Punjab University
Journal of Mathematics (ISSN 1016-2526)
Vol. 49(1)(2017) pp. 51-61

Repeated QR Updating Algorithm for Solution of Equality Constrained Linear Least Squares Problems

Salman Zeb
Department of Mathematics, University of Malakand, Chakdara,
Dir (Lower), Khyber Pakhtunkhwa, Pakistan
Email: zebsalman@gmail.com

Muhammad Yousaf
Department of Mathematics, University of Malakand, Chakdara,
Dir (Lower), Khyber Pakhtunkhwa, Pakistan
Email: myousafuom@gmail.com

Received: 14 June, 2016 / Accepted: 01 September, 2016 / Published online: 01 December, 2016

Abstract. We consider a repeated QR updating algorithm for the solution of equality constrained linear least squares problems. The constrained problem is first converted into the linear least squares problem using weighted factor and then it is partitioned into a small well-manageable problem by removing a pair of blocks of rows and columns. We perform the QR factorization of the small subproblem and then it is updated by appending the removed data. The proposed strategy is effective for large scale dense problems and also particulary suitable for parallel implementation due to its partitioning by using the number of passes. Some numerical experiments are given to illustrate the accuracy of the proposed algorithm and the results are compared with the solution obtained through the nullspace method.

AMS (MOS) Subject Classification Codes: 65-XX; 65Fxx; 65F20; 65F25
Key Words: Least squares problems, Equality constraints, QR decomposition, Updating.

1. INTRODUCTION

An equality constrained linear least squares problem (LSE)

$$\min_x \|Ax - b\|_2, \quad \text{subject to} \quad Bx = d, \tag{1. 1}$$

where $A \in R^{m\times n},\ b \in R^m,\ B \in R^{p\times n},\ d \in R^p$, $\|.\|_2$ denotes the Euclidean norm, with $m + p \geq n \geq p$ and $x \in R^n$ is considered. Further, we assume that

$$\text{rank}(B) = p, \quad \text{and} \quad \text{rank}\begin{pmatrix} A \\ B \end{pmatrix} = n, \tag{1. 2}$$

which ensures the existence and uniqueness of solution for problem (1. 1). The LSE problems arises in important applications such as constrained surface fitting, geodetic

least squares adjustment, optimal design of structures, beam-forming in signal processing, penalty function methods in nonlinear optimization, analysis of large scale structure, curve fitting, electromagnetic data processing and obtaining the solution of inequality constrained least squares problems [3, 5, 6, 7, 13, 14, 15, 16, 17, 22]. There exists various direct methods such as direct elimination, nullspace method, method of weighting, generalized singular value decomposition (GSVD), generalized QR factorization (GQR) for the solution of LSE problems [1, 7, 12, 13, 19, 21, 24, 18]. In weighting method, the LSE problem is converted into the unconstrained linear least squares problem (LLS) and then it can be solved using existing methods such as QR factorization, singular value decomposition (SVD) [7, 13].

In the present article, we re-write (1. 1) as

$$\min_{x(\gamma)} \left\| \begin{pmatrix} \gamma B \\ A \end{pmatrix} x - \begin{pmatrix} \gamma d \\ b \end{pmatrix} \right\|_2, \tag{1. 3}$$

where γ is a weighted factor and chosen as $\gamma \geq \|A\|_2/\|B\|_2\epsilon_M$, where ϵ_M is the machine epsilon [7, 21]. The assumption in (1. 2) ensures that it is a full rank least-squares problem and has a unique solution denoted by $x(\gamma)$ such that $\lim_{\gamma\to\infty} x(\gamma) = x_{LSE}$, where x_{LSE} represents the solution of LSE problem (1. 1). The problem (1. 3) is then solved using repeated QR updating algorithm. Numerous aspects of updating in various matrix factorizations are discussed in [2, 4, 7, 8, 9, 10, 11, 20, 23]. The updating approach is desired when further information is arrived and the user is interested to avoid solving the problem afresh. In [23], the author studied the updating techniques as a solution tool for LLS problems. We exploit the large structure of the problem (1. 3) by removing block of rows and columns using P times partition process to reduce it to a small incomplete problem, where P stands for the number of passes as explained in section 3. The QR Householder factorization of the small incomplete problem is obtained and then it is updated upto P times after appending the removed data in order to get the required structure for the back substitution. The proposed repeated QR updating algorithm is useful for large scale dense problems. Moreover, it is also suitable for parallel implementation due to its usage of the number of passes for reducing the problem data into the small well-manageable problems and then applying the updating techniques.

The present article is organized as follows. We discuss the basic concepts in section 2. In section 3, we present the repeated QR updating algorithm for LSE problem (1. 1). Numerical experiments and conclusion are provided in section 4 and 5 respectively.

2. PRELIMINARIES

We recall some important concepts in this section.

2.1. **The Method of Weighting.** The method of weighting is used for the solution of problem (1. 3) by applying the subroutines or programs which are applicable for the solution of linear least squares problems. However, care is required in selection of large values of γ for solution of (1. 3). Otherwise, the matrix under consideration will become poorly conditioned. In particular, the method of normal equations fails for large values of γ. For details, we refer to [7].

2.2. **The Nullspace Method.** The solution of LSE problem (1. 1) can be obtained using nullspace method [7]. Assuming the conditions given in (1. 2) are satisfied, we compute

the QR factorization of matrix B^T

$$Q_B^T B^T = \begin{pmatrix} R_B \\ 0 \end{pmatrix}, \tag{2. 4}$$

where

$$Q_B = (Q_1, Q_2), \quad Q_1 \in \mathbf{R}^{n\times p}, \quad Q_2 \in \mathbf{R}^{n\times(n-p)},$$

and $R_B \in \mathbf{R}^{p\times p}$ is upper triangular and nonsingular. Then Q_2 forms an orthogonal basis for the nullspace of B i.e. $\mathcal{N}(B) = \mathcal{R}(Q_2)$. Hence, any vector $x \in \mathbf{R}^n$ which satisfies $Bx = d$ is then written as

$$x = x_1 + Q_2 y_2, \quad x_1 = B^{\dagger} d = Q_1 R_B^{-T} d. \tag{2. 5}$$

Therefore, we can write

$$Ax - b = Ax_1 + AQ_2 y_2 - b, \quad y_2 \in \mathbf{R}^{n-p}.$$

Now, we need to solve the LLS problem

$$\min_{y_2} \|(AQ_2)y_2 - (b - Ax_1)\|_2, \tag{2. 6}$$

by applying existing techniques for its solution. Let y_2 be the minimum length solution to (2. 6) such that

$$y_2 = (AQ_2)^{\dagger}(b - Ax_1),$$

and let x_1 be given in (2. 5). Then, since $x_1 \perp Q_2 y_2$, it follows that

$$\|x\|_2^2 = \|x_1\|_2^2 + \|Q_2 y_2\|_2^2 = \|x_1\|_2^2 + \|y_2\|_2^2,$$

where x is the minimum norm solution to LSE problem (1. 1).

3. Repeated QR Updating Algorithm for Solution of LSE Problems

This section is devoted to repeated QR updating algorithm. We consider the problem (1. 3) and write it as

$$\min_{x(\gamma)} \| \ Cx - f \ \|_2, \tag{3. 7}$$

where $C = \begin{pmatrix} \gamma B \\ A \end{pmatrix} \in R^{(m+p)\times n}$ and $f = \begin{pmatrix} \gamma d \\ b \end{pmatrix} \in R^{m+p}$,

and $\lim_{\gamma\to\infty} x(\gamma) = x_{LSE}$. The problem (3. 7) is reduced to a small well-manageable incomplete problem by repeated partition process of removing pairs of block of rows and columns. Let P be the number of passes used for partitioning of the problem (3. 7). The partitioning process is carried out in the following manner:
We remove the block of rows $U_{1r} = C(m_1 + 1 : m + p, :) \in \mathcal{R}^{r_1\times n}$ from matrix C of the problem (3. 7) and $u_1 = f(m_1+1 : m+p) \in \mathcal{R}^{r_1}$ from the corresponding right-hand-side (RHS) as follow:

$$C_{1r} = C(1 : m_1, :), \quad f_1 = f(1 : m_1),$$

where $m_1 = (m+p) - r_1$ and r_1 is the size of the removed block of rows. The incomplete problem is

$$\min_{x_{1r}} \|C_{1r}x_{1r} - f_1\|_2 . \tag{3. 8}$$

Now, let us remove the block of columns $U_{1c} = C_{1r}(:, n_1+1 : n), \in \mathcal{R}^{m_1\times c_1}$ from matrix C_{1r} of incomplete problem (3. 8) as

$$C_{1c} = C_{1r}(:, 1 : n_1), \quad f_1 = f(1 : m_1),$$

where $n_1 = n - c_1$ and c_1 is the size of the removed block of columns. Hence, we get the following incomplete problem after a single pass

$$\min_{x_{1c}} \|C_{1c}x_{1c} - f_1\|_2 . \tag{3. 9}$$

In pass-2, the incomplete problem (3. 9) is reduced further to a small problem by removing a block of rows and a block of columns respectively. By removing block of rows, we obtain

$$\min_{x_{2r}} \|C_{2r}x_{2r} - f_2\|_2 , \tag{3. 10}$$

where

$$C_{2r} = C_{1c}(1 : m_2, :), \quad f_2 = f_1(1 : m_2),$$

and $U_{2r} = C_{1c}(m_2 + 1 : m_1, :),\ u_2 = f_1(m_2 + 1 : m_1)$, is the removed block of rows and $m_2 = m_1 - r_2$. Furthermore, removing a block of columns $U_{2c} = C_{2r}(:, n_2 + 1 : n_1) \in \mathcal{R}^{m_2 \times c_2}$ from the matrix C_{2r} of the incomplete problem (3. 10), we have

$$C_{2c} = C_{2r}(:, 1 : n_2), \quad f_2 = f_1(1 : m_2),$$

where $n_2 = n_1 - c_2$ and c_2 is the size of the removed block of columns. Thus, the incomplete problem after this pass is now

$$\min_{x_{2c}} \|C_{2c}x_{2c} - f_2\|_2 . \tag{3. 11}$$

Continuing in the same fashion until the P^{th}-pass, we obtain

$$\min_{x_{Pc}} \|C_{Pc}x_{Pc} - f_P\|_2 . \tag{3. 12}$$

The algorithm for the partitioning process is described below.

Algorithm 1 Partitioning

Input: $C \in \mathcal{R}^{(m+p)\times n}$, $f \in \mathcal{R}^{m+p}$, P the number of passes

1: **for** $i = 1 : P$ **do**
2: $m_i =$ Enter the number of rows for sub-problem in the ith pass
3: $U_r(i) = \{C_i(m_i + 1 : end, :)\}$
4: $C_i = C_i(1 : m_i, :)$
5: $u(i) = \{f_i(m_i + 1 : end, :)\}$
6: $f_i = f_i(1 : m_i, :)$
7: $n_i =$ Enter the number of columns for sub-problem in the ith pass
8: $U_c(i) = \{C_i(:, n_i + 1 : end)\}$
9: $C_i = C_i(:, 1 : n_i)$
10: **end for**

Now, we compute the QR decomposition of the reduced incomplete problem (3. 12) using the following algorithm.

Algorithm 2 QR factorization

Input: $C_{Pc} \in \mathcal{R}^{m_P \times n_P}$, $f_P \in \mathcal{R}^{m_P}$
Output: $Q_P \in \mathcal{R}^{m_P \times m_P}, R_P \in \mathcal{R}^{m_P \times n_P}$, $l_P \in \mathcal{R}^{m_P}$

$R_P \longleftarrow C_{Pc}$
$Q_P \longleftarrow I$
$n_1 = min(m_P - 1, n_P)$
for $i = 1 : n_1$ **do**
  $[v, \tau, R_P(i,i)] = house\,(R_P(i,i), R_P(i+1 : m_P, i))$
  $W_1 = R_P(i, i+1 : n_P) + v^T R_P(i+1 : m_P, i+1 : n_P)$
  $R_P(i, i+1 : n_P) = R_P(i, i+1 : n_P) - \tau W_1$
  if $i < n_P$ **then**
    $R_P(i+1, i+1 : n_P) = R_P(i+1, i+1 : n_P) - \tau v W_1$
  end if
  $W_2 = Q_P(1 : m_P, i) + Q_P(1 : m_P, i+1 : m_P)v$
  $Q_P(1 : m_P, i) = Q_P(1 : m_P, i) - \tau W_2$
  $Q_P(1 : m_P, i+1 : m_P) = Q_P(1 : m_P, i+1 : m_P) - \tau W_2 v^T$
end for
$l_P = Q_P^T f_P$

where $house$ represents the Householder algorithm applied on the concerned matrix and W_1 and W_2 are self explanatory intermediary variables used in the algorithm. Here, we use the Householder algorithm based on Parlett's formula [10] which provides a triangular matrix R with positive diagonal elements in contrast with the MATLAB build-in command qr used for the QR factorization of the matrices. Hence, we obtain

$$R_P = H_P \cdots H_1 C_{Pc}, \quad l_P = H_P \cdots H_1 f_P, \tag{3. 13}$$

where l_P is the updated (RHS) of f_P. In addition to it, we also find the updated block of columns $V_{Pc} = Q_P^T U_{Pc}$.

From now onward, we are ready for updating the factor R_P and the corresponding RHS l_P after appending the blocks of columns and rows respectively. By appending the block of columns V_{Pc} to the R_P factor of (3. 13), we obtain

$$R_{Pc} = \begin{pmatrix} R_P & V_{Pc} \end{pmatrix}. \tag{3. 14}$$

Now if R_{Pc} in equation (3. 14) is upper trapezoidal or upper triangular, then no further efforts are needed. Otherwise, we reduce R_{Pc} to upper triangular form by introducing the Householder matrices $H_{n_{P-1}}, \cdots, H_{n_{P+1}}$ such that

$$\tilde{R}_{Pc} = H_{n_{P-1}}, \cdots, H_{n_{P+1}} R_{Pc} \quad \text{and} \quad \tilde{l}_{Pc} = H_{n_{P-1}}, \cdots, H_{n_{P+1}} l_P, \tag{3. 15}$$

where $n_{P-1} = n_P + c_P$.

The process of appending the block of columns and then updated the R_P factor is performed with the help of the following algorithm.

Algorithm 3 Updating after appending a block of columns V_{Pc} to R_P

Input: $Q_P \in \mathcal{R}^{m_P \times m_P}$, $R_P \in \mathcal{R}^{m_P \times n_P}$, $U_{Pc} \in \mathcal{R}^{m_P \times c_P}$, $l_P \in \mathcal{R}^{m_P}$
Output: $\tilde{Q}_{Pc} \in \mathcal{R}^{m_P \times m_P}$, $\tilde{R}_{Pc} \in \mathcal{R}^{m_P \times (n_P + c_P)}$, $\tilde{l}_{Pc} \in \mathcal{R}^{m_P}$

$V_{Pc} = Q_P^T U_{Pc}$
$R_P(1 : m_P, n + 1 : n_P + c_P) \longleftarrow V_{Pc}(1 : m_P, 1 : c_P)$
if $m_P \leq n_P$ **then**
  $\tilde{R}_{Pc} = triu(R_{Pc})$
  $\tilde{l}_{Pc} \longleftarrow l_P$
  $\tilde{Q}_{Pc} \longleftarrow Q_P$
else
  for $j = k$ to $min(m_P, n_P + c_P)$ **do**
    $[v, \tau, R_P(j, j)] = house(R_P(j, j), R_P(j + 1 : m_P, j))$
    $W_1 = R_P(j, j + 1 : n_P + c_P) + v^T R_P(j + 1 : m_P, j + 1 : n_P + c_P)$
    $W_2 = Q_P(1 : m_P, j) + Q_P(1 : m_P, j + 1 : n_P + c_P)v$
    $R_P(j, j + 1 : n_P + c_P) = R_P(j, j + 1 : n_P + c_P) - \tau W_1$
    if $j < n_P + c_P$ **then**
      $R_P(j+1 : m_P, j+1 : n_P+c_P) = R_P(j+1 : m_P, j+1 : n_P+c_P) - \tau v W_1$
    end if
    $Q_P(1 : m_P, j) = Q_P(1 : m_P, j) - \tau W_2$
    $Q_P(1 : m_P, j + 1 : n_P + c_P) = Q(1 : m_P, j + 1) - \tau W_2 v^T$
    $l_{Pj} = l_P(j)$
    $l_P(j) = (1 - \tau) l_P(j) - \tau v^T l_P(j + 1 : m_P)$
    $l_P(j + 1 : m_P) = l_P(j + 1 : m_P) - \tau v l_{Pj} - \tau v v^T l_P(j + 1 : m_P)$
  end for
end if
$\tilde{R}_{Pc} = triu\begin{pmatrix} R_{Pc} \end{pmatrix}$
$\tilde{Q}_{Pc} \longleftarrow Q_P$
$\tilde{l}_{Pc} \longleftarrow l_P$

Next, the block of rows removed during the partition process will be appended and the resulting factor will be updated accordingly. Let us append the block of rows U_{Pr} to $\tilde{R}_{Pc}$ and u_{Pr} to its corresponding RHS as

$$R_{Pr} = \begin{pmatrix} \tilde{R}_{Pc} \\ U_{Pr} \end{pmatrix} \quad \text{and} \quad l_{Pr} = \begin{pmatrix} \tilde{l}_{Pc} \\ u_{Pr} \end{pmatrix}.$$

Furthermore, we construct the Householder matrices $H_{n_{P-1}} \cdots H_1$ in order to obtain the upper triangular factor $\tilde{R}_{Pr}$ as follow:

$$\tilde{R}_{Pr} = H_{n_{P-1}} \cdots H_1 \begin{pmatrix} \tilde{R}_{Pc} \\ U_{Pr} \end{pmatrix} \quad \text{and} \quad \tilde{l}_{Pr} = H_{n_{P-1}} \cdots H_1 \begin{pmatrix} \tilde{l}_{Pc} \\ u_{Pr} \end{pmatrix}. \qquad (3.\ 16)$$

The procedure described above is presented in the following algorithm.

Algorithm 4 Updating after appending a block of rows U_{Pr} to $\tilde{R}_{Pc}$

Input: $\tilde{Q}_{Pc} \in \mathcal{R}^{m_P \times m_P}$, $\tilde{R}_{Pc} \in \mathcal{R}^{m_P \times (n_P+n_c)}$, $U_{Pr} \in \mathcal{R}^{r \times n_P}$, $\tilde{l}_{Pc} \in \mathcal{R}^{m_P}$, $u_{Pr} \in \mathcal{R}^r$

Output: $\tilde{Q}_{Pr} \in \mathcal{R}^{(m_P+r)\times(m_P+r)}$, $\tilde{R}_{Pr} \in \mathcal{R}^{(m_P+r)\times(n_P+n_c)}$, $\tilde{l}_{Pr} \in \mathcal{R}^{m_P+r}$

$Q_{Pr} \longleftarrow \begin{pmatrix} Q_{Pc} & 0 \\ 0 & I_{Pr} \end{pmatrix}$
for $k = 1$ to $min(m_P, (n_P + n_c))$ **do**
  $[v, \tau, R_{Pr}(k,k)] = house(R_{Pr}(k,k), U_r(1:r,k))$
  $W_1 = R_{Pr}(k, k+1:(n_P+n_c)) + v^T U_{Pr}(1:r, k+1:(n_P+n_c))$
  $R_{Pr}(k, k+1:(n_P+n_c)) = R_{Pr}(k, k+1:(n_P+n_c)) - \tau W_1$
  if $k < n$ **then**
    $U_{Pr}(1:r, k+1:(n_P+n_c)) = U_{Pr}(1:r, k+1:(n_P+n_c)) - \tau v W_1$
  end if
  $W_2 = Q_{Pr}(1:m_P+r, k) + Q_{Pr}(1:m_P+r, m_P+1:m_P+r)v$
  $Q_{Pr}(1:m_P+r, k) = Q_{Pr}(1:m_P+r, k) - \tau W_2 v$
  $Q_{Pr}(1:m_P+r, m_P+1:m_P+r) = Q_{Pr}(1:m_P+r, m_P+1:m_P+r) - \tau W_2 v v^T$
  $l_{Pck} = l_{Pc}(k)$
  $l_{Pc}(k) = (1-\tau) l_{Pc}(k) - \tau v^T u_{Pr}(1:r)$
  $u_{Pr}(1:r) = u_{Pr}(1:r) - \tau v l_{Pck} - \tau v v^T u_{Pr}(1:r)$
end for
if $m_P < (n_P + n_c)$ **then**
  $[Q_{Pr}, R_{Pr}] = qr(U_r(:, m_P+1:(n_P+n_c))$
  $R_{Pr}(m_P+1:m_P+r, m_P+1:(n_P+n_c)) \longleftarrow R_{Pr}$
  $u_{Pr} = Q_r^T u_{Pr}$
  $Q_{Pr}(1:m_P+r, m_P+1:m_P+r) = Q_{Pr}(1:m_P+r, m_P+1:m_P+r) Q_{Pc}$
end if
$\tilde{R}_{Pr} \longleftarrow \begin{pmatrix} R_{Pr} \\ 0 \end{pmatrix}$
$\tilde{Q}_{Pr} \longleftarrow Q_{Pr}$
$\tilde{l}_{Pr} \longleftarrow \begin{pmatrix} l_{Pr} \\ u_{Pr} \end{pmatrix}$

We continue in the same fashion until we reached to the 1^{st}-pass. In last step of updating process, we obtained the final upper triangular factor R of the problem (3. 7) and the corresponding RHS l after appending block of columns U_{1c} and block of rows U_{1r} which were removed in the first pass of partition. That is, we append the block of columns $U_{1c} \in \mathcal{R}^{m_1 \times c_1}$ to $\tilde{R}_{2r}$ by first computing $V_{1c} = \tilde{Q}_{2r}^T U_{1c}$, we have

$$R_{1c} = \begin{pmatrix} \tilde{R}_{2r} & V_{1c} \end{pmatrix}. \tag{3. 17}$$

Now if matrix R_{1c} of equation (3. 17) is upper triangular then we will move to the next step. If that is not the case, then we define Householder matrices $H_n \cdots H_{n_1+1}$ in order to reduce R_{1c} of equation (3. 17) to upper triangular form. We obtain

$$\tilde{R}_{1c} = H_n \cdots H_{n_1+1} \begin{pmatrix} \tilde{R}_{2r} & V_{1c} \end{pmatrix} \quad \text{and} \quad \tilde{l}_{1c} = H_n \cdots H_{n_1+1} \tilde{l}_{2r}, \tag{3. 18}$$

where $n = n_1 + c_1$. This procedure is performed by using Algorithm 3.
Furthermore, the block of rows $U_{1r} \in \mathcal{R}^{r_1 \times n}$ and $u_1 \in \mathcal{R}^{r_1}$ are appended to matrix $\tilde{R}_{1c}$ and to its corresponding RHS $\tilde{l}_{1c}$ of equation (3. 18).

Here, we used again the Algorithm 4, and the process is performed as below:

$$R_{1r} = \begin{pmatrix} \tilde{R}_{1c} \\ U_{1r} \end{pmatrix}, \quad l_{1r} = \begin{pmatrix} \tilde{l}_{1c} \\ u_1 \end{pmatrix},$$

where U_{1r} and u_1 are blocks of rows and the corresponding RHS which were removed in the 1^{st}-pass of partition process. In order to obtain the upper triangular factor $\tilde{R}_{1r}$, we construct Householder matrices $H_1, \cdots, H_n$ as follows.

$$\begin{pmatrix} \tilde{R}_{1r} \\ 0 \end{pmatrix} = H_1 \cdots H_{n_1+c_1} \begin{pmatrix} \tilde{R}_{1c} \\ U_{1r} \end{pmatrix} \quad \text{and} \quad \tilde{l}_{1r} = H_1 \cdots H_{n_1+c_1} \begin{pmatrix} \tilde{l}_{1c} \\ u_1 \end{pmatrix}. \quad (3.\ 19)$$

We know that $n = n_1 + c_1$ and $(m + p) = m_1 + r_1$. So we can write

$$R = \begin{pmatrix} \tilde{R}_{1r} \\ 0 \end{pmatrix} \quad \text{and} \quad l = \tilde{l}_{1r},$$

where $R \in \mathcal{R}^{(m+p)\times n}$ is the updated factor of QR factorization of matrix C and $l \in \mathcal{R}^{(m+p)}$ is the corresponding RHS of the problem (3. 7) obtained through repeated updating process. The solution to the problem (3. 7) is then obtained through the MATLAB built-in command *backsub* for back-substitution.

Finally, we state the repeated QR updating algorithm in a compact manner as follow.

Algorithm 5 Repeated QR Updating Algorithm

Input: $A \in \mathcal{R}^{m\times n}$, $b \in \mathcal{R}^m$, $B \in \mathcal{R}^{p\times n}$, $d \in \mathcal{R}^p$,
Output: $x_{LSE} \in \mathcal{R}^n$

1: $C = \begin{pmatrix} \gamma B \\ A \end{pmatrix}$ and $f = \begin{pmatrix} \gamma d \\ b \end{pmatrix}$
2: **for** $i = 1$ to P **do**
3: $[C_{ir}, f_{ir}, U_{ir}, C_{ic}, U_{ic}] \longleftarrow$ **partition**(C, f)
4: **end for**
5: $[Q_P, R_P, l_P] \longleftarrow$ **qrv**(C_{Pc}, f_{Pc})
6: **for** $i = P : -1 : 1$ **do**
7: $[Q_{ic}, R_{ic}, l_{ic}] \longleftarrow$ **minsertbcols**$(Q_{ir}, R_{ir}, l_{ir}, U_{ic})$
8: $[Q_{ir}, R_{ir}, l_{ir}] \longleftarrow$ **minsertbrows**$(Q_{ic}, R_{ic}, l_{ic}, U_{ir}, u_{ir})$
9: **end for**
10: $[R, d] \longleftarrow [R_{ir}, l_{ir}]$
11: $x_{LSE} \longleftarrow$ **backsub**$(R(1 : n, :), l(1 : n))$

This algorithm calls upon the partition process, qrv, minsertbcols, minsertbrows and backsub which are the MATLAB implementations of Algorithm 1, Algorithm 2, Algorithm 3, Algorithm 4 and the backward substitution respectively.

In addition to this, we also performed complexity analysis of Algorithm 5 as follow:

TABLE 1. Complexity Analysis of Proposed Algorithm 5

Function	Cost	Comments
qrv	$2/3n_P^3 + 2s_P n_P^2 - 2c_P n_P^2 + 4c_P s_P n_P$	QR factorization of subproblem (3. 12) and $V_{Pc} = Q_P^T U_{Pc}$.
minsertbcols	$2s_P(2n_P + c_P^2) - 2/3(c_P^3 + 3n_P c_P^2)$	Appending block of columns c_P to R_P.
minsertbrows	$2n_P^2 r_P$	Appending block of rows r_P to $\tilde{R}_{Pc}$.
backsub	n^2	Back-substitution of $R(1:n,:)$ and $d(1:n)$.

The total approximated cost of the algorithm is $O(n^2 + n_P^3 + (s_P + r_P - c_P)n_P^2 - c_P^3 - n_P c_P^2 + (c_P + 1)s_P n_P)$ which depends on the number of passes P used in the reduction of the problem (3. 7) to a small subproblem and $m_P \ll s, n_P \ll n, r_P \ll s, c_P \ll n$, and $s = m + p \geq n \geq p$.

4. NUMERICAL EXPERIMENTS

In this section, we consider equality constrained linear least squares problems. The problem matrices are generated randomly with fixed elements using MATLAB built-in function $rand('twister')$. The matrices are taken as dense which means that most of its elements are non-zero. The description of the problem matrices with size, condition number κ and frobenius norm $\|.\|_F$ are given in Table 2. The condition number and frobenius norm of a matrix X are calculated with MATLAB command $cond(X)$ and $norm(X, \ 'fro')$ respectively.

TABLE 2. Description of Test Problems

Problem	Size of (A)	$\kappa(A)$	$\|A\|_F$	Size of (B)	$\kappa(B)$	$\|B\|_F$
1.	20×15	3.9189e+02	3.9050e+02	10×15	8.1331e+01	9.3514e+01
2.	50×30	3.0930e+02	8.7314e+02	20×30	1.6462e+02	1.8742e+02
3.	80×70	2.2092e+03	1.6860e+03	60×70	3.8523e+02	4.9304e+02
4.	500×300	1.1473e+03	8.7325e+03	300×300	4.0400e+04	2.2920e+03
5.	1000×500	1.1602e+03	1.5942e+04	400×500	9.3239e+02	3.4156e+03

The error obtained between the two solutions are given in Table 3 where x_{LSE} denotes the solution obtained by the proposed updating algorithm and x is the solution through nullspace method. Moreover, the size of the subproblems are given in the table which shows that for problem (1) and (2) we used two number of passes while for the rest of the problems we considered three passes in our experiments.

TABLE 3. Results Comparison

Problem	Proposed Algorithm		Nullspace Method	Relative Error
	Size of Subproblem	Elapsed time (sec)	Elapsed time (sec)	$\|x - x_{LSE}\|_2 / \|x\|_2$
1.	(8,6), (3,3)	0.0580	0.0013	4.0040e-15
2.	(15,15), (5,3)	0.0623	0.0016	1.1842e-14
3.	(50,50), (30,20), (10,5)	0.0723	0.0056	1.0079e-14
4.	(100,90), (50,40), (5,5)	2.8662	0.0886	3.4076e-14
5.	(500,500), (100,100), (50,50)	23.7295	0.0987	1.7551e-14

The relative error obtained between the two solutions showed that the proposed Algorithm 5 provided satisfactory results despite large weighted factor. Moreover, the obtained solution x_{LSE} satisfy the equality constraint system $Bx = d$. We calculated the elapsed time of both the methods. The elapsed time for our algorithm can be reduced further through its parallel implementation.

5. CONCLUSION

The linear least squares problem with equality constraint is considered. The problem is first reduced to unconstrained linear least squares problem using weighted factor. Then repeated QR updating algorithm is used in order to solve the obtained linear least squares problem. The proposed strategy is effective for large scale dense problems and particularly suitable for parallel implementation. In future, the work needs to be carried out in order to test the applicability of the proposed algorithm on ill-conditioned type matrices arising in constrained linear least squares problems.

6. ACKNOWLEDGEMENTS

We are thankful to the referees for their valuable comments and suggestions which improved the manuscript.

REFERENCES

[1] E. Anderson, Z. Bai and J. Dongarra, *Generalized QR factorization and its applications*, Linear Algebra and its Applications Elsevier **162**, (1992) 243-271.
[2] R. Andrew and N. Dingle, *Implementing QR factorization updating algorithms on GPUs*, Parallel Computing Elsevier **40**, No. 7 (2014) 161-172.
[3] R. J. Auton, B. Van and L. Michael, *Investigation of procedures for automatic resonance extraction from noisy transient electromagnetics data*, Final Report, Contract N00014-80-C-029, Office of Naval Research Arlington, VA 22217, 1981.
[4] $\mathring{A}$. Björck, L. Eldén and H. Park, *Accurate downdating of least squares solutions*, SIAM Journal on Matrix Analysis and Applications **15**, (1994) 549-568.
[5] J. L. Barlow and S. L. Handy, *The direct solution of weighted and equality constrained least-squares problems*, SIAM J. Sci. and Stat. Comput. **9**, No. 4 (1988) 704-716.
[6] J. L. Barlow, N. K. Nichols and R. J. Plemmons, *Iterative methods for equality-constrained least squares problems*, SIAM J. Sci. and Stat. Comput. **9**, No. 5 (1988) 892-906.
[7] $\mathring{A}$. Björck, *Numerical methods for least squares problems*, SIAM Philadelphia, PA, USA, 1996.
[8] J. Daniel, W. B. Gragg, L. Kaufman and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Mathematics of Computation **30**, (1996) 772-795.
[9] P. E. Gill, G. H. Golub, W. Murray and M. Saunders, *Methods for modifying matrix factorizations*, SIAM Mathematics of Computation **28**, (1974) 505-535.
[10] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press Baltimore, 1996.
[11] S. Hammarling and C. Lucas, *Updating the QR factorization and the least squares problem*, Manchester Institute for Mathematical Sciences, University of Manchester UK, 2008.

[12] S. Hammarling, *The numerical solution of the general Gauss-Markov linear model*, Mathematics in, 1987.
[13] C. L. Lawson and R. J. Hanson, *Solving least squares problems*, SIAM Philadelphia, 1995.
[14] L. Eldén, *Perturbation theory for the least squares problem with linear equality constraints*, SIAM Journal on Numerical Analysis **17**, No. 3 (1980) 338-350.
[15] O. Leringe and P. $\mathring{A}$. Wedin, *A comparison between different methods to compute a vector x which minimizes $\|Ax - b\|_2$ when $Gx = h$*, Department of computer science, Lund University, Lund, Sweden, 1970.
[16] C. V. Loan, *A generalized SVD analysis of some weighting methods for equality constrained least squares*, Matrix Pencils, Springer Berlin Heidelberg, (1983) 245-262.
[17] C. V. Loan, *On the method of weighting for equality-constrained least-squares problems*, SIAM Journal on Numerical Analysis **22**, No. 5 (1985) 851-864.
[18] D. Orban, *The projected Golub-Kahan process for constrained linear least-squares problems*, Cahier DU GERAD (2014) 1-15.
[19] C. C. Paige, *Some aspects of generalized QR factorizations, Reliable Numerical Computation*, (1990) 71-91.
[20] L. Reichel and W. B. Gragg, *FORTRAN subroutines for updating the QR decomposition*, ACM Trans. Math. Software **16**, (1990) 369-377.
[21] G. W. Stewart, *On the weighting method for least squares problems with linear equality constraints*, BIT Numerical Mathematics Springer **37**, No. 4 (1997) 961-967.
[22] M. Wei, *Algebraic properties of the rank-deficient equality-constrained and weighted least squares problems*, Linear algebra and its Applications Elsevier **161**, (1992) 27-43.
[23] M. Yousaf, *Repeated updating as a solution tool for linear least squares problems*, PhD thesis, University of Essex, UK, 2010.
[24] A. I. Zhdanov and S. Y. Gogoleva, *Solving least squares problems with equality constraints based on augmented regularized normal equations*, Applied Mathematics E-Notes **15**, (2015) 218-224.