SIAM J. MATRIX ANAL. APPL.
Vol. 41, No. 2, pp. 838–851

RESTRICTIVELY PRECONDITIONED CONJUGATE GRADIENT METHOD FOR A SERIES OF CONSTANTLY AUGMENTED LEAST SQUARES PROBLEMS[*]

RUI LI[†] AND ZENG-QI WANG[‡]

Abstract. In this study, we analyze the real-time solution of a series of augmented least squares problems, which are generated by adding information to an original least squares model repetitively. Instead of solving the least squares problems directly, we transform them into a batch of saddle point linear systems and subsequently solve the linear systems using restrictively preconditioned conjugate gradient (RPCG) methods. Approximation of the new Schur complement is generated effectively based on a previously approximated Schur complement. Owing to the variations of the preconditioned conjugate gradient method, the proposed methods generate convergence results similar to the conjugate gradient method and achieve a very fast convergent iterative sequence when the coefficient matrix is well preconditioned. Numerical tests show that the new methods are more effective than some standard Krylov subspace methods. Updated RPCG methods meet the requirement of real-time computing successfully for multifactor models.

Key words. least squares problem, conjugate gradient method, saddle point linear system, updated preconditioning

AMS subject classifications. 65F08, 65F10

DOI. 10.1137/19M1284853

1. Introduction.

Least squares is a standard approach in statistic regression analysis, which is one of the most important models in the fields of quantitative finance, signal processing, etc. In particular, the arbitrage pricing theory [16] of modern finance research holds that the expected return of a risk asset can be modeled as a linear function of the asset's sensitivities to a group of factors, which is commonly known as the multiple factor model, given as

$$b = Ax + e, \tag{1.1}$$

where $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$ consist of risky asset returns and the values of factors, respectively; $e \in \mathbb{R}^m$ refers to the asset's idiosyncratic random shock with mean zeros; and $x \in \mathbb{R}^n$ is the asset's sensitivity to the factors. Because e satisfies the Gauss–Markov hypothesis, the model problem could be rewritten as the least squares problem

$$\min_x \|b - Ax\|_2,$$

with the reasonable assumption that A is of full column rank. In some practical financial problems, as batches of linear equations are added in (1.1) with the real-time continuous production of data, the least squares problem is enlarged continually.

[*]Received by the editors September 3, 2019; accepted for publication (in revised form) by M. Stoll March 24, 2020; published electronically June 2, 2020.
https://doi.org/10.1137/19M1284853
Funding: This work was supported by the National Natural Science Foundation of China grant 11371022.

[†]PiTech, Beijing, 100098, People's Republic of China (lirui@pie314tech.com).
[‡]School of Mathematical Sciences and Ministry of Education Key Lab of Scientific and Engineering Computing, Shanghai Jiao Tong University, Shanghai, 200240, People's Republic of China (wangzengqi@sjtu.edu.cn).

We refer to the enlarged problems as the augmented least squares problems given as

$$\min_x \left\| \begin{bmatrix} b \\ d \end{bmatrix} - \begin{bmatrix} A \\ B \end{bmatrix} x \right\|_2, \tag{1.2}$$

where

$$B = \begin{bmatrix} \Delta B_1 \\ \Delta B_2 \\ \vdots \\ \Delta B_N \end{bmatrix} \in \mathbb{R}^{k\times n}, \quad d = \begin{bmatrix} \Delta d_1 \\ \Delta d_2 \\ \vdots \\ \Delta d_N \end{bmatrix} \in \mathbb{R}^{k}$$

include the added data $\Delta B_i \in \mathbb{R}^{k_i\times n}$, $\Delta d_i \in \mathbb{R}^{k_i}$, and $k = k_1 + k_2 + \cdots + k_N$, and $i = 1, 2, \ldots, N$ is the number of updates.

Direct methods based on QR factorizations are efficient for small or middle sized least squares problems. To solve large scale least squares problems, the most commonly used algorithmic frameworks are the LSQR [15] and CGLS methods [3]. In recent years, these methods have been developed for different types of modified least squares problems. Updating QR decomposition is an essential technique for the LSQR method. Updating QR decomposition for the least squares problems in which rows or columns are added or deleted is proposed in [10]. Updating QR decomposition for modified least squares problems with small perturbation is studied in [14]. We refer the reader to [1] and the references therein for more implementation details of updating QR factorizations. With respect to CGLS methods, updating Cholesky decomposition plays an important role in providing good preconditioners for modified least squares problems. We refer the reader to [5, 6, 7] and the reference therein for details.

Recently, Marin et al. [13] generated the updating preconditioner of (1.2) by approximating the inverse of saddle point problems given as

$$\mathcal{A}y = \begin{bmatrix} A^TA & B^T \\ -B & I \end{bmatrix} \begin{bmatrix} x \\ Bx \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix}, \tag{1.3}$$

with incomplete Cholesky factorization [12, 17]. This preconditioned CGLS (PCGLS) method is efficient when the added rows are not very large in number. Several different types of modified least squares problems are analyzed, including row-adding, row-removing, variable-adding, and variable-removing. We call this method the updating preconditioned CGLS method (UPCGLS).

In this study, we exploit a real-time computing iterative method for solving batches of augmented least squares problems. Instead of solving the least squares problem (1.2) by the LSQR method or solving the equivalent normal equations

$$Nx :\equiv (A^TA + B^TB)x = A^Tb + B^Td :\equiv c \tag{1.4}$$

by the conjugate gradient method, we study the solution of a constantly augmented saddle point system (1.3). We apply a restrictively preconditioned conjugate gradient (RPCG) method [2] and update the preconditioner gradually with row addition in B.

This paper is organized as follows. In section 2, we introduce the framework of the RPCG method. In section 3, we consider an updated restrictively preconditioned conjugate gradient (URPCG) method for solving the augmented least squares problems. The numerical results are reported in section 4. Finally, in section 5, we present the conclusions briefly.

2. RPCG method. The conjugate gradient (CG) method [11] is a state-of-the-art method for solving symmetric positive definite linear systems. When the coefficient matrix is indefinite or nonsymmetric, the CG method usually fails to generate or break down the convergent sequence. As a result, a class of CG methods in nonstandard inner products was defined by introducing the magical preconditioners, for instance, the Bramble–Pasciak CG method [4]. We refer the reader to [18] and references therein for further information. In [2], the authors proposed a framework of the preconditioned CG method for solving a nonsingular system of linear equations given as

$$\mathbf{K}\mathbf{x} = \mathbf{b}, \qquad \mathbf{K} \in \mathbb{R}^{n\times n}, \quad \text{and} \quad \mathbf{x}, \mathbf{b} \in \mathbb{R}^n. \tag{2.1}$$

It is said that any nonsingular matrix possesses the factorization $\mathbf{K} = \mathbf{PHQ}$, where $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{n\times n}$ are nonsingular and $\mathbf{H}$ is symmetric positive definite. Let $\mathbf{G} = \mathbf{L}^T\mathbf{L}$ be a symmetric positive definite approximation of $\mathbf{H}$; then (2.1) can be equivalently written as the following symmetric positive definite system:

$$\mathbf{R}\hat{\mathbf{x}} = \hat{\mathbf{b}}, \quad \mathbf{R} = \mathbf{L}^{-T}\mathbf{H}\mathbf{L}^{-1}, \quad \hat{\mathbf{x}} = \mathbf{L}\mathbf{Q}\mathbf{x}, \quad \hat{\mathbf{b}} = \mathbf{L}^{-T}\mathbf{P}^{-1}\mathbf{b}.$$

The RPCG method is defined by performing the CG method on the above preconditioned symmetric positive definite linear systems. The matrix $\mathbf{M} = \mathbf{PGQ}$ is defined as the restrictive preconditioner of $\mathbf{K}$. By a series of variable substitutions, the **RPCG** method is obtained (see Method 2.2 in [2] for details). The RPCG method can be placed in the big group of nonstandard inner product CG methods even though it was defined originally in a different perspective. According to the analysis in [18],

$$\langle u, v\rangle_{\mathbf{W}} := u^T\mathbf{W}v$$

is an inner product since $\mathbf{W} = \mathbf{Q}^T\mathbf{G}\mathbf{Q}$ is symmetric positive definite. Then $\mathbf{M}^{-1}\mathbf{K}$ is positive definite in $\langle u, v\rangle_{\mathbf{W}}$. The RPCG method is equivalent to the CG method in the nonstandard inner product; see the equivalent algorithms in [9]. One of the important applications of the RPCG method is solving the generalized saddle point problem. In the present study, we consider the RPCG method for solving the sequence of linear systems (1.3). The saddle point matrix in (1.3) has the following factorization:

$$\mathcal{A} = \underbrace{\begin{bmatrix} I & 0 \\ -B(A^TA)^{-1} & I \end{bmatrix}}_{\mathcal{P}} \underbrace{\begin{bmatrix} A^TA & 0 \\ 0 & S \end{bmatrix}}_{\mathcal{H}} \underbrace{\begin{bmatrix} I & (A^TA)^{-1}B^T \\ 0 & I \end{bmatrix}}_{\mathcal{Q}}, \tag{2.2}$$

where

$$S = I + B(A^TA)^{-1}B^T \tag{2.3}$$

is the negative Schur complement of $\mathcal{A}$. We denote a symmetric positive definite approximation of S by $\widehat{S}$. Then the symmetric positive definite matrix is given as

$$\mathcal{G} = \begin{bmatrix} A^TA & 0 \\ 0 & \widehat{S} \end{bmatrix}, \tag{2.4}$$

which serves as a preconditioner for $\mathcal{H}$. Hence, $\mathcal{M} = \mathcal{PGQ}$ is the corresponding restrictive preconditioner of $\mathcal{A}$. We use the RPCG method for solving the saddle point system (1.3) as shown in Algorithm 2.1.

Algorithm 2.1 RPCG Method for Saddle Point System (1.3).

```
1: Choose $x_0 \in \mathbb{R}^n$ and compute ${r_0}^{(1)} = c - A^T A x_0 - B^T B x_0$
2: Let $r_0^{(2)} = 0$, $r_0 = ({r_0^{(1)}}^T, {r_0^{(2)}}^T)^T$, $z_0 = ({z_0^{(1)}}^T, {z_0^{(2)}}^T)^T$, and $v_0 = ({v_0^{(1)}}^T, {v_0^{(2)}}^T)^T$
3: Solve $A^T A t^{(1)} = r_0^{(1)}$
4: Solve $\widehat{S} z_0^{(2)} = B t^{(1)}$
5: Solve $A^T A \widetilde{t}^{(1)} = B^T z_0^{(2)}$
6: Compute $z_0^{(1)} = t^{(1)} - \widetilde{t}^{(1)}$
7: Compute $v_0^{(1)} = z_0^{(1)} + 2\widetilde{t}^{(1)}$
8: Let $v_0^{(2)} = z_0^{(2)}$, $p_0 = z_0$, and $q_0 = v_0$
9: for $k = 0, 1, 2, \ldots$ do
10:    $\alpha_k = -\frac{v_k^T r_k}{q_k^T \mathcal{A} p_k}$
11:    $x_{k+1} = x_k - \alpha_k p_k^{(1)}$
12:    $r_{k+1} = r_k + \alpha_k \mathcal{A} p_k$
13:    Let $r_{k+1} = ({r_{k+1}^{(1)}}^T, {r_{k+1}^{(2)}}^T)^T$, $z_{k+1} = ({z_{k+1}^{(1)}}^T, {z_{k+1}^{(2)}}^T)^T$, $v_{k+1} = ({v_{k+1}^{(1)}}^T, {v_{k+1}^{(2)}}^T)^T$
14:    Solve $A^T A t^{(1)} = r_{k+1}^{(1)}$
15:    Solve $\widehat{S} z_{k+1}^{(2)} = r_{k+1}^{(2)} + B t^{(1)}$
16:    Solve $A^T A \widetilde{t}^{(1)} = B^T z_{k+1}^{(2)}$
17:    Compute $z_{k+1}^{(1)} = t^{(1)} - \widetilde{t}^{(1)}$
18:    Compute $v_{k+1}^{(1)} = z_{k+1}^{(1)} = 2\widetilde{t}^{(1)}$
19:    Let $v_{k+1}^{(2)} = z_{k+1}^{(2)}$
20:    $\beta_k = \frac{v_{k+1}^T r_{k+1}}{v_k^T r_k}$
21:    $p_{k+1} = z_{k+1} + \beta_k p_k$
22:    $q_{k+1} = v_{k+1} + \beta_k q_k$
23: end for
```

In Algorithm 2.1, the main computing tasks are solving the symmetric positive definite systems with coefficient matrices $A^T A$ and $\widehat{S}$. Suppose we have the QR factorization of the original matrix A; then $A^T A$ could be replaced by $R^T R$. Subsequently, the computing cost will be reduced greatly. The following theorem reveals the convergence result of RPCG and the effect of restrictive factorization and preconditioner on it.

THEOREM 2.1. *Let $A \in \mathbb{R}^{m \times n}$ be a full column rank matrix and $B \in \mathbb{R}^{k \times n}$ such that $\mathcal{A}$ in* (1.3) *is a nonsingular matrix and has the factorization* (2.2). *Let $\mathcal{M} = \mathcal{P}\mathcal{G}\mathcal{Q}$ and $\mathcal{G}$ be as defined in* (2.4). *If the RPCG method starts from an initial vector $x_0 \in \mathbb{R}^n$, then after l steps of iterations, it generates an approximation x_l to the solution x^* of the system of linear equations* (1.3), *which satisfies*

$$\|x_l - x^*\|_J \leq 2 \left(\frac{\sqrt{\kappa(\mathcal{G}^{-1}\mathcal{H})} - 1}{\sqrt{\kappa(\mathcal{G}^{-1}\mathcal{H})} + 1} \right)^l ||x_0 - x^*||_J,$$

where $\|v\|_J^2 = v^T J v$ defines the J norm of any $v \in \mathbb{R}^n$ because $J = \begin{bmatrix} I & B^T \end{bmatrix} \mathcal{Q}^T \mathcal{H} \mathcal{Q} \begin{bmatrix} I \\ B \end{bmatrix}$ is a symmetric positive definite matrix. $\kappa(\mathcal{G}^{-1}\mathcal{H})$ represents the Euclidean condition number of $\mathcal{G}^{-1}\mathcal{H}$.

Proof. The RPCG method for linear system (1.3) generates the sequence $\{y_l\}$, where $y_l = ({x_l}^T, (Bx_l)^T)^T$ iteratively. According to Theorem 2.1 in [2], the approximation of y_l to solution $y^* = ({x^*}^T, (Bx^*)^T)^T$ satisfies

$$||y_l - y^*||_{\mathcal{Q}^T\mathcal{H}\mathcal{Q}} \le 2\left(\frac{\sqrt{\kappa(\mathcal{G}^{-1}\mathcal{H})}-1}{\sqrt{\kappa(\mathcal{G}^{-1}\mathcal{H})}+1}\right)^l ||y_0 - y^*||_{\mathcal{Q}^T\mathcal{H}\mathcal{Q}}.$$

By straightforward computation, we have

$$\|y_l - y^*\|_{\mathcal{Q}^T\mathcal{H}\mathcal{Q}} = \left\|\begin{bmatrix} x_l \\ Bx_l \end{bmatrix} - \begin{bmatrix} x^* \\ Bx^* \end{bmatrix}\right\|_{\mathcal{Q}^T\mathcal{H}\mathcal{Q}} = \left\|\begin{bmatrix} I \\ B \end{bmatrix}(x_l - x^*)\right\|_{\mathcal{Q}^T\mathcal{H}\mathcal{Q}} = \|x_l - x^*\|_J.$$

Similarly,

$$\|y_0 - y^*\|_{\mathcal{Q}^T\mathcal{H}\mathcal{Q}} = \|x_0 - x^*\|_J.$$

Hence, we conclude this theorem. ∎

Remark 2.2. According to Theorem 2.1, the convergence rate of the RPCG method for solving the augmented system is decided by the condition number of

$$\mathcal{G}^{-1}\mathcal{H} = \mathcal{Q}^{-1}\mathcal{G}^{-1}\mathcal{H}\mathcal{Q} = \begin{bmatrix} I & 0 \\ 0 & \widehat{S}^{-1}S \end{bmatrix},$$

which is highly dependent on the approximation of $\widehat{S}^{-1}S$. If $\widehat{S} = S$, the RPCG method finds the exact solution in one step. Therefore, finding a good approximation of S is a crucial point for the whole method.

Remark 2.3. Let $e_l = y_l - y^*$ because $e_l^{(1)} = x_l - x^*$ and $e_l^{(2)} = Bx_l - Bx^*$. Then

$$\|x_l - x^*\|_{(A^TA)^{1/2}} \le 2\left(\frac{\sqrt{\kappa(\mathcal{G}^{-1}\mathcal{H})}-1}{\sqrt{\kappa(\mathcal{G}^{-1}\mathcal{H})}+1}\right)^l ||x_0 - x^*||_J.$$

It is clear that

$$\begin{aligned}
\|x_l - x^*\||_J^2 = \|\mathcal{H}^{1/2}\mathcal{Q}e_l\|^2 &= \left\|\begin{bmatrix} (A^TA)^{1/2} & 0 \\ 0 & S^{1/2} \end{bmatrix}\begin{bmatrix} I & (A^TA)^{-1}B^T \\ 0 & I \end{bmatrix}\begin{bmatrix} e_l^{(1)} \\ e_l^{(2)} \end{bmatrix}\right\|_2^2 \\
&= \left\|\begin{bmatrix} (A^TA)^{1/2} & (A^TA)^{-1/2}B^T \\ 0 & S^{1/2} \end{bmatrix}\begin{bmatrix} e_l^{(1)} \\ e_l^{(2)} \end{bmatrix}\right\|_2^2 \\
&= \|(A^TA)^{1/2}e_l^{(1)} + (A^TA)^{-1/2}B^Te_l^{(2)}\|^2 + \|S^{1/2}e_l^{(2)}\|^2 \\
&\ge \|(A^TA)^{1/2}e_l^{(1)}\|^2 - \|(A^TA)^{-1/2}B^Te_l^{(2)}\|^2 + \|S^{1/2}e_l^{(2)}\|^2 \\
&= \|(A^TA)^{1/2}e_l^{(1)}\|^2 + \|e_l^{(2)}\|_2^2 \\
&= \|x_l - x^*\|_{(A^TA)^{1/2}}^2 + \|x_l - x^*\|_{B^TB}^2.
\end{aligned}$$

The result of this remark could be obtained directly.

Because the approximation of the Schur complement plays an important role in the RPCG method, we study its approximation and updating techniques efficiently.

3. RPCG method for augmented least squares problems. We presently discuss the URPCG method for solving the following sequence of augmented least squares problems:

$$\underbrace{\begin{bmatrix} A^TA & \Delta {B_1}^T \\ -\Delta B_1 & I \end{bmatrix}}_{\mathcal{W}_1} \begin{bmatrix} x \\ \Delta B_1 x \end{bmatrix} = \begin{bmatrix} c_1 \\ \mathbf{0} \end{bmatrix},$$

$$\vdots$$

$$\underbrace{\begin{bmatrix} A^TA & \Delta {B_1}^T & \cdots & \Delta {B_N}^T \\ -\Delta B_1 & I & & \\ \vdots & & \ddots & \\ -\Delta B_N & & & I \end{bmatrix}}_{\mathcal{W}_N} \begin{bmatrix} x \\ \Delta B_1 x \\ \vdots \\ \Delta B_N x \end{bmatrix} = \begin{bmatrix} c_N \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{3.1}$$

where

$$c_i = c_0 + \sum_{j=1}^{i} \Delta {B_j}^T \Delta d_j = c_{i-1} + \Delta {B_i}^T \Delta d_i, \quad i = 1, \ldots, N,$$

with $c_0 = A^T b$. Let

$$B_i = \begin{bmatrix} \Delta B_1 \\ \vdots \\ \Delta B_i \end{bmatrix}.$$

Then the negative Schur complement in moment i is

$$\begin{aligned} S_i &= I + B_i (A^TA)^{-1} B_i^T \\ &= I + \begin{bmatrix} \Delta B_1 \\ \vdots \\ \Delta B_i \end{bmatrix} (A^TA)^{-1} \begin{bmatrix} \Delta B_1^T \cdots & \Delta B_i^T \end{bmatrix}. \end{aligned}$$

Assume that we have the decomposition of the normal matrix as

$$A^TA = R_a^T R_a,$$

where R_a is an upper triangular matrix, which might be obtained by performing QR factorization on A. Let

$$\Delta E_i = R_a^{-T} \Delta B_i^T \quad \text{and} \quad E_i = R_a^{-T} B_i^T;$$

then the negative Schur complement

$$\begin{aligned} S_{i+1} &= I + E_{i+1}^T E_{i+1} \\ &= \begin{bmatrix} S_i & E_i^T \Delta E_{i+1} \\ \Delta {E_{i+1}}^T E_i & I + \Delta {E_{i+1}}^T \Delta E_{i+1} \end{bmatrix}, \end{aligned} \tag{3.2}$$

and $E_{i+1} = \left[\begin{smallmatrix} E_i & \Delta E_{i+1} \end{smallmatrix}\right]$ could be updated recurrently. Let $S_i = \bar{R}^T \bar{R}$ be obtained in hand; then

$$S_{i+1} = R^T R = \begin{bmatrix} \bar{R}^T & 0 \\ R_{12}^T & R_{22}^T \end{bmatrix} \begin{bmatrix} \bar{R} & R_{12} \\ 0 & R_{22} \end{bmatrix}.$$

Because there are only a few rows contained in ΔB_{i+1}, it is not a consuming task to solve

$$R_{12} = \bar{R}^{-T} E_i^T \Delta E_{i+1}.$$

Subsequently, R_{22} could be computed by Cholesky decomposition given as

$$R_{22}^T R_{22} = I + \Delta E_{i+1}^T \Delta E_{i+1} - R_{12}^T R_{12}.$$

Remark 3.1. According to Remark 2.2, if the Schur complement S_{i+1} is updated exactly, the RPCG method needs only one step to find the solution of the augmented saddle point problem in exact arithmetic.

Actually, it is not necessary to compute the Schur complements S_{i+1} exactly. In practical implementations, we recommend two ways to generate approximations $\widehat{S}_{i+1}$ to enhance the computing efficiency.

Strategy A. Let $\widehat{S}_i = \bar{R}_i^T \bar{R}_i$ be the approximation of S_i, and let $\hat{E}_i$ be the approximation of E_i. Then

- $\Delta\widehat{B}_{i+1} = \text{sparse}(\Delta B_{i+1})$;
- $\Delta\bar{E}_{i+1} = R_a^{-T} \Delta\widehat{B}_{i+1}^T$;
- $\Delta\widehat{E}_{i+1} = \text{sparse}(\Delta\bar{E}_{i+1})$;
- $\widehat{R}_{12} = \bar{R}_i^{-T} \hat{E}_i^T \Delta\widehat{E}_{i+1}$;
- $\widehat{R}_{22} = \text{ichol}(I + \Delta\widehat{E}_{i+1}^T \Delta\widehat{E}_{i+1} - \widehat{R}_{12}^T \widehat{R}_{12})$.

Here, ichol$(\cdot)$ refers to incomplete Cholesky decomposition [12]. The approximation of Schur complement could be written as

$$\widehat{S} = \bar{R}_{i+1}^T \bar{R}_{i+1} = \begin{bmatrix} \bar{R}_i^T & 0 \\ \widehat{R}_{12}^T & \widehat{R}_{22}^T \end{bmatrix} \begin{bmatrix} \bar{R}_i & \widehat{R}_{12} \\ 0 & \widehat{R}_{22} \end{bmatrix}.$$

$\widehat{E}_{i+1}$ is merged by $\widehat{E}_{i+1} = \begin{bmatrix} \widehat{E}_i & \Delta\widehat{E}_{i+1} \end{bmatrix}$. In lines 4 and 15 of Algorithm 2.1, we solve the lower and upper triangular systems in succession.

In what follows, we introduce a block diagonal approximation of Schur complement in the URPCG method.

Strategy B. Let $\widehat{S}_i = \bar{R}_i^T \bar{R}_i$ be the block diagonal approximation of S_i. Then we approximate S_{i+1} in (3.2) by its diagonal blocks. Let ΔR_{i+1} be the incomplete Cholesky factor of $I + \Delta E_{i+1}{}^T \Delta E_{i+1}$; then

$$\widehat{S}_{i+1} = \begin{bmatrix} \widehat{S}_i & O \\ O & \Delta R_{i+1}^T \Delta R_{i+1} \end{bmatrix} = \begin{bmatrix} \bar{R}_i & 0 \\ 0 & \Delta R_{i+1} \end{bmatrix}^T \begin{bmatrix} \bar{R}_i & 0 \\ 0 & \Delta R_{i+1} \end{bmatrix}.$$

In lines 4 and 15 of Algorithm 2.1, we solve two block diagonal systems in succession.

Now, we analyze the eigenvalue distribution of the preconditioned Schur complement matrix $\hat{S}_{i+1}^{-1} S_{i+1}$. For the sake of analysis, we assume that ΔR_i is achieved by complete Cholesky factorization exactly. In the first augmented moment,

$$S_2 = \begin{bmatrix} S_1 & E_1^T \Delta E_2 \\ \Delta E_2^T E_1 & \Delta S_2 \end{bmatrix} \quad \text{and} \quad \hat{S}_2 = \begin{bmatrix} S_1 & 0 \\ 0 & \Delta S_2 \end{bmatrix},$$

in which

$$S_1 = I + E_1^T E_1 \quad \text{and} \quad \Delta S_2 = I + \Delta E_2^T \Delta E_2.$$

Let λ_2 be an eigenvalue of $\hat{S}_2^{-1}S_2$ and $x = [u; v]$ be the corresponding eigenvector, i.e.,

$$S_2 x = \lambda_2 \hat{S}_2 x.$$

Therefore,

$$\begin{bmatrix} 0 & E_1^T \Delta E_2 \\ \Delta E_2^T E_1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = (\lambda_2 - 1) \begin{bmatrix} S_1 & 0 \\ 0 & \Delta S_2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}.$$

We get

$$\Delta E_2^T E_1 S_1^{-1} E_1^T \Delta E_2 v = (\lambda_2 - 1)^2 \Delta S_2 v$$

by direct calculation. It holds that

$$\begin{aligned}(\lambda_2 - 1)^2 &= \frac{v^T \Delta E_2^T E_1 S_1^{-1} E_1^T \Delta E_2 v}{v^T \Delta S_2 v} = \frac{v^T \Delta E_2^T E_1 S_1^{-1} E_1^T \Delta E_2 v}{v^T \Delta E_2^T \Delta E_2 v} \frac{v^T \Delta E_2^T \Delta E_2 v}{v^T \Delta S_2 v} \\ &= \frac{w^T E_1 S_1^{-1} E_1^T w}{w^T w} \frac{v^T \Delta E_2^T \Delta E_2 v}{v^T \Delta S_2 v},\end{aligned}$$

where $\omega = \Delta E_2 v$. We denote the minimal and maximal singular values of E_1 and ΔE_2 by $\sigma_{\min}(E_1)$, $\sigma_{\max}(E_1)$ and $\sigma_{\min}(\Delta E_2)$, $\sigma_{\max}(\Delta E_2)$ respectively. Then

$$\frac{\sigma_{\min}^2(E_1)}{1 + \sigma_{\min}^2(E_1)} \leq \frac{w^T E_1 S_1^{-1} E_1^T w}{w^T w} \leq \frac{\sigma_{\max}^2(E_1)}{1 + \sigma_{\max}^2(E_1)}$$

and

$$\frac{\sigma_{\min}^2(\Delta E_2)}{1 + \sigma_{\min}^2(\Delta E_2)} \leq \frac{v^T \Delta E_2^T \Delta E_2 v}{v^T \Delta S_2 v} \leq \frac{\sigma_{\max}^2(\Delta E_2)}{1 + \sigma_{\max}^2(\Delta E_2)}.$$

Therefore, we get

$$0 < \lambda_2 < 2. \tag{3.3}$$

In the $(i+1)$th augmented saddle point system,

$$\hat{S}_{i+1} = \begin{bmatrix} \hat{S}_i & 0 \\ 0 & \Delta S_{i+1} \end{bmatrix}$$

is a block diagonal approximation of

$$S_{i+1} = \begin{bmatrix} S_i & E_i^T \Delta E_{i+1} \\ \Delta E_{i+1}^T E_i & \Delta S_{i+1} \end{bmatrix}, \tag{3.4}$$

where $\Delta S_{i+1} = I + \Delta E_{i+1}^T \Delta E_{i+1}$ and $\hat{S}_i$ is the block diagonal approximation of S_i in the previous step. Let λ_{i+1} be an eigenvalue of $\hat{S}_{i+1}^{-1} S_{i+1}$, and let $x = [u; v]$ be the corresponding eigenvector. Therefore,

$$\begin{bmatrix} S_i & E_i^T \Delta E_{i+1} \\ \Delta E_{i+1}^T E_i & \Delta S_{i+1} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda_{i+1} \begin{bmatrix} \hat{S}_i & 0 \\ 0 & \Delta S_{i+1} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}.$$

When $\lambda_{i+1} \neq 1$, it holds that

$$v = \frac{1}{\lambda_{i+1} - 1} \Delta S_{i+1}^{-1} \Delta E_{i+1}^T E_i u$$

and

$$S_i u + \frac{1}{\lambda_{i+1}-1} E_i^T \Delta E_{i+1} \Delta S_{i+1}^{-1} \Delta E_{i+1}^T E_i u = \lambda_{i+1} \hat{S}_i u.$$

Multiplying each side by u^T and reformulating the above equality, we have

$$(\lambda_{i+1}-1) u^T S_i u + u^T E_i^T \Delta E_{i+1} \Delta S_{i+1}^{-1} \Delta E_{i+1}^T E_i u = \lambda_{i+1}(\lambda_{i+1}-1) u^T \hat{S}_i u.$$

Denote

$$\begin{aligned} b_i &= \frac{u^T E_i^T \Delta E_{i+1} \Delta S_{i+1}^{-1} \Delta E_{i+1}^T E_i u}{u^T S_i u} \\ &= \frac{u^T E_i^T E_i u}{u^T (I + E_i^T E_i) u} \frac{u^T E_i^T \Delta E_{i+1} \Delta S_{i+1}^{-1} \Delta E_{i+1}^T E_i u}{u^T E_i^T E_i u} \end{aligned}$$

and $\tau_i = \frac{u^T S_i u}{u^T \hat{S}_i u}$. The quadratic equation of λ_{i+1} is simplified to

$$\lambda_{i+1}^2 - (1+\tau_i)\lambda_{i+1} + (1-b_i)\tau_i = 0.$$

Notice that $0 < b_i < 1$, and we have

$$\lambda_{i+1}(\lambda_{i+1} - 1 - \tau_i) < 0.$$

Denote the maximum eigenvalue of $\hat{S}_i^{-1} S_i$ by $\lambda_i^{\max}$. It is clear that

$$0 < \tau_i \le \lambda_i^{\max}.$$

Therefore, the eigenvalues of $\hat{S}_{i+1}^{-1} S_{i+1}$ are in the interval

$$0 < \lambda_{i+1} < 1 + \lambda_i^{\max}. \tag{3.5}$$

According to (3.3) and (3.5), we draw a conclusion.

Remark 3.2. Consider the RPCG method for solving the $(i+1)$th augmented least squares problem. If the Schur complement S_{i+1} in (3.2) is approximated by its block diagonal matrix $\hat{S}_{i+1}$ in (3.4), then the eigenvalues of the preconditioned matrix with restrictive preconditioner (2.4) are in the interval

$$0 < \lambda < i + 2.$$

The upper bound of the eigenvalues increases by 1 with every data augmentation. The lower bound depends on E_{i+1}, or rather B_{i+1}.

By combining Algorithm 2.1 with Strategy A or Strategy B, we achieve the RPCG methods for solving the augmented least squares problems (3.1). The preconditioner generated with Strategy A is more precise than the preconditioner generated with Strategy B. However, the latter is much easier to be computed and operated.

4. Numerical results.

In this section, we denote the URPCG method using Strategies A and B by URPCG1 and URPCG2, respectively. We compare the performance of the proposed methods using PCGLS and UPCGLS methods by testing two groups of problems. In the PCGLS method, the preconditioner $R_a^T R_a$ does not change for a series of augmented least squares problems. The UPCGLS method refers to the method in [13]. All computations start from the initial vector $x_0 = 0$. x_k is a satisfied solution of any modified least squares problem if the relative residual (RES) of the normal equation (1.4), defined by $RES = \|c - Nx_k\|/\|c\|$, is reduced to no more than

TABLE 1
Experimental data from SuiteSparse matrix collection.

Name	m	n	δ	N	γ (%)
`c-55`	22,946	16,390	100	90	39.22
`G2_circuit`	120,081	30,021	100	300	24.98
`helm3d01`	25,780	6,446	20	300	23.27
`jnlbrng1`	28,000	13,334	120	90	38.57
`Kemelmacher`	19,916	9,693	50	120	30.13
`mesh_deform`	163,816	9,393	200	300	36.63
`psse0`	16,033	11,028	10	900	56.13
`psse2`	18,612	11,028	10	900	48.36
`vsp_barth5`	27,380	6,443	40	60	9.86
`wathen120`	29,152	7,289	20	360	24.70

10^{-6}. The approximation solution of the previous problem is used as the initial vector for the subsequent augmented least squares problem. All experiments are carried out in MATLAB R2017b on a personal computer with an Intel Core i5 CPU with 8 GB of installed memory RAM and Windows operating system (Windows 10).

In practical implementations, QR factorization of A is performed after a column approximate minimum degree permutation with MATLAB function "colamd." The Cholesky factorization of a matrix is performed after a symmetric approximate minimum degree permutation with MATLAB function "symamd." The permutations offer a sparse structure to enhance the computing efficiency.

The first group of problems was generated artificially. The matrices were from the SuiteSparse matrix collection [8]. We divided a matrix into several parts by rows. The leading part was counted as the initial matrix A. The remaining submatrix B was partitioned into N parts denoted by ΔB_i, where $i = 1, 2, \ldots, N$. Each ΔB_i consisted of δ rows. The right-hand side vector was generated randomly with the same partition as the coefficient matrices. In Table 1, we list the information of each matrix, including the number of rows (m) and columns (n), number of added blocks (N), number of rows in each added block (δ), and number of added rows as a percentage of the number of original rows (γ).

In Table 2, we report the numerical behavior of different methods for solving augmented least squares problems. "$\overline{\text{itn}}$" and "time" refer to average iteration steps and total computing time in seconds, respectively, for solving a series of augmented least squares problem. From the table we see that URPCG1 and URPCG2 vastly outperform UPCGLS and PCGLS in terms of both iteration count and computing time. The numerical efficiencies of URPCG1 and URPCG2 are comparable. The iteration counts of URPCG1 are fewer than that of URPCG2 because Strategy A adopts a closer approximation to Schur complement than Strategy B. In each iteration, URPCG2 takes less computing time than URPCG1 because of its sparser structure of approximation to the Schur complement. UPCGLS is a nearly direct method, especially when the rows of B are not large in number; hence, the Schur complement could be approximated exactly. However, when a large number of rows are added, the incomplete Cholesky factorization of Schur complement is difficult to compute. Moreover, the solution of a saddle point system is involved in every iteration of UPCGLS. This means that the CG method changes the preconditioner in every iteration for solving normal equations. The flexible preconditioning technique is not stable for the CG method. Hence, UPCGLS fails in solving the problem with matrix `c-55` and is not as efficient as the URPCG method for solving a series of augmented least squares problems. The PCGLS method performs fixed preconditioning on a series of

TABLE 2
Numerical behavior of the tested algorithms.

Name	PCGLS $\overline{\text{itn}}$/time	UPCGLS $\overline{\text{itn}}$/time	URPCG1 $\overline{\text{itn}}$/time	URPCG2 $\overline{\text{itn}}$/time
c-55	195 / 43,714.07	≀	69/360.01	76/375.67
G2_circuit	17/8.14	10/406.37	1/5.74	3/6.62
helm3d01	23/177.98	23/200.15	15/42.18	15/43.71
jnlbrng1	5/1.66	5/17.57	2/1.27	3/1.32
Kemelmacher	84/217.81	467 / 1,212.01	6/18.59	14/21.35
mesh_deform	46/341.37	2956 /22,926.58	13/69.31	18/79.40
psse0	3/20.58	3/124.73	2/16.43	2/12.28
psse2	9/52.79	12/171.11	2/19.06	3/14.92
vsp_barth5	10/4.74	11/6.74	2/1.93	5/2.37
wathen120	14/50.54	14/87.42	3/14.73	6/15.12

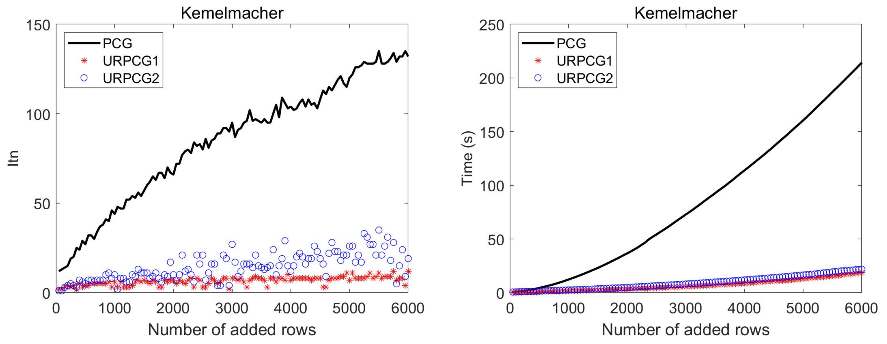

FIG. 1. *Comparison based on iteration number and computing time using the Kemelmacher matrix.*

normal systems. As the number of added rows increases, the preconditioner loses its efficiency gradually. The number of iterations of the URPCG-type methods are only 1/2 to 1/5 or even less than that of the PCGLS method. The computing time of the URPCG method is much less than that of the PCGLS method. Strategies A and B play a key role. In Figure 1, we consider the "Kemelmacher" matrix as an example to show the variation of iteration counter and computing time with increasing number of added rows. The curves show that the iteration counters of the URPCG method, especially UPRCG1, are almost independent of the increasing number of added rows. Therefore, the relationship between the growth of computing time and the number of added rows is nearly linear.

Now, we utilize the URPCG methods to solve the real-time multifactor model. We collect minute-level data of the SSE 50 index constituent stocks as the data source. The stock pool is $\mathbb{S} = \{s_1, s_2, \ldots, s_{50}\}$, where $i = 1, 2, \ldots, 50$. s_i refers to the ith component stock. In this experiment, we test the data of 8 groups arising from different periods. In Table 3, *Basic days* and *Year* refer to the initial trading days and years, respectively. *Updating day* refers to the data-updating day. There are 240 minutes in a trading day and therefore 12,000 new added rows per day. In four basic trading days, we delete some invalid rows and m rows remain in each test. In the initial model, the added data are achieved in real time. Every minute, we get 50 new rows from 50 stocks. Subsequently, we receive 7,500 new rows in 150 minutes and generate matrix B gradually. The percentage of newly added data to the original data

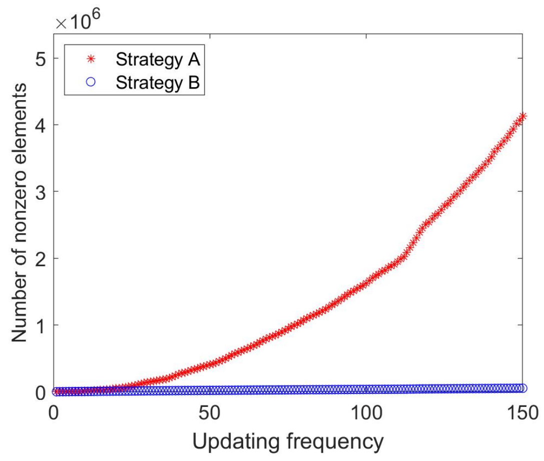

FIG. 2. *Number of nonzero elements in Cholesky factor of Schur complement.*

TABLE 3
Multifactor model.

Year	Basic days	Updating day	D	m	γ (%)
2010	06/24 $\sim$ 06/29	06/30	27	38,492	19.48
2011	08/04 $\sim$ 08/09	08/10	29	40,396	18.57
2012	04/26 $\sim$ 05/03	05/04	29	39,278	19.09
2013	07/23 $\sim$ 07/26	07/29	28	41,314	18.15
2014	02/27 $\sim$ 03/04	03/05	28	40,733	18.41
2015	09/24 $\sim$ 09/29	09/30	28	39,597	18.94
2016	07/25 $\sim$ 07/28	07/29	30	40,128	18.69
2017	04/24 $\sim$ 04/27	04/28	28	36,665	20.46

is denoted by γ. Assume we just fixed the multifactor model at time i, $i = 1, \ldots, 149$, and have the current $\bar{R}_i$ in hand. To generate the upper triangular factor $\bar{R}_{i+1}$ of $\hat{S}_{i+1}$, we compute $\hat{R}_{12} \in \mathbb{R}^{(50\cdot i)\times 50}$ and $\hat{R}_{22} \in \mathbb{R}^{50\times 50}$ in Strategy A or $\Delta R_{i+1} \in \mathbb{R}^{50\times 50}$ in Strategy B. Meanwhile, benefiting from the sparsity techniques and incomplete Cholesky decomposition, $\bar{R}_{i+1}$ in both Strategy A and Strategy B are quite sparse and can be computed and stored easily in every multifactor model. In Figure 2, we can see that the number of nonzero elements in $\bar{R}_i$ increases mildly with the matrix enlargement. $\bar{R}_i$ updated by Strategy B is much sparser and easier to be operated.

In Table 4, we list the average iteration counters, average computing time, and total computing time. URPCG1 and URPCG2 solve every augmented least squares problem within 3 seconds. The computing time is much less than the data-updating time (1 minute). In Figure 3, we plot the computing time and iteration number of every updated least squares problem. This shows that URPCG1 finds a satisfactory solution within 10 steps, while URPCG2 finds a satisfactory solution in no more than 14 steps. The computing time of each augmented least squares problem does not increase with increasing data. Both tested methods meet the requirements of real-time computing; however, URPCG1 is more efficient than URPCG2.

5. Remarks and conclusions.

The RPCG method is effective for solving saddle point linear systems. Based on the framework of this method, we propose URPCG methods for solving a series of augmented least squares problems, which are equivalent to saddle point systems. The crucial points are performing restrictive factorization

TABLE 4
Numerical results of the multifactor model.

	URPCG1	URPCG2
Year	$\overline{\text{itn}}/\bar{t}/\text{time}$	$\overline{\text{itn}}/\bar{t}/\text{time}$
2010	6.05 / 1.85 / 276.85	13.64 / 2.26 / 339.18
2011	5.07 / 1.51 / 226.30	10.13 / 1.82 / 272.62
2012	4.91 / 1.51 / 226.17	9.11 / 1.81 / 272.07
2013	4.52 / 1.87 / 280.60	7.26 / 2.06 / 309.06
2014	4.67 / 1.47 / 221.13	8.31 / 1.75 / 261.94
2015	4.34 / 1.63 / 244.39	8.73 / 1.92 / 288.53
2016	5.39 / 1.80 / 270.72	13.35 / 2.41 / 361.40
2017	4.56 / 1.58 / 236.55	6.99 / 1.76 / 263.72

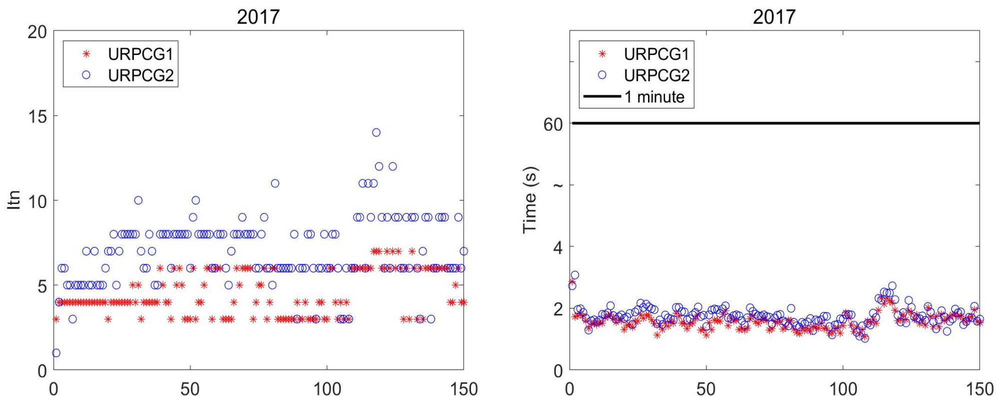

FIG. 3. *Comparison based on iteration number and computing time.*

on the augmented saddle point system and approximating the updated Schur complement efficiently by considering the special structure and sparsity of matrices. The URPCG methods show high performance in solving real-time multifactor models.

Acknowledgment. The authors would like to thank Professor Martin Stoll and anonymous referees for their careful reading of the manuscript and helpful suggestions.

REFERENCES

[1] R. Andrew and N. Dingle, *Implementing QR factorization updating algorithms on GPUs*, Parallel Comput., 40 (2014), pp. 161–172.
[2] Z.-Z. Bai and G.-Q. Li, *Restrictively preconditioned conjugate gradient methods for systems of linear equations*, IMA J. Numer. Anal., 23 (2003), pp. 561–580.
[3] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996, https://doi.org/10.1137/1.9781611971484.
[4] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comp., 50 (1988), pp. 1–17.
[5] T. A. Davis and W. W. Hager, *Modifying a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 606–627, https://doi.org/10.1137/S0895479897321076.
[6] T. A. Davis and W. W. Hager, *Multiple-rank modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 997–1013, https://doi.org/10.1137/S0895479899357346.
[7] T. A. Davis and W. W. Hager, *Row modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 621–639, https://doi.org/10.1137/S089547980343641x.
[8] T. A. Davis and Y. F. Hu, *The University of Florida sparse matrix collection*, ACM Trans.

Math. Software, 38 (2011), pp. 1–25, https://doi.org/110.1145/2049662.2049663.

[9] H. S. Dollar, N. I. M. Gould, M. Stoll, and A. J. Wathen, *Preconditioning saddle-point systems with applications in optimization*, SIAM J. Sci. Comput., 32 (2010), pp. 249–270, https://doi.org/10.1137/080727129.

[10] S. Hammarling and C. Lucas, *Updating the QR Factorization and the Least Squares Problem*, MIMS EPrint 2008.111, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2008.

[11] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[12] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[13] J. Marin, J. Mas, D. Guerrero, and K. Hayami, *Updating preconditioners for modified least squares problems*, Numer. Algorithms, 75 (2017), pp. 491–508, https://doi.org/10.1007/s11075-017-0315-z.

[14] O. Olsson and T. Ivarsson, *Using the QR Factorization to Swiftly Update Least Squares Problems*, Master's Theses in Mathematical Sciences, 2014.

[15] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71, https://doi.org/10.1145/355984.355989.

[16] S. Ross, *The arbitrage theory of capital asset pricing*, J. Econom. Theory, 13 (1976), pp. 341–360.

[17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 1996, https://doi.org/10.1137/1.9780898718003.

[18] M. Stoll and A. Wathen, *Combination preconditioning and the Bramble–Pasciak+ preconditioner*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 582–608, https://doi.org/10.1137/070688961.