Iran J Sci Technol Trans Sci
https://doi.org/10.1007/s40995-020-01002-2

RESEARCH PAPER

Augmented Subspaces in the LSQR Krylov Method

Zahra Asgari[1] · Faezeh Toutounian[2] · Esmail Babolian[1]

Received: 28 October 2019 / Accepted: 2 October 2020

Abstract
The LSQR iterative method is a Krylov subspace method for solving least-squares problems. Early termination is rare, and it is common for LSQR to require many iterations before an approximation of the solution with desired accuracy has been determined. In this paper, we present a restarted LSQR method and we use a new technique for accelerating the convergence of restated by adding some approximate error vectors to the Krylov subspace. The effectiveness of the new method is illustrated by several examples.

Keywords Iterative method · Least-squares approximation · Krylov subspaces methods · LSQR

1 Introduction

In this paper, we consider the problem of finding a solution of least-squares problems

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

where $A \in \mathbf{R}^{n\times l}$ is a large sparse matrix with $n \geq l$ and $b \in \mathbf{R}^n$. For solving systems of linear equation (1), Krylov subspace methods have become one of the popular choices for solving (1); see (Hayami et al. 2010; Piage and Saunders 1982) and references therein. The LSQR method is a famous approach that is proposed by Piage and Saunders (1982). In this method, the matrix A is used only to compute products of the form Av and A^Tu for various vectors v and u. Hence, A will normally be large and sparse or will be expressible as a product of matrices that are sparse or have special structure. A typical application is to the large least-squares problems arising from the solution of the diffusion–convection equation with variable velocity field through the use of the dual reciprocity method in multidomains (Popov et al. 2007). Also, LSQR has been shown to be numerically more reliable in various circumstances than the other methods considered for solving some inverse problems (Jiang et al. 2007).

In LSQR method, the Golub–Kahan bidiagonalization process is applied, with initial vectors $u_1 = \frac{r_0}{\|r_0\|}$ and $v_1 = \frac{A^Tu_1}{\|A^Tu_1\|}$ to construct orthonormal bases $\{u_1, u_2, \ldots, u_m\}$ and $\{v_1, v_2, \ldots, v_m\}$ for the Krylov subspaces

$$\begin{aligned}\mathcal{K}_m(AA^T, u_1) &= \text{span}\{u_1, (AA^T)u_1, \ldots, (AA^T)^{m-1}u_1\},\\ \mathcal{K}_m(A^TA, v_1) &= \text{span}\{v_1, (A^TA)v_1, \ldots, (A^TA)^{m-1}v_1\}.\end{aligned} \tag{2}$$

The LSQR method finds an approximate solution x_m by minimizing $\|Ax-b\|_2$ over the subspace $x_0 + \mathcal{K}_m(A^TA, v_1)$. The associated residual vector $r_m = b - Ax_m$ lies in $\mathcal{K}_m(AA^T, u_1)$. The LSQR method will in exact arithmetic terminate before m steps have been carried out if the Krylov subspace $\mathcal{K}_m(A^TA, v_1)$ is of dimension less than m. LSQR delivers, in this situation, the solution of (1). However, early termination is rare and it is common for LSQR to require many iterations before an approximation of the solution x^* of (1) of desired accuracy has been determined. In Baglama et al. (2013), Baglama et al. explore the idea of an augmented LSQR method. Their method consists of two stages: first, the augmenting stage, which uses restarted LSQR to approximate the singular vectors associated with the smallest singular values of A and simultaneously

✉ Faezeh Toutounian
toutouni@math.um.ac.ir

Zahra Asgari
za.asgari93@gmail.com

Esmail Babolian
babolian@khu.ac.ir

[1] Department of Mathematics, Faculty of Mathematical Science and Computer, Kharazmi University, Tehran, Iran

[2] Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran

Published online: 19 October 2020

improve an available approximation of the solution of (1), and second, the LSQR stage, in which LSQR is applied using the augmented Krylov subspaces with fixed harmonic Ritz vectors to solve the least square problem. The proposed iterative method is not a restarted LSQR method. In this paper, we propose a restarted version of LSQR method. In restarted LSQR (LSQR(m)), the method is "restarted" once the Krylov subspace reaches dimension m, and the current approximate solution becomes the new initial guess for the next m iterations. The restarted parameter m is generally chosen small relative to n to keep the storage and computation requirements reasonable. In general, restarting slows the convergence of LSQR. When an iterative approach is restarted, the current approximation space is discarded at each restart. Therefore, a well-known drawback of LSQR(m) is that orthogonality to previously generated subspaces is not preserved at each restart. In fact LSQR(m) can stall as a result. Stalling means that there is no decrease in the residual norm at the end of a restart cycle. In this paper, we describe a new method for accelerating restarted LSQR method by adding approximate errors to the next restarted subspace.

This paper is organized as follows. In Sect. 2, we will give a review of LSQR method. In Sect. 3, we introduce our new technique. We present numerical results in Sect. 4. Finally, Sect. 5 summarizes our finding.

2 The LSQR Method

In this section, we recall some necessary properties of LSQR algorithm (Piage and Saunders 1982) which is one of the well-known iterative methods, for solving square and rectangular system of Eq. (1). The LSQR method uses an algorithm of Golub and Kahan (1965), which reduces A to the lower bidiagonal form. Let x_0 be an initial approximate solution of (1) and $r_0 = b - Ax_0$. The procedure bidiagonalization with starting vector r_0 can be described as follows:

$$\beta_1 u_1 = r_0, \alpha_1 v_1 = A^T u_1, \tag{3}$$

$$\left.\begin{aligned} \beta_{i+1} u_{i+1} &= A v_i - \alpha_i u_i, \\ \alpha_{i+1} v_{i+1} &= A^T u_{i+1} - \beta_{i+1} v_i, \end{aligned}\right\} \quad i = 1, 2, \ldots m, \tag{4}$$

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

$$U_m = [u_1, u_2, \ldots, u_m], V_m = [v_1, v_2, \ldots, v_m],$$

$$B_m = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_m & \alpha_m \\ & & & \beta_{m+1} \end{bmatrix},$$

the recurrence relations (3) and (4) can be rewritten as:

$$U_{m+1}(\beta_1 e_1) = r_0,$$
$$AV_m = U_{m+1} B_m,$$
$$A^T U_{m+1} = V_m B_m^T + \alpha_{m+1} v_{m+1} e_{m+1}^T.$$

In exact arithmetic, we have $U_m^{\mathrm{T}} U_m = I$ and $V_m^{\mathrm{T}} V_m = I$, where I is the identity matrix. The columns of U_m and V_m are orthonormal bases for the Krylov subspaces (2). By taking $x_m = x_0 + V_m y_m$, where $y_m \in \mathbf{R}^m$, and solving the least squares problem $min\|\beta_1 e_1 - B_m y\|_2$, the LSQR method constructs an approximation solution of (1), where $x_m \in x_0 + \mathcal{K}_m(A^T A, v_1)$ and the associated residual vector $r_m = b - Ax_m$ lies in $\mathcal{K}_m(AA^T, u_1)$. More details about LSQR method can be found in Piage and Saunders (1982).

3 A New Augmented LSQR Method

When an iterative approach is restarted, the current approximation space is discarded at each restart. Our technique attempts to accelerate the convergence of LSQR(m) by retaining some of the information that is typically discarded at the time of restart. Suppose that x^* is the true solution to the problem (1). The error after the ith restart cycle of LSQR(m) is denoted by e_i , where

$$e_i = x^* - x_i.$$

If our approximation space contains the exact correction e_i such that $x^* = x_i + e_i$, then we have solved the problem. We define

$$z_i \equiv x_i - x_{i-1}$$

as the approximation to the error after the ith LSQR(m) restart cycle, and $z_j \equiv 0$ for $j < 1$. From the fact that $x_i \in x_{i-1} + \mathcal{K}_m(A^T A, A^{\mathrm{T}} r_{i-1})$, we observe that $z_i \in \mathcal{K}_m(A^T A, A^{\mathrm{T}} r_{i-1})$. So, in some sense, this error approximation z_i represents the space $\mathcal{K}_m(A^T A, A^{\mathrm{T}} r_{i-1})$ generated in the previous cycle and subsequently discarded. Therefore, as pointed out in Baker et al. (2005), including an approximation to e_i (such as z_i) to the next approximation space $\mathcal{K}_m(A^T A, A^{\mathrm{T}} r_i)$ is a reasonable strategy. As LGMRES(m, k) Baker et al. (2005), the new restarted augmented LSQR algorithm, denoted by LLSQR(m, k), appends k previous approximations to the error to the current Krylov approximation space, and at the end of restart cycle $i + 1$, it finds an approximate solution to (1) in the following way:

$$x_{i+1} = x_i + q_{i+1}^{m-1}(A^T A) A^T r_i + \sum_{j=i-k+1}^{i} \beta_{ij} z_j,$$

where polynomial q_{i+1}^{m-1} and β_{ij} are chosen such that $\|r_{i+1}\|_2$ is minimized. Note that $k = 0$ corresponds to LSQR(m).

Let m be the dimension of Krylov subspace, and suppose that k given vectors $z_{i-k+1}, z_{i-k+2}, \ldots, z_i$ are the most recent errors used for adding to the Krylov subspace, and $s = m + k$. Typically, the number of vectors appended, k, is much smaller than the restart parameter m. Let $\Upsilon_k = [z_{i-k+1}, z_{i-k+2}, \ldots, z_i]$ and $[Q_k, R_k]$ be the reduced QR decomposition of $A\Upsilon_k$. Then, by defining $Y_k = [y_1, y_2, \ldots, y_k] = Q_k$ and starting with

$$\beta_1 u_1 = r_i - \sum_{j=1}^{k} h_{j1} y_j, \alpha_1 v_1 = A^T u_1, \tag{5}$$

we implement m steps of modified Golub–Kahan bidiagonalization process as follows:

$$\left.\begin{array}{l} \beta_{l+1} u_{l+1} = A v_l - \alpha_l u_l - \sum_{j=1}^{k} \bar{h_{jl}} y_j, \\ \alpha_{l+1} v_{l+1} = A^T u_{l+1} - \beta_{l+1} v_l, \end{array}\right\} \quad l = 1, 2, \ldots m, \tag{6}$$

where the coefficients $\bar{h_{jl}}, l = 1, \ldots, m$, are obtained by imposing orthogonality conditions

$$u_{l+1} \perp [y_1, y_2, \ldots, y_k], \quad \text{for} \quad l = 1, \ldots, m,$$

and the scalars $\alpha_l \geq 0$ and $\beta_l \geq 0$ are chosen so that $\|u_l\|_2 = \|v_l\|_2 = 1$. Let $\bar{H}_m = \{\bar{h_{jl}}\}_{j=1:k, l=1:m}$, then the recurrence relations (5) and (6) can be rewritten as:

$$\begin{aligned} U_{m+1}(\beta_1 e_{k+1}) &= r_i - \textstyle\sum_{j=1}^{k} h_{j1} y_j, \\ AV_m &= [Y_k \quad U_{m+1}] \begin{bmatrix} \bar{H}_m \\ B_m \end{bmatrix}, \\ A^T U_{m+1} &= V_m B_m^T + \alpha_{m+1} v_{m+1} e_{m+1}^T. \end{aligned}$$

Then, we have

$$A[\Upsilon_k \quad V_m] = [Y_k \quad U_{m+1}] \begin{bmatrix} R_k & \bar{H}_m \\ 0 & B_m \end{bmatrix}.$$

By defining

$$\hat{G}_s = \begin{bmatrix} R_k & \bar{H}_m \\ 0 & B_m \end{bmatrix}, \quad \hat{V}_s = [\Upsilon_k \quad V_m], \quad \hat{U}_{s+1} = [Y_k \quad U_{m+1}], \tag{7}$$

and using the fact that $\hat{U}_{s+1}^T \hat{U}_{s+1} = I_{s+1}$, the approximate solution over the subspace spanned by the columns of $\hat{V}_s$ can be computed by solving the following minimization problem:

$$\begin{aligned} \xi_s &= \min_{\xi \in \mathbf{R}^s} \|r_i - A\hat{V}_s \xi\| \\ &= \min_{\xi \in \mathbf{R}^s} \|r_i - \hat{U}_{s+1} \hat{G}_s \xi\| \\ &= \min_{\xi \in \mathbf{R}^s} \|\hat{e} - \hat{G}_s \xi\|, \end{aligned}$$

where $\hat{e} = [h_{11}, h_{21}, \ldots, h_{k1}, \beta_1, 0, \ldots, 0]^T$. The approximate solution can be formed by

$$x_{i+1} = x_i + z_{i+1}, \quad \text{with} \quad z_{i+1} = \hat{V}_s \xi_s.$$

Now, we can summarize the LLSQR(m, k) algorithm as shown in Algorithm 1.

For the LLSQR(m, k), the following theorem can be stated.

Theorem 1 *Suppose that we augment the Krylov space with k error approximation vectors $z_j = x_j - x_{j-1}, j = i - k + 1, \ldots, i$, then*

(i) $z_{i+1} \perp_{A^T A} \{z_j\}_{j=(i-k+1):i}$,

(ii) $cos\angle(r_{i+1}, r_{i-j}) = \frac{\|r_{i+1}\|}{\|r_{i-j}\|}$, for $0 \leq j \leq k$.

Proof The proofs of (i) and (ii) are similar to those of Theorems 3 and 6 in Baker et al. (2005), respectively. □

Algorithm 1 LLSQR (m,k)

1. Choose m, the maximum iterations of the Golub-Kahan bidagonalization process, k_1, the desired number of approximate error vectors, tol, the convergence tolerance. Set $s = m + k_1$, $x_0 = 0$, $k = 1$, $i = 1$, $Imax = 2500$.
2. Let $\beta_1 u_1 = b$, $\alpha_1 v_1 = A^T u_1$, perform s step of Golub-Kahan bidiagonalization process, solving $min\|\beta_1 e_1 - B_s y\|_2$ for y and generating V_s, U_{s+1} and B_s. Compute $x_1 = x_0 + V_s y$, $\Upsilon_k(:, 1) = V_s y$,
3. While $\frac{\|A^T r_i\|}{\|A^T r_0\|} > tol$ or $i < Imax$ do
   Let $[Q_k, R_k]$ be the reduced QR decomposition of $A\Upsilon_k$.
   Set $Y_k = [y_1, y_2, \ldots, y_k] = Q_k$.
   Let $\beta_1 u_1 = r_i - \sum_{j=1}^{k} h_{j1} y_j$, $\alpha_1 v_1 = A^T u_1$.
   for $l = 1 : m$ do
      $\beta_{l+1} u_{l+1} = A v_l - \alpha_l u_l - \sum_{j=1}^{k} \bar{h}_{jl} y_j$,
      $\alpha_{l+1} v_{l+1} = A^T u_{l+1} - \beta_{l+1} v_l$,
   end for.
   Determine the matrices $\hat{V}_s$, $\hat{U}_{s+1}$ and $\hat{G}_s$ by (7). Solve $\min_{\xi \in \mathbf{R}^s} \|\hat{e} - \hat{G}_s \xi\|$ for ξ,
   Compute $x_{i+1} = x_i + \hat{V}_s \xi_s$, $r_{i+1} = r_i - U_{s+1} \hat{G}_s \xi_s$, $z_{i+1} = x_{i+1} - x_i = \hat{V}_s \xi_s$
   If $i < k_1$, set $k = k + 1$, else set $\Upsilon_k(:, 1 : k - 1) = \Upsilon_k(:, 2 : k)$ end if.
   Set $\Upsilon_k(:, k) = z_{i+1}$. $i = i + 1$.
4. end while.

Table 1 Test problems information

	Matrix	Property			Matrix	Property	
		Order	nnz			Order	nnz
1	jpwh_991	991	6027	6	psmigr_ 3	3140	543,160
2	Sherman4	1104	3786	7	poisson3Da	13514	352,762
3	cavity05	1182	32,632	8	appu	14000	1,853,104
4	poli3	16,955	37,849				
5	add20	2395	13,151				

Table 2 Cycles, matrix–vector products, and CPU time required for convergence $\|A^{\mathrm{T}}r_i\|/\|A^{\mathrm{T}}r_0\| \leq 10^{-8}$ of methods

Matrix	LSQR		m	k	RLSQR (m + k)		LLSQR(m, k)	
	Iter (CPU)	Mvp			Cycle (CPU)	Mvp	Cycle (CPU)	Mvp
jpwh_991	322 (0.01)	646	60	4	12 (0.023)	1561	4 (0.02)	630
sherman4	914 (0.02)	1830	70	15	1392 (2.34)	239425	14 (0.11)	2370
cavity05	7534 (0.57)	15070	100	40	†	†	34 (0.82)	8680
add20	3633 (0.23)	7268	75	10	751 (3.86)	129173	40 (0.63)	6342
psmigr_3	121 (0.17)	244	20	5	62 (2.05)	3225	5 (0.22)	282
poisson3Da	2254 (2.91)	4510	175	10	208 (48.40)	77377	12 (5.75)	4686
appu	740 (4.92)	1482	60	10	13 (6.67)	1847	9 (4.99)	1330
cavity10	7385 (1.09)	14,772	118	40	†	†	40 (2.49)	11,398
poli3	2301 (0.95)	4604	120	15	38 (2.16)	10337	17 (4.73)	4596

The dagger (†) symbol indicates that no convergence is achieved after 2500 iterations

4 Numerical Results

In this section, we report some numerical results obtained by executing the new method for computing the solution of the least-squares problem (1). All the numerical experiments were performed in double precision floating point arithmetic in MATLAB R2018b. The machine we have used is a Intel(R) Core(TM) i7-4500U, CPU 1.80 GHz, 12.00 GB of RAM. For the tests, a set of nine problems were taken from the University of Florida Sparse Matrix Collection (Davis 2018). These matrices with their generic properties are given in Table 1. In all the examples, the starting guess was taken to be zero. The vector b also was chosen from University of Florida Sparse Matrix Collection (Davis 2018) when available, otherwise we considered the right-hand side $b = \mathrm{rand}(n, 1)$, where function rand creates an $n \times 1$ random vector with entries uniformly distributed in the interval [0, 1]. The stopping criterion

$$\|A^{\mathrm{T}}r_i\|/\|A^{\mathrm{T}}r_0\| \leq 10^{-8}$$

was used. We compare the results obtained by the LSQR, the RLSQR, and the LLSQR algorithms in terms of the number of iterations (Iter), the number of cycles (Cycles), the CPU time in seconds (CPU, in parentheses), and the

Table 3 Results for RLSQR(m + k) and LLSQR, m, k, median skip angle, and median sequential angle are listed for each problem

Matrix	m	k	RLSQR(m + k)		LLSQR(m, k)	
			Median skip angle $\angle(r_i, r_{i-1})$	Median seq. angle $\angle(r_{i+1}, r_{i-1})$	Median skip angle $\angle(r_i, r_{i-1})$	Median seq. angle $\angle(r_{i+1}, r_{i-1})$
jpwh_991	60	4	84.48	71.74	89.93	88.30
sherman4	70	15	9.24	6.54	33.47	24.68
add20	75	10	8.33	5.90	46.94	33.73
psmigr_3	20	5	37.11	26.74	84.18	83.26
poisson3Da	175	10	27.73	19.81	84.32	68.97
appu	60	10	79.17	64.24	85.18	72.92
poli3	120	15	61.86	46.63	79.45	65.28

number of matrix–vector products (Mvp) required for convergence. The results obtained are presented in Table 2. In this table, a dagger † indicates that no convergence is achieved after 2500 iterations. This table shows that except for matrix poli3 (its CPU time), for all other matrices the LLSQR algorithm is better (in terms of the number of cycles, the CPU time, and the number of matrix–vector products needed for convergence) than the RLSQR algorithm. Moreover, we observe that the number of matrix–vector products (Mvp) required for convergence of the LLSQR is smaller than the one for the LSQR (except for matrices Sherman4, psmigr_3, and poisson3Da; however, they are almost close). We also observe that for all matrices, the LSQR algorithm needs less CPU time than the RLSQR and LLSQR algorithms.

Finally, from Theorem 1, we observe that LLSQR is effective if it has large sequential angles and large skip angles. For some matrices of Table 1, we present in Table 3 the median sequential and median skip angle values. We observed that the LLSQR(m, k) algorithm has a larger median skip angle and median skip angle than does RLSQR(m + k) algorithm. These results indicate that the LLSQR(m, k) algorithm is preferable for its better numerical behavior convergence.

5 Conclusion

In this paper, we have presented an augmented LSQR method for solving large-scale linear LS problems or linear systems of equations, along with details of its implementation and experimental results. As we observed, the LLSQR augmentation scheme is an effective accelerator for RLSQR(m) method. In addition, numerical results showed that the new algorithm needs fewer matrix–vector products needed for convergence than the LSQR algorithm and provides better results than the RLSQR algorithm.

Acknowledgements We would like to thank the referees for their valuable remarks and helpful suggestions.

References

Baglama J, Reichel L, Richmond D (2013) An augmented LSQR method. Numer Algorithms 64:263–293

Baker AH, Jessup ER, Manteuffel T (2005) A technique for accelerating the convergence of restarted GMRES. SIAM J Matrix Anal Appl 26:962–984

Davis T (2018) The suitesparse matrix collection, http://www.cise.ufl.edu/research/sparse/matrices/. Accessed 2018

Golub GH, Kahan W (1965) Algorithm LSQR is based on the Lanczos process and bidiagonalization procedure. SIAM J Numer Anal 2:205–224

Hayami K, Yin J, Ito T (2010) GMRES methods for least squares problems. SAIM J Matrix Anal Appl 31:2400–2430

Jiang M, Xia L, Shou G, Tang M (2007) Combination of the LSQR method and a genetic algorithm for solving the electrocardiography inverse problem. Phys Med Biol 52:1277–1294

Piage CC, Saunders MA (1982) LSQR: an algorithm for sparse linear equations and sparse least squares. ACM Trans Math Softw 8 1:43–71

Popov V, Power H, Skerget L (2007) Domain decomposition techniques for boundary elements: application to fluid flow. WIT Press, Boston