. The preconditioner is based on the A^TA-orthogonalization process which furnishes an incomplete upper-lower factorization of the inverse of the normal matrix A^TA. The main advantage of this preconditioner is that we apply only one of the factors as a right preconditioner for the LSQR algorithm applied to the least squares problem $\min\|Ax - b\|_2$. The preconditioner needs only the sparse matrix-vector product operations and significantly reduces the solution time compared to the unpreconditioned iteration. Finally, some numerical experiments on test matrices from Harwell-Boeing collection are presented to show the robustness and efficiency of this preconditioner.

AMS Mathematics Subject Classification : 65F10, 65F20, 65N25, 65F50.
Key words and phrases : Least squares problem, the LSQR algorithm, preconditioner, iterative methods.

1. Introduction

In this paper we consider the solution of linear least squares problem

$$\min_{x\in\mathbb{R}^n}\|Ax - b\|_2, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$ are given, and $m \geq n$. We assume that A has full column rank. The linear least squares problems may be encountered in many scientific and engineering applications such as linear programming, geodetic survey problems, augmented Lagrange methods for computational fluid dynamics and natural factor method in structural engineering analysis [2, 4, 5, 7, 8, 11]. For large scale sparse problems iterative solution methods are often preferable

Received November 28, 2005. *Corresponding author.

to direct methods. Paige and Saunders presented an iterative algorithm for solving the problem (1), namely the LSQR algorithm [10]. This method is based on the bidiagonalization procedure of Golub and Kahan [6]. It is analytically equivalent to the standard method of conjugate gradients, but possesses more favorable numerical properties. It generates a sequence of approximations $\{x_k\}$ such that the residual norm $\|r_k\|_2$ decreases monotonically, where $r_k = b - Ax_k$. The LSQR algorithm generates two sets of vectors, $v_1, v_2, ..., v_k$ and $u_1, u_2, ..., u_k$ which form the orthonormal basis of the Krylov subspaces $\mathcal{K}_k(A^TA, v_1)$ and $\mathcal{K}_k(AA^T, u_1)$, respectively. We can easily show that the convergence rate of the LSQR algorithm is related to the condition number of the matrix A^TA. It is known that the condition number of A^TA is the square of the condition number of A, thus the LSQR algorithm may take many iterations to converge. A good preconditioner is needed to speedup the LSQR convergence. In this paper we observe that if we obtain an incomplete upper-lower factorization of $(A^TA)^{-1}$ of the form $(A^TA)^{-1} \approx \bar{R}\bar{R}^T$, then a preconditioner will be available for the LSQR algorithm. Based on an incomplete inverse factorization of A^TA which presented in [1], we obtain an approximate factor $\bar{R}$ and use it as a right preconditioner for (1). Applying this preconditioner requires only the matrix-vector products and can be done in parallel. Sparsity in $\bar{R}$ is preserved by applying a (relative) drop tolerance: fill elements are dropped if they are small according to some criterion. Numerical experiments show that the preconditioner significantly reduces the solution time compared to the unpreconditioned iteration.

Throughout this paper, we use the notation $< . >_2$ for the usual inner product in $\mathbb{R}^n$ and the associated norm denoted by $\|.\|_2$. For two vectors x and y, we define the following C-inner product:

$$< x, y >_C = < Cx, y >_2, \tag{2}$$

where C is symmetric positive definite matrix. The associated norm with respect to this inner product is as follows:

$$\|x\|_C = \sqrt{< x, x >_C}.$$

This paper is organized as follows. In section 2, a brief description of the LSQR algorithm and the proposed preconditioner are given. In section 3, numerical experiments on test matrices from Harwell-Boeing collection are presented to show the robustness and efficiency of the preconditioning technique. Finally, we give some concluding remarks in section 4.

2. The LSQR and the preconditioned LSQR algorithm

In this section, we recall some fundamental properties of the LSQR algorithm [10], which is an iterative method for solving real linear systems of the form

$$Ax = b, \tag{3}$$

where A is a large sparse matrix of size $m \times n, m \geq n$ and $x, b \in \mathbb{R}^n$.

The LSQR algorithm uses an algorithm of Golub and Kahan [6], which was stated as procedure Bidiag 1 in [10], to reduce A to the lower bidiagonal form. The procedure Bidiag 1 can be described as follows.

Bidiag 1 (starting vector b; reduction to lower bidiagonal form):

$$\beta_1 u_1 = b,\ \alpha_1 v_1 = A^T u_1$$

$$\left.\begin{array}{l} \beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i \\ \alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i \end{array}\right\},\ i = 1, 2, ... \qquad (4)$$

The scalars $\alpha_i \geq 0$ and $\beta_i \geq 0$ are chosen so that $\|u_i\|_2 = \|v_i\|_2 = 1$. With the definitions

$$U_k \equiv [u_1, u_2, ..., u_k],\quad V_k \equiv [v_1, v_2, ..., v_k],\quad B_k \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix},$$

the recurrence relations (4) may be rewritten as

$$\begin{aligned} U_{k+1}(\beta_1 e_1) &= b, \\ A V_k &= U_{k+1} B_k, \\ A^T U_{k+1} &= V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \end{aligned}$$

As we observe the procedure Bidiag 1 will be stop if $Av_i - \alpha_i u_i = 0$ or $A^T u_{i+1} - \beta_{i+1} v_i = 0$, for some i. In exact arithmetic, we have $U_{k+1}^T U_{k+1} = I$ and $V_k^T V_k = I$, where I is the identity matrix.

For the procedure Bidiag 1, we have the following propositions. The proof of these propositions are similar to those given in [12] for the classical Arnoldi process.

Proposition 1. *Suppose that k step of the procedure Bidiag 1 has been taken. Then the vectors* $v_1, v_2, ..., v_k$, *and* $u_1, u_2, ..., u_k$ *are orthonormal basis of the Krylov subspaces* $\mathcal{K}_k(A^T A, v_1)$ *and* $\mathcal{K}_k(AA^T, u_1)$, *respectively.*

Proposition 2. *The procedure Bidiag 1 will stop at step m if and only if* $min\{\mu, \lambda\}$ *is m, where* μ *is the grade of* v_1 *with respect to* $A^T A$ *and* λ *is the grade of* u_1 *with respect to* AA^T.

By using the procedure Bidiag 1 the LSQR algorithm constructs an approximation solution of the form $x_k = V_k y_k$ which solves the least-squares problem, $\min\|b - Ax\|_2$. The main steps of the LSQR algorithm can be summarized as follows (for more details see [10]).

Algorithm 1. The LSQR algorithm

Set $x_0 = 0$

Compute $\beta_1 = \|b\|_2,\ u_1 = b/\beta_1, \alpha_1 = \|A^T u_1\|_2,\ v_1 = A^T u_1/\alpha_1$

Set $w_1 = v_1$, $\bar{\phi}_1 = \beta_1$, $\bar{\rho}_1 = \alpha_1$
For $i = 1, 2, ...,$ until convergence Do:
$w = Av_i - \alpha_i u_i$
$\beta_{i+1} = \|w\|_2$
$u_{i+1} = w/\beta_{i+1}$
$z = A^T u_{i+1} - \beta_{i+1} v_i$
$\alpha_{i+1} = \|z\|_2$
$v_{i+1} = z/\alpha_{i+1}$
$\rho_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$
$c_i = \bar{\rho}_i/\rho_i$
$s_i = \beta_{i+1}/\rho_i$
$\theta_{i+1} = s_i \alpha_{i+1}$
$\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$
$\phi_i = c_i \bar{\phi}_i$
$\bar{\phi}_{i+1} = s_i \bar{\phi}_i$
$x_i = x_{i-1} + (\phi_i/\rho_i) w_i$
$w_{i+1} = v_{i+1} - (\theta_{i+1}/\rho_i) w_i$
(Test of convergence rate)
EndDo.

The following proposition can be stated for the convergence rate of the LSQR algorithm.

Proposition 3. *Let x_k be the approximate solution obtained at the k-th step of the LSQR algorithm, and let $r_k = b - Ax_k$. Then the 2−norm of the residual satisfies*

$$\|r_m\|_2 \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m \|r_0\|_2, \tag{5}$$

where κ is the condition number of the matrix $A^T A$ and r_0 is the initial residual.

Proof. This is a consequence of the fact that the LSQR algorithm are mathematically equivalent to the CGNR algorithm and for the conjugate gradient method applied to the normal equations $A^T Ax = A^T b$, the residual is reduced according to (5)(see [3]). □

As we observe, the convergence rate of the LSQR algorithm is related to the condition number of normal matrix $A^T A$. The condition number should in general be small to have fast convergence. However, in many application $A^T A$ have quite large condition number. Consequently, it is important then to modify equation (3). If we apply the LSQR algorithm to the transformed system

$$ARy = b, \; x = Ry, \tag{6}$$

where R is the factor of the inverse upper-lower factorization $(A^T A)^{-1} = RR^T$, then we reach to exact solution of the original system after one step. This is due

to the fact that

$$(AR)^T(AR) = R^T A^T AR = I, \tag{7}$$

where I is the identity matrix of order n. Therefore, if we obtain an incomplete inverse upper-lower factorization $(A^TA)^{-1} \approx \bar{R}\bar{R}^T$, then we can use the factor $\bar{R}$ as a right preconditioner for the LSQR algorithm applied to the problem (1).

We start by recalling that since A has full column rank, then the $n \times n$ matrix $C = A^TA$ is SPD and therefore it defines an inner product on $\mathbb{R}^n$ via (2). As explained in [1], by using the set of unit basis vectors $e_1, e_2, ..., e_n \in \mathbb{R}^n$, we can build a C-orthogonal (or C-conjugate) set of vectors $z_1, z_2, ..., z_n \in \mathbb{R}^n$ by a conjugate Gram-Schmidt process, i.e., a Gram-Schmidt process with respect to the inner product (2). Written as a modified Gram-Schmidt process, the (right-looking) algorithm starts by setting $z_j = e_j$, $j = 1, 2, ..., n$ and then performs the following nested loop:

$$z_i \leftarrow z_i - \frac{< z_j, z_i >_{A^TA}}{< z_j, z_j >_{A^TA}} z_j, \tag{8}$$

where $j = 1, 2, ..., n-1$ and $i = j+1, ..., n$. Letting $Z = [z_1, z_2, ..., z_n]$ and $D = diag(d_1, d_2, ..., d_n)$ with $d_j =< z_j, z_j >_{A^TA}$, we obtain the inverse upper-lower factorization

$$(A^TA)^{-1} = ZD^{-1}Z^T. \tag{9}$$

By noting that D is diagonal with entries $d_j = \|Az_j\|_2^2 > 0$, we can define $R = ZD^{-1/2}$. So we have the inverse upper-lower factorization $(A^TA)^{-1} = RR^T$.

A sparse preconditioner, an inverse approximate factorization $(A^TA)^{-1} \approx \bar{R}\bar{R}^T$, can be obtained by carrying out the updates in the A^TA-conjugation process (8) incompletely. Given a drop tolerance $0 < \tau < 1$, the entries of z_i are scanned after each update and entries that are smaller than τ in absolute value are discarded. We denote by $\bar{z}_i$ the sparsified vectors and we set

$$\bar{Z} = [\bar{z}_1, \bar{z}_2, ..., \bar{z}_n].$$

Letting $\bar{D} = diag(\bar{d}_1, \bar{d}_2, ..., \bar{d}_n)$ with $\bar{d}_j = \|A\bar{z}_j\|_2^2 > 0$, we have a sparse incomplete inverse factorization of A^TA of the form

$$(A^TA)^{-1} \approx \bar{Z}\bar{D}^{-1}\bar{Z}^T = (\bar{Z}\bar{D}^{-1/2})(\bar{D}^{-1/2}\bar{Z}^T) = \bar{R}\bar{R}^T. \tag{10}$$

The upper triangular matrix $\bar{R}$ is an approximate inverse factor that can be used as a right preconditioner for LSQR algorithm applied to (6).

The algorithm can be summarized as follows (for more details see [1]).

Algorithm 2. *C*-orthogonalization Process

Let $\bar{z}_j = e_j$, $j = 1, 2, ..., n$
For $j = 1, 2, ..., n-1$ Do:
  For $i = j+1, 2, ..., n$ Do:
    $\bar{z}_i = \bar{z}_i - (< \bar{z}_j, \bar{z}_i >_{A^TA} / < \bar{z}_j, \bar{z}_j >_{A^TA})\bar{z}_j$

Use a dropping strategy for the vector $\bar{z}_i$
EndDo
$\bar{d}_j = \|A\bar{z}_j\|_2^2$
EndDo.
Set $\bar{Z} = [\bar{z}_1, \bar{z}_2, ..., \bar{z}_n], \bar{D} = \mathtt{diag}(\bar{d}_1, \bar{d}_2, ..., \bar{d}_n)$, and $\bar{R} = \bar{Z}\bar{D}^{-1/2}$.

Note that the construction of the preconditioner does not require forming $C = A^TA$ explicitly. Indeed, in this algorithm, the main loop involves the computation of the inner products

$$< \bar{z}_j, \bar{z}_i >_{A^TA} = \bar{z}_j A^T A \bar{z}_i = (A\bar{z}_j)^T A\bar{z}_i, \ i = j, ..., n, \ j = 1, 2, ..., n-1.$$

Hence, computing these multipliers in the algorithm involve only matrix-vector products of the form $A\bar{z}_i$, to be computed as a linear combination of the columns of A corresponding to nonzero entries in $\bar{z}_i$, and inner products of two sparse vectors $A\bar{z}_i$ and $A\bar{z}_j$. Typically, most of these products will be structurally zero (that is,$A\bar{z}_i$ and $A\bar{z}_j$ have no zero entries in the same position) and the corresponding update in $\bar{z}_i$ can be skipped. We mention that the preconditioner is easily applied in parallel, since its application requires only matrix-vector products.

The straightforward application of the LSQR algorithm to the linear system (6) yields the following preconditioned version of the LSQR algorithm.

Algorithm 3. The LSQR algorithm with right preconditioning

Compute approximate inverse factor $\bar{R}$ by algorithm 2
Set $y_0 = 0$
Compute $\beta_1 = \|b\|_2$, $u_1 = b/\beta_1, q_1 = A^Tu_1, \alpha_1 = \|R^Tq_1\|_2$, and $v_1 = R^Tq_1/\alpha_1$
Set $w_1 = v_1$, $\bar{\phi}_1 = \beta_1$, and $\rho_1 = \alpha_1$
For $i = 1, 2, ...,$ until convergence Do:
$p_i = Rv_i$
$w = Ap_i - \alpha_i u_i$
$\beta_{i+1} = \|w\|_2$
$u_{i+1} = w/\beta_{i+1}$
$q_{i+1} = A^Tu_{i+1}$
$z = R^Tq_{i+1} - \beta_{i+1}v_i$
$\alpha_{i+1} = \|z\|_2$
$v_{i+1} = z/\alpha_{i+1}$
$\bar{\rho}_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$
$c_i = \bar{\rho}_i/\rho_i$
$s_i = \beta_{i+1}/\rho_i$
$\theta_{i+1} = s_i\alpha_{i+1}$
$\bar{\rho}_{i+1} = -c_i\alpha_{i+1}$
$\phi_i = c_i\bar{\phi}_i$
$\bar{\phi}_{i+1} = s_i\bar{\phi}_i$
$y_i = y_{i-1} + (\phi_i/\rho_i)w_i$

$w_{i+1} = v_{i+1} - (\theta_{i+1}/\rho_i)w_i$
If $|\bar{\phi}_{i+1}|$ is small enough then compute $x_i = Ry_i$.
EndDo.

3. Numerical examples

All the numerical experiments presented in this section were computed in double precision with some codes in the Fortran PowerStation on a PC Pentium 4, 256MHz. For the numerical experiments, we use some general matrices from Harwell-Boeing collection [9]. These matrices with their generic properties were shown in TABLE 1. This table gives, for each matrix, the order n and the number of nonzero entries nnz, condition number estimation (Cond-Est) and their application. We first consider the linear systems which their coefficient matrices are presented in TABLE 1. For all the examples, the right hand side of the system of equations were taken such that the exact solution is $x = [1, \cdots, 1]^T$. The stopping criterion

$$\frac{\| b - Ax_i \|_2}{\| b \|_2} < 10^{-7}, \tag{11}$$

was used and the initial guess was taken to be zero vector. Hereafter we use a dagger (†) to indicate that there was no convergence in 25000 iterations. We use the LSQR algorithm and the preconditioned LSQR (PLSQR) algorithms for solving the linear sytems of equations. The numerical results are given in TABLE 2. In columns 2 through 5 of TABLE 2 we give CPU times to compute the preconditioner (P-time), CPU times for the iterations (PIts-time), T-time (=P-time+PIts-time) of the PLSQR algorithm for convergence, and the number of iterations (P-Its), respectively. In the two last columns of TABLE 2, the number of iterations of unpreconditioned LSQR algorithm and its CPU times for convergence are presented. For all the examples we used the drop tolerance $\tau = 0.1$ for discarding the entries of $\bar{z}_i$ that are smaller than τ in absolute value.

Some observation can be posed based on these experimental results. From the point of view of robustness, we see that the LSQR algorithm in conjunction with the proposed preconditioner is a robust technique for solving the linear system of equations. As TABLE 2 shows, in the all examples, the sum of setup times to compute the preconditioner and CPU times for the iterations is always very less than the CPU times of the unpreconditioned LSQR algorithm. We also observe that the unpreconditioned LSQR algorithm fails on matrices SHERMAN3, SHERMAN5, ORSIRR1, SAYLR1, and has poor convergence on the other matrices, but the PLSQR algorithm never fails and on these matrices has reasonable convergence.

The rest of this section is devoted to solve large sparse least squares problems. Again, three rectangular matrices are chosen from the Harwell-Boeing collection; WELL1850 (1850 by 712 and $nnz = 8758$), ILLC1033 (1033 by 320 and $nnz = 4732$) and ILLC1850 (1850 by 712 and $nnz = 8758$). All the assumptions are as the first part of the numerical examples except for matrix ILLC1033 the drop

TABLE 1. Test problems information

matrix	n	nnz	Cond-Est	application
SHERMAN1	1000	3750	2.3e+04	Oil reservoir simulation
SHERMAN3	5005	20033	6.9e+16	Oil reservoir simulation
SHERMAN4	1104	3786	7.2e+03	Oil reservoir simulation
SHERMAN5	3312	20793	3.9e+05	Oil reservoir simulation
ADD20	2395	17319	1.8e+04	circuit modelling
ORSIRR1	1030	6858	1.0e+02	Oil reservoir simulation
ORSIRR2	886	5970	1.7e+05	Oil reservoir simulation
HOR131	434	4710	1.3e+05	Flow network problem
STEAM2	600	13760	3.5e+06	Enhanced oil recovery
SAYLR1	238	1128	1.6e+09	Oil reservoir simulation
SAYLR3	1000	3750	1.0e+02	Oil reservoir simulation
PDE900	900	4380	2.9e+02	Partial differential equation
PDE2961	2961	14585	9.5e+02	Partial differential equation

TABLE 2. Numerical results of the LSQR and PLSQR algorithms for the linear system of equations

	Prec. LSQR				Unprec. LSQR	
matrix	P-time	PIts-time	T-time	P-Its	It-time	Unp-Its
SHERMAN1	0.44	1.48	1.92	505	29.83	13494
SHERMAN3	9.28	16.04	25.32	1126	284.03	†
SHERMAN4	0.38	0.44	0.82	170	2.25	925
SHERMAN5	4.56	36.47	41.03	4268	198.72	†
ADD20	2.86	9.56	12.42	1573	29.66	5224
ORSIRR1	0.77	10.05	10.82	2996	61.90	†
ORSIRR2	0.6	4.55	5.15	1424	49.1	23071
HOR131	0.33	3.63	3.96	2218	24.17	21455
STEAM2	0.16	0.01	0.17	10	0.33	184
SAYLR1	0.01	0.33	0.34	217	13.01	†
SAYLR3	0.44	1.49	1.93	506	29.82	13487
PDE900	0.38	0.33	0.71	99	0.99	480
PDE2961	4.73	3.41	8.14	354	13.74	2004

tolerance $\tau = 10^{-5}$ was used. Numerical results are given in TABLE 3. As we observe, the results are similar to those of TABLE 2. This table shows that the PLSQR algorithm is more efficient than the LSQR algorithm for solving large sparse least squares problems.

TABLE 3. Numerical results for the least squares problems

	Prec. LSQR				Unprec. LSQR	
matrix	P-time	PIts-time	T-time	P-Its	It-time	Unp-Its
WELL1850	0.39	0.5	0.89	140	1.37	405
ILLC1033	1.1	0.6	1.7	159	6.27	3108
ILLC1850	0.39	4.62	5.01	1227	5.49	1634

TABLE 4. Numerical results for the matrix TOLS4000

TOLS(:,*nc*)	Prec. LSQR				Unprec. LSQR	
nc, *nnz*	P-time	PIts-time	T-time	P-Its	It-time	Unp-Its
800, 1106	0.11	0.00	0.11	3	90.85	22195
900, 1912	0.11	0.66	0.77	138	112.49	22636
1000, 2412	0.22	2.36	2.58	464	122.43	24190
1100, 2912	0.33	5.22	5.55	986	127.65	†
1200, 3412	0.44	9.39	9.83	1677	133.47	†
1300, 3912	0.44	15.05	15.49	2578	140.94	†

For the last part of this section we choose matrix TOLS4000 of order 4000 with 8784 nonzero entries from the Harwell-Boeing collection. Let TOLS(:,*nc*) denotes a matrix of order $n \times nc$ containing nc first columns of TOLS4000. We consider TOLS(:,*nc*) with different values of nc as the coefficient matrices of the least squares problems. The numerical results are presented in TABLE 4. In the first column of this table the number of columns of TOLS(:, nc), i.e, nc and the number of its nonzero entries are given. The rest of the assumptions are as before. As shown, all least squares problems are solved in a relatively small number of iterations by using PLSQR algorithm, but the unpreconditioned LSQR algorithm fails or has poor convergence. We also observe that the preconditioner significantly reduces the solution time compared to the unpreconditionded iteration.

4. Conclusion

We have presented the numerical experiments of a known preconditioner which is obtained by means of A^TA-orthogonalization process which furnishes an incomplete upper-lower factorization of the inverse of the normal matrix A^TA, applied to the LSQR algorithm. Numerical results show that this preconditioner is very effective to improve the convergence rate of the LSQR algorithm and PLSQR algorithm is much better than the LSQR algorithm in the number of iterations and CPU time.

REFERENCES

1. 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.
2. M.W. Berry and R.J. Plemmons, *Algorithm and experiments for structural mechanics on high-performance architectures*, Comput. Methods Appl. Mech. Engrg. **64** (1987), 487-507.
3. A. Björck, *Numerical Methods for Least Squares Problems*, SIAM. Philadelphia, PA, 1996.
4. I. Chio, C.L. monna and D. Shanno, *Future development of a primal-dual interior point method*, ORSA J. Coput. **2** (1990), 304-311.
5. M. Fortin and R. Glowinski, Augmented Lagrange methods: Application to the numerical solution of boundary-value problems, north-Holland, Amsterdam, 1983.
6. G.H. Golub and W. Kahan, *Algorithm LSQR is based on the Lanczos process and bidiagonalization procedure*, SIAM J. Numer. Anal., **2** (1965), 205-224.
7. M.T. Heath, R.J. Plemmos and R. C. Ward, *Sparse orthogonal schemes for structural optimization using force method*, SIAM J. Sci. Statist. Comput., **5** (1984), 514-532.
8. E.G. Kolata, *Geodesy: Dealing with an enormous computer task*, Science **200** 1978, 421-422.
9. Matrix Market, URL: `http://math.nist.gov/MatrixMarket`, October 2002.
10. C.C. Paige and M.A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Transactions on Mathematical Software, **8** (1982), 43-71.
11. J.R. Rice, PARRVEC workshop on very large least squares problems and supercomputers, Technical Report CSD-TR 464, Department of Computer Science, Purdue University, West Lafayette, IN, 1983.
12. Y. Saad, *Iterative Methods for Sparse linear Systems*, PWS press, New York, 1990.

Saeed Karimi received his B.Sc from Shiraz University, Shiraz, Iran and his M.Sc from Iran University of Science and Technology. He received his Ph.D degree under supervision of professor Faezeh Toutounian at Ferdowsi University of Mashhad in 2006. He is currently an assistant professor of Mathematics at Persian Gulf University, Bushehr, Iran. His research interests are mainly iterative methods for sparse linear system of equations and finite element method.

Department of Mathematics, Persian Gulf University, Bushehr, Iran
e-mail: karimi@pgu.ac.ir

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 Mohaghegh Ardabili University of Ardabil, Iran. His research interests are mainly iterative methods for sparse linear system of equations and finite element method.

Department of Mathematics, Mohaghegh Ardabili University,
P. O. Box. 56199-11367, Ardabil, Iran.
e-mail: khojaste@uma.ac.ir

Faezeh Toutounian received her B.Sc in Mathematics from Ferdowsi University of Mashhad, Iran, two degree of M.Sc in Mathematical statistics and applied computer and her Ph.D in Mathematics from Paris VI University, France. She spent two sabbatical years in 1985 and 1996 at Paris VI University. She is currently a professor of Mathematics at Ferdowsi University of mashhad. Her research interests are mainly numerical linear algebra, iterative methods and error analysis.
Department of Mathematics, School of Mathematical Sciences, Ferdowsi University of Mashhad, P.O. Box 1159-91775, Mashhad, Iran.
e-mail: toutouni@math.um.ac.ir