Computational and Applied Mathematics (2020) 39:43
https://doi.org/10.1007/s40314-020-1089-9

Limited memory BFGS algorithm for the matrix approximation problem in Frobenius norm

Chungen Shen[1] · Changxing Fan[1] · Yunlong Wang[1] · Wenjuan Xue[2]

Received: 11 February 2019 / Revised: 11 November 2019 / Accepted: 15 November 2019 /
Published online: 6 February 2020

Abstract
This paper proposes an L-BFGS algorithm based on the active set technique to solve the matrix approximation problem: given a symmetric matrix, find a nearest approximation matrix in the sense of Frobenius norm to make it satisfy some linear equalities, inequalities and a positive semidefinite constraint. The problem is a convex optimization problem whose dual problem is a nonlinear convex optimization problem with non-negative constraints. Under the Slater constraint qualification, it has zero duality gap with the dual problem. To handle large-scale dual problem, we make use of the active set technique to estimate the active constraints, and then the L-BFGS method is used to accelerate free variables. The global convergence of the proposed algorithm is established under certain conditions. Finally, we conduct some preliminary numerical experiments, and compare the L-BFGS method with the inexact smoothing Newton method, the projected BFGS method, the alternating direction method and the two-metric projection method based on the L-BFGS. The numerical results show that our algorithm has some advantages in terms of CPU time when a large number of linear constraints are involved.

Keywords Matrix approximation · Active set · Dual problem · L-BFGS · Global convergence

Mathematics Subject Classification 65K05 · 95C55 · 90C30

Communicated by Joerg Fliege.

This research is supported by National Natural Science Foundation of China (No. 11601318).

✉ Chungen Shen
shenchungen@gmail.com

1 College of Science, University of Shanghai for Science and Technology, Shanghai 200093, China

2 School of Mathematics and Physics, Shanghai University of Electric Power, Shanghai 200090, China

1 Introduction

In this paper, we are concerned with the following matrix approximation problem

$$\begin{cases} \min & \frac{1}{2}\|X - C\|_F^2 \\ \text{s.t.} & \langle A_i, X\rangle = b_i, \quad i \in \mathcal{E}, \\ & \langle A_i, X\rangle \geq b_i, \quad i \in \mathcal{I}, \\ & X \in \mathcal{S}_+^n, \end{cases} \tag{1.1}$$

where $\mathcal{E} = \{1, \ldots, p\}$, $\mathcal{I} = \{p+1, \ldots, m\}$, $\mathcal{S}_+^n$ denotes the cone of positive semidefinite matrices in the space $\mathcal{S}^n$ of $n \times n$ symmetric matrices, $C \in \mathcal{S}^n$, $A_i \in \mathcal{S}^n$, $i = 1, \ldots, m$, $b = (b_1, \ldots, b_m)^T \in \mathbb{R}^m$, and $\langle A, B\rangle := \operatorname{tr}(A^T B)$ with $A, B \in \mathcal{S}^n$. This special case of this problem is the nearest correlation matrix problem

$$\begin{cases} \min & \frac{1}{2}\|X - C\|_F^2 \\ \text{s.t.} & X_{ii} = 1, \quad i = 1, \ldots, n \\ & X \in \mathcal{S}_+^n, \end{cases} \tag{1.2}$$

which often arises in the field of statistics, finance and insurance/reinsurance industries (Qi and Sun 2010; Rebonato and Jäckel 1999; Werner and Schöttle 2007). Higham (2002) formulated the problem (1.2) and applied Dysktra's alternating projection algorithm to solve it. Later, Qi and Sun (2006) proposed a semi-smooth Newton method for solving the problem (1.2), and Borsdorf and Higham (2010) refined the semi-smooth Newton method with an efficient preconditioner. Numerical experiments (Qi and Sun 2006; Borsdorf and Higham 2010) demonstrate that the semi-Newton method outperforms some other first-order methods (say the alternating projection method).

In financial industries, scenario analysis in stress testing requires to restrict some correlation coefficients to stay with their confidence intervals. But it might result in indefinite data matrices. Hence, a nearest correlation matrix for a given "bad" data matrix is sought such that some equalities and inequalities are satisfied, which is the optimization problem described in (1.1).

Problem (1.1) is a convex optimization problem which has been investigated by some researchers (such as Gao and Sun 2009; He et al. 2011). If the feasible set of problem (1.1) is nonempty, then it has a unique optimizer $\bar{X}$. Define a linear mapping $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$ by

$$\mathcal{A}(X) = \begin{pmatrix} \langle A_1, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{pmatrix}, \quad X \in \mathcal{S}^n. \tag{1.3}$$

The Lagrangian dual (Malick 2004; Boyd and Xiao 2005; Gao and Sun 2009) of (1.1) can be written as follows

$$\begin{cases} \min & \phi(y) := \frac{1}{2}\|(C + \mathcal{A}^* y)_+\|^2 - b^T y - \frac{1}{2}\|C\|^2 \\ \text{s.t.} & y_i \geq 0, \ i \in \mathcal{I}, \end{cases} \tag{1.4}$$

where X_+ denotes the metric projection of X onto $\mathcal{S}_+^n$, and $\mathcal{A}^* : \mathbb{R}^m \to \mathcal{S}^n$ is the adjoint of $\mathcal{A}$ defined by $\mathcal{A}^* y = \sum_{i=1}^m y_i A_i$. Here the metric projection X_+ is defined as the unique solution to

$$\begin{cases} \min & \|X - Y\|^2 \\ \text{s.t.} & Y \in \mathcal{S}_+^n. \end{cases}$$

If X has the spectral decomposition $X = P\Lambda P^T$ with $\Lambda := \operatorname{diag}(\lambda_1, \dots, \lambda_n))$ where $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of X and P is a corresponding orthogonal matrix of orthonormal eigenvectors of X, then the metric projection (see Malick 2004, Theorem 2.4, Schwertman and Allen 1979)

$$X_+ = P\operatorname{diag}(\max(0, \lambda_1), \dots, \max(0, \lambda_n))P^T.$$

The dual objective function $\phi(y)$ is convex and continuously differentiable (see Zarantonello 1971), and its gradient is given by

$$\nabla\phi(y) = \mathcal{A}(C + \mathcal{A}^* y)_+ - b, \;\; y \in \mathbb{R}^m. \tag{1.5}$$

Under the Slater constraint qualification, it has zero duality gap between the primal problem (1.1) and the dual problem (1.4), which implies that one can solve the dual problem instead of the primal one.

Malick (2004) presented a BFGS method to deal with the problem (1.1) with equality constraints only, and Boyd and Xiao (2005) proposed a projected gradient algorithm for the problem (1.1) with both equalities and inequalities. Later, (Gao and Sun 2009) proposed a quadratically convergent inexact smoothing Newton method which is very efficient for the problem with a medium amount of constraints. We remark that all three methods (Boyd and Xiao 2005; Malick 2004; Gao and Sun 2009) are applied to the dual problem (1.4) instead of the primal problem (1.1). However, the BFGS method in Malick (2004) cannot deal with the problem with a large m due to large memory requirement, the projected gradient algorithm in Boyd and Xiao (2005) converges slowly although it can tackle the problem with a number of constraints, and the inexact smoothing Newton method (Gao and Sun 2009) is restrained by the size of the Newton system of linear equations. As opposed to Boyd and Xiao (2005), Malick (2004), Gao and Sun (2009), He et al. (2011) introduced an auxiliary matrix variable, built the corresponding augmented Lagrangian function and then applied the alternating direction methods (Gabay 1983; Gabay and Mercier 1976; Ye and Yuan 2007) to solve the problem (1.1). Their numerical evidence shows that the proposed alternating direction method has the superiority for large-scale problems. Along the dual framework, Li and Li (2011) proposed a projected semismooth Newton method for solving the problem (1.1). Very recently, Sun and Vandenberghe (2015) considered to use decomposition methods for matrix nearness problems with some sparsity pattern.

As we all know, the L-BFGS method (Nocedal and Wright 2006, Chapter 7) is one of the most effective and widely used methods in the community of optimization. The main advantage is that the L-BFGS approach does not require calculating or forming a full Hessian matrix, which might be too expensive or prohibitive for large scale problems. The optimization problem with box constraints has been investigated in nonlinear programming communities for many years, for instance (Byrd et al. 1995; Cristofari et al. 2017; Hager and Zhang 2006; Kelley 1999; Lin and Moré 1999; Ni and Yuan 1997; Rahpeymaii et al. 2016; Xiao and Wei 2007; Xiao and Li 2008; Yuan and Lu 2011), to name only a few. Quite a few of them are based on the L-BFGS technique. Byrd et al. (1995) first computed a generalized Cauchy point for detecting the active set and then the L-BFGS technique is used to accelerate those free variables. It needs to minimize a quadratic subproblem over a subspace to obtain the search direction, which might be time-consuming for large m. Ni and Yuan (1997) presented a subspace limited memory quasi-Newton method which does not require to compute the generalized Cauchy point and the search direction can be given explicitly instead. Similarly, Kelley (1999), Rahpeymaii et al. (2016), Xiao and Wei (2007), Xiao and Li (2008), Yuan and Lu (2011) also proposed and analyzed active-set type of L-BFGS methods which are based on different techniques for identifying active/inactive sets. In this paper, we focus on the problem (1.1) with a large number of constraints. Based on the framework given in Ni and Yuan (1997), we present an improved active-set detection technique which can deal with inequality constraints effectively, and then employ the L-BFGS approximation to accelerate free variables, where, of course, the search direction's formula is given explicitly. Unlike (Byrd et al. 1995; Kelley 1999), the estimated active set defined in our algorithm meets the optimality condition of the dual problem very well. Compared with Ni and Yuan (1997), our method can reasonably identify free variables close to the boundary, and the L-BFGS method is applied to accelerate them as much as possible while the global convergence is ensured. More differences and connections with the algorithms in related references will be given in Remark 2.5 in Sect. 2.

The remaining parts of this paper are organized as follows. In the next section, we review the L-BFGS method, describe the estimated active set and the search direction, and state our main algorithm. In Sect. 3, the global convergence is proved under some conditions. In Sect. 4, some numerical results and comparison on random data and real data are illustrated. Some conclusions are made in the final section.

2 Active-set L-BFGS algorithm

Let y^k denote the current iterate. For simplicity, we denote $\phi(y^k)$ and $\nabla\phi(y^k)$ by ϕ^k and g^k respectively. To estimate the active set of inequalities in (1.4), we define the KKT error of this problem as $r_g^k = \|y^k - P_\Omega(y^k - g^k)\|$, where $\Omega = \{y \in \mathbb{R}^m | y_i \geq 0, i \in \mathcal{I}\}$ is the feasible region of (1.4) and $P_\Omega(y^k)$ is the projection onto Ω. It is straightforward that

$$P_\Omega(y^k) = \begin{cases} y_i^k, & i \in \mathcal{E}, \\ 0, & i \in \mathcal{I} \text{ with } y_i^k < 0, \\ y_i^k, & i \in \mathcal{I} \text{ with } y_i^k \geq 0. \end{cases}$$

If y^k is a KKT point of the problem (1.4), then $r_g^k = 0$ which is equivalent to the KKT condition

$$\begin{cases} g_i^k = 0, & i \in \mathcal{E}, \\ g_j^k \geq 0, g_j^k y_j^k = 0, y_j^k \geq 0, & j \in \mathcal{I}, \end{cases} \tag{2.1}$$

of the problem (1.4). We will see later that r_g^k can help us determine the active set of inequalities.

2.1 L-BFGS procedure

We recall that the BFGS method (Nocedal and Wright 2006) is famous for solving the unconstrained minimization problem

$$\min \quad \phi(y). \tag{2.2}$$

At iterate y^k, the iterative formula of the BFGS method is given as

$$y^{k+1} = y^k - \alpha^k \bar{H}^k g^k, \quad k = 0, 1, 2, \ldots,$$

where α^k is a stepsize, $g^k = \nabla\phi(y^k)$, and $\bar{H}_k$ is the inverse Hessian approximation updated by means of the formula

$$\bar{H}^{k+1} = (V^k)^T \bar{H}^k V^k + \rho^k s^k (s^k)^T, \tag{2.3}$$

where $\rho^k = 1/(w^k)^T s^k$, $V^k = I - \rho^k w^k (s^k)^T$, and $s^k = y^{k+1} - y^k$, $w^k = g^{k+1} - g^k$ (see Nocedal and Wright (2006), Chapter 7). Note that the inverse approximation $\bar{H}^{k+1}$ is obtained by updating $\bar{H}^k$ using the pair $\{s^i, w^i\}$, and $\bar{H}_k$ has to be stored at each iteration. However, it is memory-consuming or even prohibitive for large problems.

The L-BFGS method (see Liu and Nocedal 1989; Nocedal and Wright 2006) is a variant of the BFGS method and works well for large-scale unconstrained optimization problems. The matrix $\bar{H}^k$ is formed implicitly by the l most recent vectors pairs $\{s^i, w^i\}$. At iteration k, using an initial Hessian approximation $(\bar{H}^k)^0$ and the l most recent vector pairs $\{s^i, w^i\}$, $i = k-l, \dots, k-1$, we obtain by applying the BFGS formula (2.3) recursively that

$$\begin{aligned}
\bar{H}^k = {} & ((V^{k-1})^T \cdots (V^{k-l})^T)(\bar{H}^k)^0 (V^{k-l} \cdots V^{k-1}) \\
& + \rho^{k-l}((V^{k-1})^T \cdots (V^{k-l+1})^T) s^{k-l} (s^{k-l})^T (V^{k-l+1} \cdots V^{k-1}) \\
& + \rho^{k-l+1}((V^{k-1})^T \cdots (V^{k-l+2})^T) s^{k-l+1} (s^{k-l+1})^T (V^{k-l+2} \cdots V^{k-1}) \\
& + \cdots \\
& + \rho^{k-1} s^{k-1} (s^{k-1})^T
\end{aligned} \tag{2.4}$$

which is the L-BFGS approximation. From this formula, an efficient recursive procedure is derived to compute the matrix-vector product $\bar{H}^k g^k$. Here, $(\bar{H}^k)^0$ is generally set to be $\gamma^k I$ with $\gamma^k = \frac{(s^{k-1})^T w^{k-1}}{(w^{k-1})^T w^{k-1}}$ which has proved to be effective in practice. For completeness, we describe the L-BFGS recursive procedure (Nocedal and Wright 2006, Algorithm 7.4) for $\bar{H}^k g^k$ in the following.

Algorithm 1: L-BFGS two-loop recursion

```
1  Input $S^k = [s^{k-l}, \dots, s^{k-1}]$ and $W^k = [w^{k-l}, \dots, w^{k-l}]$;
2  Compute $\gamma^k = \frac{(s^{k-1})^T w^{k-1}}{(w^{k-1})^T w^{k-1}}$;
3  $q \leftarrow g^k$;
4  for $i = k-1 : k-l$ do
5      $\alpha^i \leftarrow \rho^i (s^i)^T q$;
6      $q \leftarrow q - \alpha^i w^i$;
7  end
8  $r \leftarrow \gamma^k q$;
9  for $i = k-l : k-1$ do
10     $\beta \leftarrow \rho^i (w^i)^T r$;
11     $r \leftarrow r + s^i(\alpha^i - \beta)$;
12 end
13 Return the result $\bar{H}^k g^k = r$;
```

We emphasize that $d^k = -\bar{H}^k g^k$ cannot be taken as a search direction for the dual problem (1.4) due to the nonnegative constraints. We might only need to apply the Hessian information $\bar{H}^k$ to the gradient corresponding to free variables of the problem (1.4) instead of the full gradient. We will give the details in the next subsection.

2.2 Detection of active set and determination of the search direction

In this subsection, we describe how to detect the active set of inequalities and how to determine the search direction. We first define index sets $C(y^k)$ and $D(y^k)$ as

$$C(y^k) = \{i | 0 \le y_i^k \le \epsilon^k, i \in \mathcal{I}\}$$

and

$$D(y^k) = \mathcal{I} \backslash C(y^k),$$

where $\epsilon^k = \min\{\epsilon, r_g^k\}$ is a tolerance for estimating the active set at a minimizer and $\epsilon \in (0, \frac{1}{3})$ is a safeguard parameter which prevents ϵ^k from being too large. It should be noted that the index sets $C(y^k)$ and $D(y^k)$ can be regarded as the approximate active set of inequalities and the approximate inactive set of inequalities respectively. We remark that $C(y^k)$ can detect the active set of inequalities accurately for all sufficiently large k if y^k tends to a minimizer of the problem (1.4).

Next, we determine the search direction d^k which is composed of the safe step d^{k1} and the fast step d^{k2}, where the safe step is for global convergence, and the fast step is for fast local convergence.

(I) Generate the safe step d^{k1}. In this situation, the approximate active set $C(y^k)$ is divided into three parts:

$$\begin{aligned}
C_1(y^k) &= \{i | y_i^k = 0 \text{ and } g_i^k \ge 0, i \in \mathcal{I}\},\\
C_2(y^k) &= \{i | 0 \le y_i^k \le \epsilon^k \text{ and } g_i^k < 0, \quad i \in \mathcal{I}\},\\
C_3(y^k) &= \{i | 0 < y_i^k \le \epsilon^k \text{ and } g_i^k \ge 0, \quad i \in \mathcal{I}\}.
\end{aligned}$$

A few comments about the index sets $C_i(y^k),\ i = 1, 2, 3$ and $D(y^k) \cup \mathcal{E}$ are made here. (a) The variables corresponding to $C_1(y^k)$ meet the optimality condition (2.1), and then they are kept to stay on the boundary. (b) Variables $y_i^k,\ i \in C_2(y^k) \cup C_3(y^k)$, are close to the boundary, and we prefer to the steepest direction and truncate them on the boundary. As for free variables corresponding to $D(y^k) \cup \mathcal{E}$, the L-BFGS technique is used to accelerate them.

For simplicity, we denote $D(y^k) \cup \mathcal{E}$ by $D_e(y^k)$. Let N^k and $P_i^k, i = 1, 2$ be matrices whose columns are $\{e_j | j \in D_e(y^k)\}$ and $\{e_j | j \in C_i(y^k), i = 3, 2\}$, respectively, where e_j is the j-th column of the $m \times m$ identity matrix. The safe step d^{k1} is given by

$$d_i^{k1} = \begin{cases} 0, & i \in C_1(y^k),\\ -\left[P_1^k \left(P_1^k\right)^T T^k g^k\right]_i, & i \in C_3(y^k),\\ -\left[P_2^k \left(P_2^k\right)^T g^k\right]_i, & i \in C_2(y^k),\\ -\left[N^k H^k (N^k)^T g^k\right]_i, & i \in D_e(y^k), \end{cases} \tag{2.5}$$

where T^k is a diagonal matrix with diagonals

$$t_i^k = \begin{cases} 0, & i \notin C_3(y^k),\\ \min\left\{\dfrac{y_i^k}{g_i^k}, 1\right\}, & i \in C_3(y^k), \end{cases} \tag{2.6}$$

$H^k = (N^k)^T \bar{H}^k N^k$, and the positive definite matrix $\bar{H}^k$ is from (2.4). It should be emphasized that we calculate

$$d_i^{k1} = -\left[N^k H^k (N^k)^T g^k\right]_i, \ i \in D_e(y^k)$$

by Algorithm 1 instead of forming $\bar{H}^k$ explicitly by (2.4).

(II) Generate the fast step d^{k2}.

We are aware that $g_i^k \geq 0$ in $C_1(y^k)$ and $C_3(y^k)$ is too tight to estimate the active set in the case that both y_i and $g_i(y)$ vanish at the minimizer. Thus, we define

$$\bar{C}_1(y^k) = \{i | 0 \leq y_i^k \leq \epsilon^k \text{ and } g_i^k \geq -\epsilon^k, i \in \mathcal{I}\},$$
$$\bar{C}_2(y^k) = \{i | 0 \leq y_i^k \leq \epsilon^k \text{ and } g_i^k < -\epsilon^k, i \in \mathcal{I}\}$$

where the condition $g_i^k \geq 0$ is relax to be $g_i^k \geq -\epsilon^k$. Since $\bar{C}_1(y^k)$ is a working set for estimating the active set of inequalities we prefer to push all $y_i^k, i \in \bar{C}_1(y^k)$ onto the boundary. As for the inactive set candidate $\bar{C}_2(y^k)$, the negative gradient direction is preferred. Therefore, the fast step d^{k2} is defined as

$$d_i^{k2} = \begin{cases} -y_i^k, & i \in \bar{C}_1(y^k), \\ -g_i^k, & i \in \bar{C}_2(y^k), \\ -\left[N^k H^k (N^k)^T g^k\right]_i, & i \in D_e(y^k), \end{cases} \tag{2.7}$$

where H^k is $(N^k)^T \bar{H}^k N^k$ with $\bar{H}^k$ from (2.4) and Algorithm 1 is also invoked to calculate d^{k2} as well as d^{k1}. If the fast step d^{k2} is a "sufficiently" descent direction for $\phi(y)$ compared to the safe step d^{k1}, we take $d^k = d^{k2}$, otherwise $d^k = d^{k1}$. Specifically,

$$d^k = \begin{cases} d^{k2}, & (g^k)^T d^{k2} \leq \gamma (g^k)^T d^{k1} \text{ with } \gamma \in \left(0, \frac{1}{2}\right), \\ d^{k1}, & \text{otherwise.} \end{cases} \tag{2.8}$$

2.3 Algorithm

In this subsection, we present a nonmontone L-BFGS method for solving the dual problem (1.4). Before stating the algorithm, we first discuss the required conditions for dual-type methods, such as the conditions ensuring zero duality gap. According to Rockafellar (1974, Theorems 17 and 18), we need the Slater condition

$$\begin{cases} \{A_i\}_{i=1}^p \text{ are linearly independent}, \\ \exists \text{ a positive definite matrix } X^0 \in \mathcal{S}^n \text{ such that } \langle A_i, X^0 \rangle \geq b_i, \ i = p+1, \ldots, m, \end{cases} \tag{2.9}$$

which leads to zero duality gap between the primal and dual problems. The following proposition is a direct application of Rockafellar (1974, Theorems 17 and 18) (see also Gao and Sun 2009, Proposition 2.1).

Proposition 2.1 *If the Slater condition* (2.9) *holds, then*

(i) there exists at least one $y^* \in \mathbb{R}^m$ which is the solution of the problem (1.4), *and $X^* = (C + \mathcal{A}^* y^*)_+$ gives the unique solution of the problem* (1.1);

(ii) the level set $\{y \in \mathbb{R}^m | \phi(y) \leq \bar{c}\}$ is closed, bounded, and convex, where $\bar{c}$ is any prefixed real number.

Given the iterate y^k, we compute the search direction d^k defined in (2.8) by invoking Algorithm 1, and then determine the step length α^k along d^k. One may adopt the Armijo backtracking line search technique to determine α^k, that is, finding the smallest α^k in $\{1, t, t^2, \ldots\}$

with $t \in (0,1)$ such that

$$\phi(P_\Omega(y^k + \alpha^k d^k)) \le \phi(y^k) + \sigma\alpha^k (g^k)^T d^k, \tag{2.10}$$

which is the so-called Armijo condition, where $\sigma \in (0, \frac{1}{2})$ is a scalar. To make (2.10) flexible, we employ the nonmonotone reduction condition on $\phi(y)$ (Zhang and Hager 2004):

$$\phi(P_\Omega(y^k + \alpha^k d^k)) \le \phi_p^k + \sigma\alpha^k (g^k)^T d^k, \tag{2.11}$$

where $Q^0 = 1$, $Q^{k+1} = \delta Q^k + 1$ with $\delta \in (0,1)$, $\phi_p^0 = \phi(y^0)$, and $\phi_p^{k+1} = (\delta Q^k \phi_p^k + \phi(y^{k+1}))/Q^{k+1}$. Here ϕ_p^{k+1} is actually an weighted average of the past function values which is better than the maximum of the several past function values (see Zhang and Hager 2004).

Now we present our main algorithm below.

Algorithm 2: Nonmonotone Limited memory BFGS algorithm (N-L-BFGS)

```
1  Given $\varepsilon > 0$, $\gamma \in (0, \frac{1}{2})$, $\sigma \in (0, \frac{1}{2})$, $\tau \in (0,1)$, $t \in (0,1)$, $\delta \in (0,1)$;
2  Initialization: $y^0 \in \mathbb{R}^m$, $Q^0 = 1$, $\phi_p^0 = \phi(y^0)$, $k = 0$ ;
3  while $\|r_g^k\| > \varepsilon$ do
4      Compute the search direction $d^k$ in (2.8);
5      Compute gp=$(g^k)^T d^k$, and set $\alpha = 1$;
6      while $\phi(P_\Omega(y^k + \alpha d^k)) > \phi_p^k + \sigma\alpha gp$ do
7          Reduce $\alpha = t\alpha$;
8          Compute $\phi(P_\Omega(y^k + \alpha d^k))$;
9      end
10     Set $\alpha^k = \alpha$, $y^{k+1} = P_\Omega(y^k + \alpha^k d^k)$, $Q^{k+1} = \delta Q^k + 1$, and
       $\phi_p^{k+1} = (\delta Q^k \phi_p^k + \phi(y^{k+1}))/Q^{k+1}$;
11     Update $\bar{H}^{k+1}$ by $S^k$ and $W^k$;
12     Set $k = k + 1$;
13 end
```

Remark 2.1 Noting that $\phi_p^0 = \phi(y^0)$ and $Q^0 = 1$, it follows that

$$\phi_p^1 = \frac{\delta Q^0 \phi_p^0 + \phi(y^1)}{Q^1} \le \frac{\delta Q^0 \phi_p^0 + \phi_p^0}{Q^1} = \phi_p^0,$$

where the inequality follows from (2.11) and $(g^k)^T d^k \le 0$ (it will be proved in Lemma 3.1). Repeating above process yields $\phi_p^{k+1} \le \phi_p^k$ for all k, which means that $\{\phi_p^k\}_{k\ge 0}$ is a monotonically decreasing sequence.

Remark 2.2 From the definitions of ϕ_p^{k+1} and Q^{k+1}, it holds that

$$\phi(y^{k+1}) = \phi_p^{k+1} + \left(\phi_p^{k+1} - \phi_p^k\right)\delta Q^k \le \phi_p^{k+1},$$

where the last inequality follows from monotonicity of $\{\phi_p^k\}_{k\ge 0}$. That is, $\phi(y^k) \le \phi_p^k$ holds for all k. It means that the nonmonotone reduction condition (2.11) does relax the classical Armijo condition (2.10).

Remark 2.3 We emphasize that the full matrix $\bar{H}^k$ does not need to be formed. What we need is to store S^k and W^k, and then using them we invoke Algorithm 1 to calculate d^k.

Remark 2.4 Our algorithm as well as those in Malick (2004), Boyd and Xiao (2005), Gao and Sun (2009) is not suitable for the problem (1.1) with large n. The reason is that the spectral decomposition of an $n \times n$ symmetric matrix is required at each iteration when computing $\phi(y)$ and $\nabla\phi(y)$. As we all know, the spectral decomposition for a large matrix is computationally expensive or even prohibited. But if $C + \mathcal{A}^* y$ in (1.4) has some special structure (such as low-rank or narrow-band structures), we can obtain $\phi(y)$ and $\nabla\phi(y)$ from a small portion of eigenvalues/eigenvectors of $C + \mathcal{A}^* y$ which are less expensive to calculate. Denote $C + \mathcal{A}^* y$ by G. If G is block-diagonal, we can readily calculate eigen-decomposition of each block, and then $\phi(y)$ and $\nabla\phi(y)$ are obtained. There are also other cases where the dual objective function and its gradient are easy to calculate. Suppose that the eigenvalues of G are $\lambda_1 \geq \cdots \geq \lambda_s \geq 0 > \lambda_{s+1} \geq \cdots \geq \lambda_n$ and their corresponding eigenvectors are $u_1, \ldots, u_n$. It is straightforward from (1.4) and (1.5) that the dual function

$$\phi(y) = \frac{1}{2} \sum_{i=1}^{(} \lambda_i)^2 - b^T y + \frac{1}{2}\|C\|_F^2$$

and its gradient

$$\begin{aligned}\nabla\phi(y) &= \mathcal{A}(G_+) - b \\ &= \mathcal{A}\left(\sum_{i=1}^{s} \lambda_i u_i u_i^T\right) - b \\ &= \sum_{i=1}^{s} \lambda_i \begin{pmatrix} u_i^T A_1 u_i \\ \vdots \\ u_i^T A_m u_i \end{pmatrix} - b.\end{aligned}$$

If s is very small (say, the final solution has low-rank structure), one can use the subroutine DSYEVX in Lapack to compute all positive eigenvalues and their corresponding eigenvectors, and then $\phi(y)$ and $\nabla\phi(y)$ are obtained from above alternative expressions. If s is very close to n, then $\phi(y)$ and $\nabla\phi(y)$ can also be calculated cheaply. The reason is that one can express $\phi(y)$ and $\nabla\phi(y)$ as

$$\phi(y) = \frac{1}{2}\|G\|_F^2 - \frac{1}{2} \sum_{i=s+1}^{n} (\lambda_i)^2 - b^T y + \frac{1}{2}\|C\|_F^2$$

and

$$\nabla\phi(y) = \mathcal{A}(G) - \sum_{i=s+1}^{n} \lambda_i \begin{pmatrix} u_i^T A_1 u_i \\ \vdots \\ u_i^T A_m u_i \end{pmatrix} - b.$$

In addition, for partial eigenvalues and eigenvectors, FEAST eigenvalue solver (FEAST solver 2009-2015; Polizzi 2009) is also a good choice, especially in the situation where linear equations $Gx = c$ (c is a vector with appropriate size) can be solved efficiently (say, G has narrow–band structure). More details about matrix approximation problem with banded structure can be seen in Xue and Shen (2018). But anyhow, in this paper, we do not make any assumption on the structure of the data matrices C and A_i, $i = 1, \ldots, m$.

Remark 2.5 We now give a few comments on related work about active set L-BFGS methods. In Xiao and Wei (2007), Xiao and Li (2008), Yuan and Lu (2011), the estimated active set corresponding to the problem (1.4) is defined as $C(y^k) = \{i | y_i^k \le a_i(y^k) g_i^k, i \in \mathcal{I}\}$ where $a_i(y^k)$, $i = 1, \ldots, m$ are nonnegative continuous functions, and then all the variables y^i, $i \notin C(y^k)$ are accelerated by quasi-Newton methods. The classic L-BFGS method (Nocedal and Wright 2006, Chapter 7) and the modified BFGS method Yuan and Wei (2010) are applied in Xiao and Wei (2007) and Yuan and Lu (2011), respectively, but global convergence needs some strong conditions (say, the strict complementarity condition in Yuan and Lu 2011, Theorem 3.1). For Rahpeymaii et al. (2016), Xiao and Li (2008), convex quadratic programs with bound constraints need to be solved to generate the search directions, but this is computationally expensive compared with our algorithm. Kelley (1999) presented a projected BFGS method where the estimated active set does not match the optimality condition (2.1) very well. Compared with the algorithm in Ni and Yuan (1997), the safe step d^{k1} in our algorithm is actually the search direction in Ni and Yuan (1997) but the fast step d^{k2} is used first if it is a good descent direction for ϕ (see (2.8)). Note that the fast step d^{k2} is related to $\bar{C}_1(y^k)$ which aims at capturing the active set even in the degenerate case (i.e., both $y_i^k \to 0$ and $g_i^k \to 0$ for some $i \in \bar{C}_1(y^k)$). In this sense, our algorithm identifies free variables (to be accelerated) close to the boundary more accurately.

3 Convergence analysis

In this section, we establish the global convergence of Algorithm 2 under the following assumptions.

- **(A1)** The sequence $\{y^k\}_{k\ge 0}$ generated by Algorithm 2 is contained a convex and bounded set of $\mathbb{R}^m$.
- **(A2)** The sequence $\{H^k\}_{k\ge 0}$ is uniformly positive definite and bounded for all k. It then follows that there exist two scalars $a > 0$ and $b > 0$ that $a\|d\|^2 \le d^T H^k d \le b\|d\|^2$ holds for all k.

The first lemma shows that d^k generated by Algorithm 2 is a descent direction for $\phi(y)$ if $d^{k1} \ne 0$.

Lemma 3.1 *Assume that (A1)–(A2) hold. Let y^k and d^k be generated by Algorithm 2. Then*

(i) Inequality $(d^k)^T g^k \le 0$ holds for all k.
(ii) Equality $(d^k)^T g^k = 0$ holds if and only if $d^{k1} = 0$.

Proof (i) From (2.8), $d^k = d^{k1}$ or d^{k2}. If $d^k = d^{k1}$, it follows from (2.5) and (2.6) that

$$\begin{aligned}(g^k)^T d^k &= (g^k)^T d^{k1}\\ &= -\sum_{i\in C_3(y^k)} [P_1^k \left(P_1^k\right)^T T^k g^k]_i g_i^k - \sum_{i\in C_2(y^k)} [P_2^k \left(P_2^k\right)^T g^k]_i g_i^k\\ &\quad - \sum_{i\in D_e(y^k)} [N^k H^k (N^k)^T T^k g^k]_i g_i^k\\ &= -\sum_{i\in C_3(y^k)} \min(y_i^k, g_i^k) g_i^k - \sum_{i\in C_2(y^k)} (g_i^k)^2 - (g^k)^T N^k H^k (N^k)^T g^k.\end{aligned} \tag{3.1}$$

Since $y_i^k \ge 0$ for all i and $g_i^k \ge 0$ for $i \in C_3(y^k)$, the first term in the right-hand side of (3.1) is non-positive. It is obvious that the second term is non-positive, and the last term is non-positive due to (A2). Hence, $(d^k)^T g^k \le 0$ for all k in the case that $d^k = d^{k1}$. If $d^k = d^{k2}$, we know from (2.8) that $(d^k)^T g^k \le \gamma (d^{k1})^T g^k \le 0$ where the last inequality follows from the earlier proof. Hence, the first assertion of this lemma holds.

(ii) If $d^k = d^{k1}$ and $(d^k)^T g^k = 0$, it follows that each term in the right-hand side of (3.1) vanishes because of its non-positiveness. The first term $-\sum_{i\in C_3(y^k)} \min(y_i^k, g_i^k) g_i^k = 0$ gives $g_i^k = 0$ for $i \in C_3(y^k)$ which yields $d_i^{k1} = 0$ for $i \in C_3(y^k)$. It immediately follows from (A2) that $g_i^k = 0$ for $i \in C_2(y^k) \cup D_e(y^k)$ since the last terms in the right-hand side of (3.1) vanishes. This together with the definition (2.5) of d^{k1} yields that $d_i^{k1} = 0$ for $i \in C_2(y^k) \cup D_e(y^k)$. As for $i \in C_1(y^k)$, $d_i^{k1} = 0$ from its definition. Overall, $d^k = 0$ if $d^k = d^{k1}$ and $(d^k)^T g^k = 0$. If $d^k = d^{k2}$ and $(d^k)^T g^k = 0$, it follows from (2.8) that $0 = (g^k)^T d^k \le \gamma (g^k)^T d^{k1} \le 0$, and then $(g^k)^T d^{k1} = 0$. Using above proof, we again get $d^{k1} = 0$.

Conversely, if $d^{k1} = 0$, it immediately follows $(g^k)^T d^k = 0$ with $d^k = d^{k1}$. If $d^k = d^{k2}$, we have from the definition (2.7) of d^{k2} that

$$(g^k)^T d^k = -\sum_{i\in \bar{C}_1(y^k)} g_i^k y_i^k - \sum_{i\in \bar{C}_2(y^k)} (g_i^k)^2 - \sum_{i\in D_e(y^k)} [N^k H^k (N^k)^T T^k g^k]_i g_i^k. \tag{3.2}$$

From $d^{k1} = 0$, we know from the definition (2.5) of d^{k1} that $y_i^k = 0$ with $i \in C_1(y^k)$, $C_2(y^k) = \emptyset$ (otherwise $g_i^k = 0$ with $i \in C_2(y^k)$, but it contradicts the definition of $C_2(y^k)$), and $\min(g_i^k, y_i^k) = 0$ with $i \in C_3(y^k)$, which leads to $\sum_{i\in \bar{C}_1(y^k)} g_i^k y_i^k + \sum_{i\in \bar{C}_2(y^k)} (g_i^k)^2 = 0$. The third term in the right-hand side of (3.2) also vanishes because $d_i^{k1} = [N^k H^k (N^k)^T T^k g^k]_i = 0$ for $i \in D_e(y^k)$. Hence, $(g^k)^T d^k = 0$ with $d^k = d^{k2}$. □

We next prove that $d^{k1} = 0$ is inequivalent to the optimality condition at y^k, which is presented in the following theorem.

Theorem 3.2 *Assume that (A1)–(A2) hold. Let y^k and d^{k1} be generated by Algorithm 2. Then y^k is a KKT point of the problem (1.4) if and only if $d^{k1} = 0$.*

Proof Suppose that $d^{k1} = 0$. We know by the definitions of $C_i(y^k)$, $i = 1, 2, 3$ and d^{k1} that $y_i^k = 0$ and $g_i^k \ge 0$ for $i \in C_1(y^k)$, and $C_2(y^k) = \emptyset$, and $\min(g_i^k, y_i^k) = 0$ for $i \in C_3(y^k)$. As for $i \in D_e(y^k)$, it follows from (A1) that $[N^k H^k (N^k)^T g^k]_i = 0$ implies $g_i^k = 0$. Therefore, $g_i^k = 0$ with $i \in \mathcal{E}$, and $\min(g_i^k, y_i^k) = 0$ with $i \in \mathcal{I}$, which implies that y^k is a KKT point of the problem (1.4). Conversely, if y^k is a KKT point, then $\epsilon^k = \min(\epsilon, r_b^k) = 0$ which leads to $C_3(y^k) = \emptyset$. Using (2.1), we have that $g_i^k \ge 0$ for all $i \in \mathcal{I}$ and then $C_2(y^k) = \emptyset$. Hence, it holds that $y_i^k > 0$ and $g_i^k = 0$ for $i \in D(y^k)$ and that $y_i^k = 0$ and $g_i^k = 0$ for $i \in \mathcal{E}$. It together with the definition (2.5) of d^{k1} yields $d_i^{k1} = 0$ for all $i \in D_e(y^k)$. □

Remark 3.1 If $\lim_{k\in\mathcal{K},k\to\infty} d^{k1} = 0$ and $\lim_{k\in\mathcal{K},k\to\infty} y^k = y^*$ for some infinite index set $\mathcal{K}$, similar to the proof of Theorem 3.2, we can also show that y^* is a KKT point of the problem (1.4).

Lemma 3.3 *Assume that (A1)–(A2) hold. Let y^k and d^k be generated by Algorithm 2. Then*

$$\beta^k \ge \min\left\{1, \frac{\epsilon^k}{\|d^k\|_\infty + 1}\right\}, \tag{3.3}$$

where $\beta^k = \sup_{0\le\eta\le 1}\{\eta | y_i^k + \eta d_i^k \ge 0,\ i \in \mathcal{I}\}$.

Proof Let $\bar\beta^k = \min\left\{1, \frac{\epsilon^k}{\|d^k\|_\infty + 1}\right\}$. Since β^k is the supremum of the set $\{\eta | y_i^k + \eta d_i^k \ge 0,\ i \in \mathcal{I}\}$, we only have to prove $y_i^k + \bar\beta^k d_i^k \ge 0,\ i \in \mathcal{I}$. Two cases are considered in the following.
Case (i) $d^k = d^{k1}$. If $i \in C_1(y^k)$, then $d_i^{k1} = 0$ and $y_i^k + \bar\beta^k d_i^k \ge 0$. If $i \in C_2(y^k)$, then $d_i^{k1} = -g_i^k > 0$ and $y_i^k + \bar\beta^k d_i^k \ge 0$. If $i \in C_3(y^k)$, then $d_i^{k1} = -\min(g_i^k, y_i^k) < 0$, and it holds that

$$y_i^k + \bar\beta^k d_i^k \ge y_i^k + d_i^{k1} = y_i^k - \min\left(g_i^k, y_i^k\right) \ge 0.$$

For $i \in D_e(y^k)$, $d_i^{k1} = [N^k H^k (N^k)^T T^k g^k]_i$. If $d_i^{k1} > 0$, then it is obvious that $y_i^k + \bar\beta^k d_i^k \ge 0$. If $d_i^{k1} < 0$, we know by the definitions of $D_e(y^k)$ and $\bar\beta^k$ that

$$y_i^k + \bar\beta^k d_i^k \ge \epsilon^k + \min\left\{1, \frac{\epsilon^k}{\|d^k\|_\infty + 1}\right\} d_i^k \ge \epsilon^k + \frac{\epsilon^k}{\|d^k\|_\infty + 1} d_i^k \ge 0.$$

Case (ii) $d^k = d^{k2}$. For $i \in \bar C_1(y^k)$, $y_i^k + \bar\beta^k d_i^{k2} = y_i^k - \bar\beta^k y_i^k \ge 0$, and for $i \in \bar C_2(y^k)$, $g_i^k < -\epsilon^k$ and therefore $y_i^k + \bar\beta^k d_i^{k2} = y_i^k - \bar\beta^k g_i^k \ge 0$. Similar to the proof of **Case (i)**, for $i \in D_e(y^k)$, we also have that $y_i^k + \bar\beta^k d_i^{k2} \ge 0$ since $d_i^{k2} = d_i^{k1}$, $i \in D_e(y^k)$. Overall, we get that $y_i^k + \bar\beta^k d_i^k \ge 0,\ i \in \mathcal{I}$ in both cases, which completes the proof. □

From Lemma 3.3, we conclude that $P_\Omega(y^k + \alpha d^k) = y^k + \alpha d^k$ if $0 < \alpha \le \beta^k$. Next lemma shows that d^k is a "sufficiently" descent direction for $\phi(y)$ at y^k.

Lemma 3.4 *Assume that (A1)–(A2) hold. Then*

$$\|d^{k1}\|^2 \le -\frac{b^2}{\gamma a}(g^k)^T d^k, \tag{3.4}$$

where a and b are from (A2).

Proof According to (2.8), $d^k = d^{k1}$ or d^{k2}. We consider the first case $d^k = d^{k1}$. It holds from (2.5) that

$$\begin{aligned}
\sum_{i\in C_3(y^k)} (d_i^{k1})^2 &= \sum_{i\in C_3(y^k)} \left[P_1^k\left(P_1^k\right)^T T^k g^k\right]_i^2 \\
&= \sum_{i\in C_3(y^k)} \left[P_1^k\left(P_1^k\right)^T T^k g^k\right]_i \min\left\{\frac{y_i^k}{g_i^k}, 1\right\} g_i^k \\
&\le \sum_{i\in C_3(y^k)} \left[P_1^k\left(P_1^k\right)^T T^k g^k\right]_i g_i^k, \qquad (3.5)\\
\sum_{i\in C_2(y^k)} (d_i^{k1})^2 &= \sum_{i\in C_2(y^k)} (g_i^k)^2 \\
&= \sum_{i\in C_2(y^k)} \left[P_2^k\left(P_2^k\right)^T g^k\right]_i g_i^k \qquad (3.6)
\end{aligned}$$

and

$$\begin{aligned}\sum_{i\in D_e(y^k)}(d_i^{k1})^2 &= \sum_{i\in D_e(y^k)}\left[N^kH^k(N^k)^Tg^k\right]_i^2\\ &\le b^2\|(N^k)^Tg^k\|^2\\ &\le \frac{b^2}{a}(g^k)^TN^kH^k(N^k)^Tg^k\\ &= \frac{b^2}{a}\sum_{i\in D_e(y^k)}\left[N^kH^k(N^k)^Tg^k\right]_i g_i^k, \end{aligned}\tag{3.7}$$

where the two inequalities in (3.7) follow from (A2). Using (3.5)–(3.7) and (2.5), we obtain that

$$\begin{aligned}\|d^{k1}\|^2 &= \sum_{i\in C_3(y^k)}(d_i^{k1})^2+\sum_{i\in C_2(y^k)}(d_i^{k1})^2+\sum_{i\in D_e(y^k)}(d_i^{k1})^2\\ &\le \frac{b^2}{a}\sum_{i\in C_3(y^k)}\left[P_1^k\left(P_1^k\right)^TT^kg^k\right]_i g_i^k+\frac{b^2}{a}\sum_{i\in C_2(y^k)}\left[P_2^k\left(P_2^k\right)^Tg^k\right]_i g_i^k\\ &\quad+\frac{b^2}{a}\sum_{i\in D_e(y^k)}\left[N^kH^k(N^k)^Tg^k\right]_i g_i^k\\ &= -\frac{b^2}{a}(g^k)^Td^k, \end{aligned}\tag{3.8}$$

which implies (3.4) due to $\gamma\in(0,\frac{1}{2})$ and (i) in Lemma 3.1.

In the second case that $d^k=d^{k2}$, it follows from (2.8) and (3.8) that

$$\|d^{k1}\|^2\le-\frac{b^2}{a}(g^k)^Td^{k1}\le-\frac{b^2}{\gamma a}(g^k)^Td^k,$$

which completes the proof. □

We next prove that the sequence $\{d^k\}_{k\ge0}$ is bounded for all k.

Lemma 3.5 *Assume that (A1)–(A2) hold. Let d^k be generated by Algorithm 2. Then there exists a scalar $\bar{d}_b>0$ such that*

$$\|d^k\|\le\bar{d}_b\tag{3.9}$$

holds for all k.

Proof (A1) and the Lipschitz continuity of $\nabla\phi(y)$ imply that g^k is bounded for all k, that is, there exists a scalar $\bar{b}>0$ such that $\|g^k\|\le\bar{b}$ for all k. By (3.8), we have that

$$\|d^{k1}\|^2\le-\frac{b^2}{\gamma a}(g^k)^Td^k\le\frac{b^2}{\gamma a}\|g^k\|\|d^k\|\quad\text{with}\quad d^k=d^{k1},$$

which together with boundedness of g^k yields that $\|d^{k1}\|\le\frac{b^2}{\gamma a}\|g^k\|\le\frac{b^2\bar{b}}{\gamma a}$. Besides, from the definition of d^{k2}, we have that $\|d^{k2}\|\le\|y^k\|+\|g^k\|+\|N^kH^k(N^k)^Tg^k\|\le\|y^k\|+\|g^k\|+b^2\|N^kg^k\|^2$, where the last inequality follows from (A2). This combining with boundedness of y^k and g^k leads to boundedness of d^{k2}. Boundedness of d^{k1} and d^{k2} implies that d^k is bounded for all k, which completes the proof. □

Lemmas 3.3 and 3.5 lead to the following lemma.

Lemma 3.6 *Assume that (A1)–(A2) hold. Then there exists a scalar* $\bar{\beta} > 0$ *such that*

$$\beta^k \geq \min(1, \bar{\beta})\epsilon^k, \tag{3.10}$$

where β^k *is from Lemma 3.3.*

In the following theorem, we prove that Algorithm 2 is well-defined, i.e., the inner loop (Lines 5–7) in Algorithm 2 can stop if the trial stepsize $\alpha > 0$ is small enough.

Theorem 3.7 *Assume that (A1)–(A2) hold. If the trial stepsize* $\alpha > 0$ *satisfies*

$$\alpha \leq \min\left(\beta^k, \nu\frac{\|d^{k1}\|^2}{\|d^k\|^2}\right),$$

then

$$\phi(P_\Omega(y^k + \alpha d^k)) \leq \phi_p^k + \sigma\alpha(g^k)^T d^k, \tag{3.11}$$

i.e., the inner loop terminates, where $\nu = \frac{2(1-\sigma)a\gamma}{Lb^2}$, β^k *is from Lemma 3.3, scalars a and b are from (A2),* $\gamma \in (0, \frac{1}{2})$ *is from (2.8), and L is the Lipschitz constant of* $\nabla\phi(y)$.

Proof By the definition of β^k, $\phi(P_\Omega(y^k + \alpha d^k)) = \phi(y^k + \alpha d^k)$ whenever $\alpha \leq \beta^k$. By the mean value theorem,

$$\phi(y^k + \alpha d^k) - \phi(y^k) = \alpha(g^k)^T d^k + \alpha\int_0^1 [\nabla\phi(y^k + \alpha s d^k) - \nabla\phi(y^k)]^T d^k ds,$$

and therefore

$$\begin{aligned}
&\phi(y^k + \alpha d^k) - \phi(y^k) - \alpha\sigma(g^k)^T d^k \\
&= \alpha(1-\sigma)(g^k)^T d^k + \alpha\int_0^1 [\nabla\phi(y^k + \alpha s d^k) - \nabla\phi(y^k)]^T d^k ds \\
&\leq -\frac{\alpha(1-\sigma)a\gamma}{b^2}\|d^{k1}\|^2 + \frac{\alpha^2 L}{2}\|d^k\|^2,
\end{aligned}$$

where the inequality follows from Lemma 3.4 and the Lipschitz continuity of $\nabla\phi(y)$. Consequently, if

$$\alpha \leq \min\left(\beta^k, \nu\frac{\|d^{k1}\|^2}{\|d^k\|^2}\right),$$

then

$$\phi(y^k + \alpha d^k) \leq \phi_p^k + \sigma\alpha(g^k)^T d^k,$$

which implies that the nonmonotone reduction condition (3.11) on $\phi(y)$ is satisfied.

As preparation for global convergence, we show in the following lemma that α^k generated by Algorithm 2 is bounded below from some positive quantity related to ϵ^k and d^{k1}. □

Lemma 3.8 *Assume that (A1)–(A2) hold. Let* α^k *be generated by Algorithm 2. Then there exists a scalar* $\bar{\alpha} > 0$ *such that*

$$\alpha^k \geq \min\left\{t, t\bar{\beta}\epsilon^k, \bar{\alpha}\|d^{k1}\|^2\right\}, \tag{3.12}$$

where $t \in (0, 1)$ *is from Algorithm 2 and* $\bar{\beta} > 0$ *is from Lemma 3.6.*

Proof By the mechanism of Algorithm 2 we know that α^k is the largest number in $\{1, t, t^2, \ldots, t^k, \ldots\}$ such that (2.11) is satisfied for α^k, which implies

$$\alpha^k \geq t\min\left(\beta^k, \nu\frac{\|d^{k1}\|^2}{\|d^k\|^2}\right)$$

where ν is from Lemma 3.7. According to Lemmas 3.5 and 3.6, we obtain that (3.12) holds for some scalar $\bar{\alpha} > 0$.

Now, it is our position to establish the global convergence of our main algorithm.

Theorem 3.9 *Assume that (A1)–(A2) hold. Let the sequence $\{y^k\}_{k\geq 0}$ be generated by Algorithm 2. Then any accumulation point of $\{y^k\}_{k\geq 0}$ is a KKT point of the problem (1.4).*

Proof As y^k and α^k are generated by Algorithm 2, the nonmonotone reduction condition (2.11) is satisfied. Since $\phi(y)$ is continuously differentiable and $\{y^k\}_{k\geq 0}$ is bounded due to (A1), we obtain that $\{\phi(y^k)\}_{k\geq 0}$ is bounded below, which combining with Remark 2.2 yields that $\{\phi_p^k\}_{k\geq 0}$ is also bounded below. It follows with Remark 2.1 that $\{\phi_p^k\}_{k\geq 0}$ is convergent. By (2.11) and the definitions of ϕ_p^{k+1} and Q^{k+1}, we obtain that

$$\begin{aligned}\phi_p^{k+1} &= \frac{\delta Q^k\phi_p^k + \phi(y^{k+1})}{Q^{k+1}}\\ &\leq \frac{\delta Q^k\phi_p^k + \phi_p^k}{Q^{k+1}} + \frac{\sigma\alpha^k(g^k)^Td^k}{Q^{k+1}}\\ &= \phi_p^k + \frac{\sigma\alpha^k(g^k)^Td^k}{Q^{k+1}}.\end{aligned}$$

Hence, it holds that

$$-\sum_{k=0}^{\infty}\frac{\alpha^k(g^k)^Td^k}{Q^{k+1}} < +\infty.$$

Using Lemma 3.4, we have that

$$+\infty > -\sum\lim_{k=0}^{\infty}\frac{\alpha^k(g^k)^Td^k}{Q^{k+1}} \geq \sum\lim_{k=0}^{\infty}\frac{a\gamma\alpha^k}{b^2Q^{k+1}}\|d^{k1}\|^2. \tag{3.13}$$

As $Q^{k+1} = \delta Q^k + 1$ and $\delta \in (0, 1)$, we have that

$$1 \leq Q^{k+1} = \frac{1-\delta^{k+2}}{1-\delta} \leq \frac{1}{1-\delta}.$$

It combining with (3.13) yields

$$\sum_{k=0}^{\infty}\alpha^k\|d^{k1}\|^2 < +\infty.$$

Putting (3.12) into above equation gives

$$\sum_{k=0}^{\infty}\min\left(t, t\bar{\beta}\epsilon^k, \bar{\alpha}\|d^{k1}\|^2\right)\|d^{k1}\|^2 < +\infty,$$

and then

$$\lim_{k\to\infty}\|d^{k1}\| = 0 \text{ or } \lim_{k\to\infty}\epsilon^k = \lim_{k\to\infty}r_b^k = 0.$$

This together with Theorem 3.2 yields that any accumulation point of $\{y^k\}_{k\geq 0}$ is a KKT point of the problem (1.4). □

4 Numerical experiments

In this section, we conduct preliminary numerical experiments of Algorithm 2 in MATLAB environment (R2010b). All the tests were conducted on a PC under Windows 7 (64bit) system with Intel Core i5-3230M CPU (2.6GHz) and 4GB memory. Our algorithm terminates if $\|r_b^k\| \leq \varepsilon$ is satisfied. The parameters in our implementation are given as follows:

$$\varepsilon = 10^{-5}, \quad t = 0.2, \quad \sigma = 10^{-4}, \quad \delta = 0.85, \quad y^0 = 0.$$

In addition, we found that if ϵ in $\epsilon^k = \min\{\epsilon, r_g^k\}$ is too large or too small, the performance of our algorithm will be degraded. We choose $\epsilon = 10^{-6}$ in our implementation. Our algorithm with ϵ belonging to $(10^{-7}, 10^{-5})$ can also work well.

In our experiments, we solve the following special least squares semidefinite program

$$\begin{cases} \min & \frac{1}{2}\|X - C\|_F^2 \\ \text{s.t.} & X_{ij} = b_{ij}, \ (i,j) \in \mathcal{B}_e \\ & X_{ij} \geq l_{ij}, \ (i,j) \in \mathcal{B}_l, \\ & X_{ij} \leq u_{ij}, \ (i,j) \in \mathcal{B}_u, \\ & X \in \mathcal{S}_+^n, \end{cases} \tag{4.1}$$

where $\mathcal{B}_e$, $\mathcal{B}_l$ and $\mathcal{B}_u$ are three index subsets of $\{(i,j) | 1 \leq i \leq j \leq n\}$. For this problem, we test instances with random data and those with real data as well.

4.1 Random data matrices

The random test examples are given below.

E1 The matrix C is a random matrix which is generated by Matlab built-in command `rand: C=2.0*rand(n,n)-ones(n,n);C=triu(C)*triu(C,1)'; for i=1:n;C(i,i)=1;end;`

The index sets $\mathcal{B}_e = \{(i,i) | i = 1,\ldots,n\}$ and $\mathcal{B}_l = \mathcal{B}_u = \{(i, \min(i+j, n)) | i = 1,\ldots,n, \ j = 1,\ldots,n_r\}$ where $n_r \leq n$ is an integer. For $(i,i) \in \mathcal{B}_e$, $b_{ii} = 1$, for $(i,j) \in \mathcal{B}_l$, $l_{ij} = -0.1$, and for $(i,j) \in \mathcal{B}_u$, $u_{ij} = 0.1$.

E2 The matrix C and the index set $\mathcal{B}_e$ are the same as in E1. The index sets $\mathcal{B}_l, \ \mathcal{B}_u \subset \{(i,j) | 1 \leq i < j \leq n\}$ consist of the indices of $\min(n_r, n-i)$ randomly generated elements at the ith row of X, $i = 1,\ldots,n$. Similar to E1, we take $b_{ii} = 1$, $(i,i) \in \mathcal{B}_e$, $l_{ij} = -0.2$, $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.2$, $(i,j) \in \mathcal{B}_u$.

For evaluation of its performance, we compare our algorithm (denoted by N-L-BFGS) with the inexact smoothing Newton method (denoted by ISNM) Gao and Sun (2009), the projected BFGS method (denoted by P-BFGS) Kelley (1999), the alternating direction method (ADM) (He et al. 2011; Ye and Yuan 2007) and the two-metric projection method based on the L-BFGS (denoted by TMP-BFGS) Gafni and Bertsekas (1984) with respect to the number of iterations, CPU time and the KKT residuals $\|r_b^k\|$. The codes of ISNM, P-BFGS and TMP-BFGS are available at http://www.math.nus.edu.sg/~matsundf/, https://ctk.math.ncsu.edu/matlab_darts.html and https://www.cs.ubc.ca/~schmidtm/Software/minConf.html, respectively. Since the code of ADM (He et al. 2011; Ye and Yuan 2007) is not available to us, we implemented ADM in Matlab by ourselves. The maximum numbers of iterations for N-LBFGS, ISNM, P-BFGS and ADM are 2000, 500, 2000, 2000 and 5000, respectively. We set $\varepsilon = 10^{-5}$ as the tolerance for all solvers, and set options.method to 'lbfgs' for **minConf_TMP**, where **minConf_TMP** is a Matlab implementation of TMP-BFGS. All default options for other parameters are used for ISNM, P-BFGS, TMP-BFGS and ADM. We remark that the stopping conditions for ADM are quite different from the other four solvers since ADM solves the primal problem (1.1) and the quantities for measuring optimality differs from dual ones used for N-LBFGS, ISNM, P-BFGS and TMP-BFGS, and therefore we do not list the final residuals got from ADM in Tables 1 and 2.

We first test E1 and E2 with different parameters: n and n_r, where n varies from 1000 to 1300, and n_r varies from 200 to 300. All the results are listed in Tables 1 and 2, where Iter, CPU and Res stand for the number of iterations, CPU time and the KKT residuals, respectively. From Tables 1 and 2, we see that N-L-BFGS as well as ISNM solves all the instances of E1 and E2 successfully while P-BFGS and ADM fails to solve all of them (labeled by *) due to reaching their corresponding maximum numbers (2000 and 5000, respectively) of iterations. TMP-BFGS also fails to solve all the instances since the algorithm terminates early for various reasons. For instance, the line search fails during iterations. We also observed from Tables 1 and 2 that N-L-BFGS is a clear winner among the five solvers in terms of CPU time.

Next, we test E1 and E2 with larger n, where n varies from 1500 to 2000, and n_r varies from 200 to 300. Due to failure of P-BFGS, ADM and TMP-BFGS, we now only compare our algorithm with ISNM. All the results are listed in Tables 3 and 4. We observed from Tables 3 and 4 that N-L-BFGS succeeds in solving all the instances of E1 and E2 while ISNM fails to solve two instances of E2 (also labeled by *) due to reaching the maximum number 500 of iterations.

For a clear comparison, we depict the performance plots (Figs. 1, 2, 3, 4, 5) of two solvers in terms of Iter, CPU and Res. Figure 1 shows that the number of iterations N-L-BFGS and ISNM required for each instance of E1 is less than 550, and ISNM needs nearly at most half of the number of iterations than N-L-BFGS does. But Fig. 2 gives another picture that N-L-BFGS wins in terms of CPU time. Especially, as n_r increases, the advantage of N-L-BFGS over CPU time is more evident. The possible reason is that ISNM takes more computational effort in solving generalized Newton equations in the case that m is large. Figures 3 and 4 describe the performance of the two solvers on E2, and we obtain the same conclusion as that observed from Figs. 1 and 2. Besides, we notice in Fig. 4 that the advantage of N-L-BFGS over CPU time is more obvious than that in Fig. 2. From Fig. 5, we can see that the KKT residuals for both N-L-BFGS and ISNM have no much difference except for the plot in the lower right corner of Fig. 5. As a matter of fact, ISNM fails to solve E2 with $n = 1800, 2000$ and $n_r = 300$, and therefore the KKT residuals does not reach the given accuracy. Detailed results can be seen in Tables 3 and 4.

4.2 Real data matrices

In this subsection, we test two instances of (4.1) with real data from financial market. As we know, the correlation matrix (So et al. 2013) is an essential part for calculating value at risk. For the purpose of stress testing, the correlation coefficients between certain underlying assets (group) and other underlying assets change largely to simulate the stress scenario (So et al. 2013). For example, some of correlation coefficients are restricted to some intervals. For this purpose, practitioners in financial industry usually seek an approximation of the restricted

Table 1 Numerical results on E1

n	n_r	m	N-L-BFGS			ISNM			P-BFGS			ADM		TMP-BFGS		
			Iter	CPU(s)	Res	Iter	CPU(s)	Res	Iter	CPU(s)	Res	Iter	CPU(s)	Iter	CPU(s)	Res
1000	200	1.81e+5	372	3.87e+2	9.09e−6	108	8.18e+2	2.75e−6	2001*	3.73e+3*	7.46e−5*	5000*	4.15e+3*	411*	7.28e+2*	8.66e-5*
1000	300	2.56e+5	438	5.13e+2	9.76e−6	181	2.01e+3	3.91e−5	2001*	4.03e+3*	2.74e−4*	5000*	4.14e+3*	420*	7.65e+2*	8.60e−5*
1100	200	2.01e+5	394	5.24e+2	8.21e−6	128	1.19e+3	2.61e−5	2001*	4.74e+3*	2.29e−4*	5000*	5.23e+3*	357*	7.87e+2*	8.76e−5*
1100	300	2.86e+5	453	6.55e+2	8.93e−6	120	1.26e+3	5.12e−6	2001*	5.26e+3*	5.13e−4*	5000*	5.36e+3*	419*	9.59e+2*	8.85e−5*
1200	200	2.21e+5	414	6.70e+2	9.72e−6	111	1.03e+3	5.81e−6	2001*	6.62e+3*	1.58e−3*	5000*	7.00e+3*	376*	1.06e+3*	9.05e−5*
1200	300	3.16e+5	454	8.22e+2	9.57e−6	171	2.33e+3	5.04e−6	2001*	6.39e+3*	6.03e−4*	5000*	8.61e+3*	397*	1.43e+3*	9.55e−5*
1300	200	2.41e+5	429	1.09e+3	9.83e−6	126	1.72e+3	3.23e−6	2001*	9.61e+3*	4.54e−4*	5000*	1.15e+4*	389*	1.79e+3*	7.64e−5*
1300	300	3.46e+5	459	1.31e+3	9.35e−6	192	4.08e+3	4.17e−6	2001*	1.03e+4*	1.00e−3*	5000*	1.24e+4*	453*	2.16e+3*	9.51e−5*

Table 2 Numerical results on E2

n	n_r	m	N-L-BFGS			ISNM			P-BFGS			ADM		TMP-BFGS		
			Iter	CPU(s)	Res	Iter	CPU(s)	Res	Iter	CPU(s)	Res	Iter	CPU(s)	Iter	CPU(s)	Res
1000	200	1.81e+5	390	5.43e+2	8.07e−6	153	1.13e+3	8.16e−6	2001*	5.06e+3*	7.75e−5*	5000*	6.05e+3*	333*	7.71e+2*	5.39e−5*
1000	300	2.56e+5	476	7.38e+2	8.89e−6	269	2.64e+3	5.31e−6	2001*	5.38e+3*	5.68e−5*	5000*	5.64e+3*	366*	8.65e+2*	7.90e−5*
1100	200	2.01e+5	412	7.32e+2	9.01e−6	174	1.63e+3	4.85e−6	2001*	6.49e+3*	3.64e−5*	5000*	7.38e+3*	300*	8.91e+2*	6.74e−5*
1100	300	2.86e+5	451	8.61e+2	9.89e−6	187	1.82e+3	6.27e−5	2001*	6.92e+3*	4.57e−5*	5000*	7.30e+3*	376*	1.13e+3*	6.13e−5*
1200	200	2.21e+5	417	9.07e+2	8.08e−6	147	1.44e+3	2.57e−5	2001*	8.10e+3*	3.08e−5*	5000*	9.68e+3*	321*	1.43e+3*	6.64e−5*
1200	300	3.16e+5	465	1.09e+3	9.95e−6	433	7.43e+3	5.11e−6	2001*	8.45e+3*	9.23e−5*	5000*	9.27e+3*	379*	1.44e+3*	7.48e−5*
1300	200	2.41e+5	440	1.31e+3	9.19e−6	144	1.71e+3	9.43e−6	2001*	1.01e+4*	2.19e−4*	5000*	1.14e+4*	338*	1.50e+3*	5.76e−5*
1300	300	3.46e+5	489	1.35e+3	9.94e−6	474	8.63e+3	7.82e−6	2001*	9.97e+3*	1.67e−4*	5000*	1.12e+4*	463*	2.08e+3*	8.26e−5*

Table 3 Numerical results on E1

n	n_r	m	N-L-BFGS			ISNM		
			Iter	CPU(s)	Res	Iter	CPU(s)	Res
1500	200	2.814e+5	440	1.03e+3	3.415e−6	181	2.45e+3	1.515e−7
1500	250	3.451e+5	460	1.13e+3	4.423e−7	195	3.06e+3	1.982e−6
1500	300	4.064e+5	487	1.24e+3	4.161e−6	207	3.40e+3	2.348e−7
1600	200	3.015e+5	458	1.27e+3	2.225e−6	190	2.99e+3	1.493e−6
1600	250	3.702e+5	454	1.31e+3	2.022e−6	200	3.43e+3	1.485e−7
1600	300	4.365e+5	506	1.52e+3	3.838e−6	219	4.09e+3	7.674e−7
1700	200	3.216e+5	477	1.57e+3	2.387e−6	198	3.67e+3	2.275e−7
1700	250	3.953e+5	480	1.63e+3	1.825e−6	211	4.12e+3	4.238e−6
1700	300	4.666e+5	500	1.77e+3	2.397e−6	303	7.29e+3	5.032e−6
1800	200	3.417e+5	438	1.66e+3	1.993e−6	206	4.25e+3	1.208e−7
1800	250	4.204e+5	497	1.93e+3	1.998e−6	225	4.99e+3	3.499e−6
1800	300	4.967e+5	524	2.11e+3	2.691e−6	244	5.94e+3	3.867e−6
1900	200	3.618e+5	498	2.19e+3	2.658e−6	209	4.83e+3	2.396e−6
1900	250	4.455e+5	535	2.45e+3	4.021e−6	301	8.48e+3	2.407e−6
1900	300	5.268e+5	533	2.51e+3	2.995e−6	337	1.04e+4	1.798e−7
2000	200	3.819e+5	507	2.53e+3	2.737e−6	189	4.64e+3	1.220e−7
2000	250	4.706e+5	524	2.70e+3	3.537e−6	327	1.02e+4	3.762e−7
2000	300	5.569e+5	520	2.78e+3	1.229e−6	349	1.17e+4	1.605e−6

Table 4 Numerical results on E2

n	n_r	m	N-L-BFGS			ISNM		
			Iter	CPU(s)	Res	Iter	CPU(s)	Res
1500	200	2.814e+5	464	1.11e+3	1.768e−6	163	1.73e+3	5.590e−7
1500	250	3.451e+5	473	1.17e+3	1.431e−6	242	3.18e+3	3.367e−7
1500	300	4.064e+5	476	1.25e+3	8.844e−7	273	3.90e+3	2.175e−7
1600	200	3.015e+5	672	1.93e+3	1.654e−6	185	2.39e+3	2.162e−7
1600	250	3.702e+5	714	2.21e+3	1.275e−6	426	7.83e+3	2.079e−7
1600	300	4.365e+5	493	1.49e+3	2.585e−6	292	5.03e+3	2.844e−6
1700	200	3.216e+5	511	1.71e+3	3.154e−6	250	4.21e+3	6.138e−7
1700	250	3.953e+5	975	3.61e+3	3.073e−6	400	8.42e+3	5.107e−7
1700	300	4.666e+5	524	1.85e+3	1.303e−6	316	6.23e+3	4.093e−7
1800	200	3.417e+5	482	1.85e+3	5.067e−6	252	4.80e+3	2.651e−6
1800	250	4.204e+5	514	2.05e+3	1.311e−6	279	5.75e+3	2.795e−7
1800	300	4.967e+5	508	2.07e+3	3.963e−6	500*	1.35e+4*	1.292e−3*
1900	200	3.618e+5	490	2.17e+3	2.061e−6	469	1.31e+4	2.139e−7
1900	250	4.455e+5	1050	5.09e+3	1.758e−6	411	1.15e+4	1.155e−6
1900	300	5.268e+5	524	2.46e+3	1.542e−6	338	8.49e+3	2.850e−7
2000	200	3.819e+5	502	2.56e+3	1.426e−6	271	6.80e+3	1.548e−6
2000	250	4.706e+5	544	2.91e+3	2.975e−6	347	9.77e+3	3.419e−6
2000	300	5.569e+5	545	2.90e+3	2.067e−6	500*	1.85e+4*	1.056e−2*

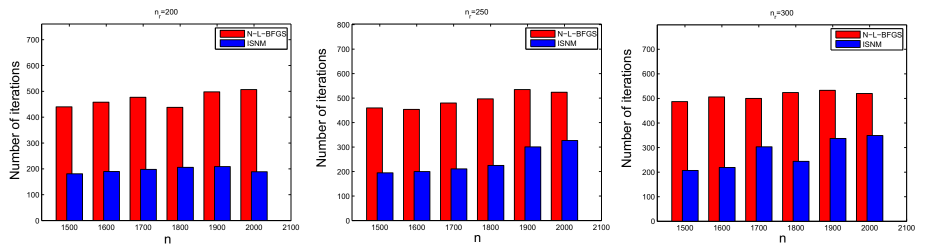

Fig. 1 Performance profile on E1 for iterations

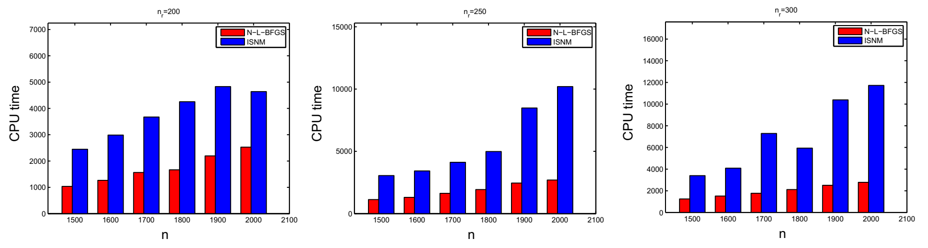

Fig. 2 Performance profile on E1 for CPU time

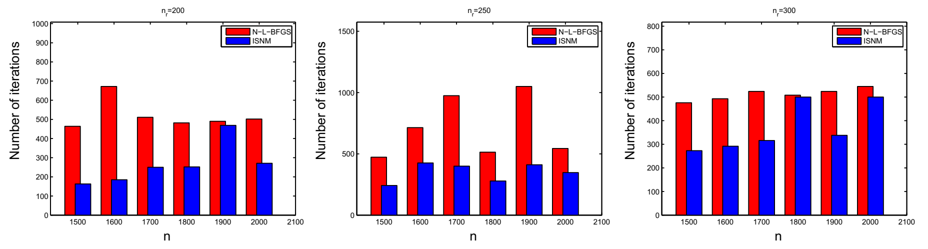

Fig. 3 Performance profile on E2 for iterations

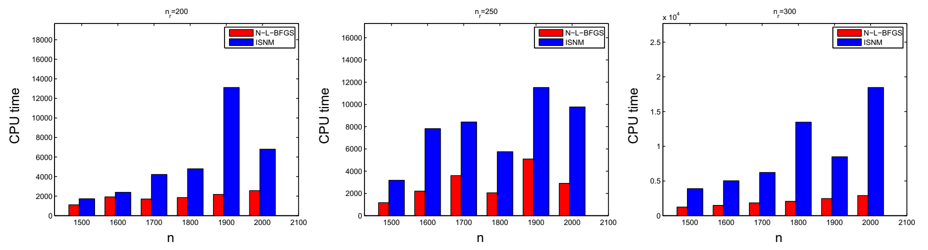

Fig. 4 Performance profile on E2 for CPU time

correlation matrix in stress testing scenario. It is mathematically equivalent to solving the model (4.1).

In our experiment, the correlation matrices to be approximated are calculated from the sample data in the Shenzhen Stock Exchange and the Shanghai Stock Exchange. We remark that the specified restrictions may correspond to an historical stressful event (such as the 1987 stock market crash and 2008 economic crisis) or may be a set of hypothetical changes corresponding to some possible future stressful market event (Kupiec 1998; Han et al. 2017;

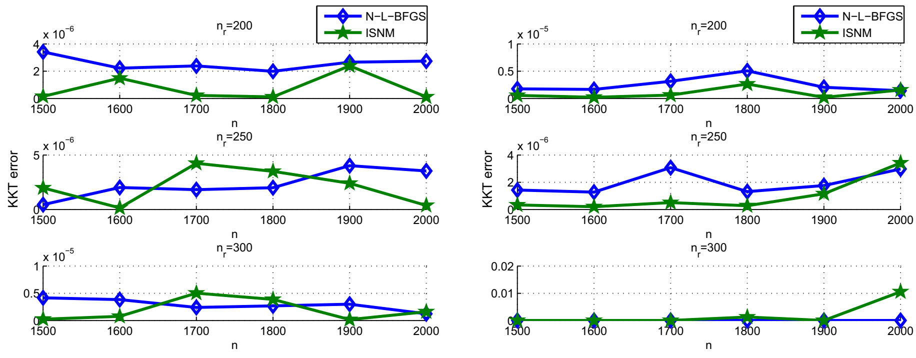

Fig. 5 Comparison on final residuals

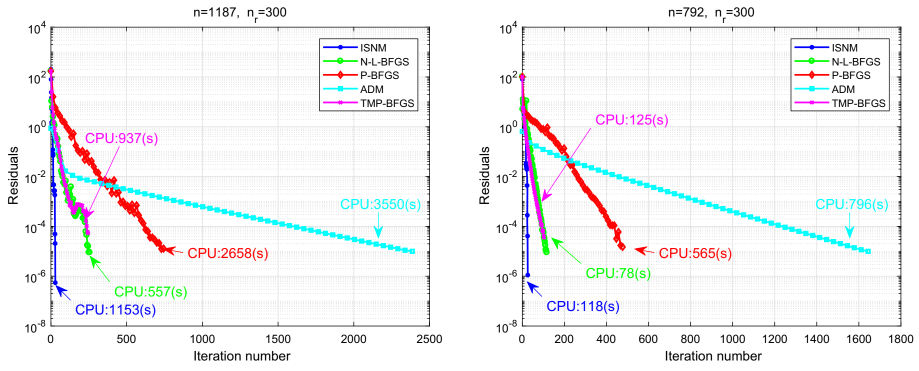

Fig. 6 Performance profile on E3 (left) and E4 (right)

Polanco-Martínez et al. 2018). But, anyhow, identifying the stress events set and restricting the correlation coefficients between stress events and other underlying events is a rather complicated and professional job (Kupiec 1998). For simplicity, we therefore randomly generate the constraints and their positions to imitate the stress testing scenarios.

For the first real example (denoted by E3), the data matrix C is obtained from the daily closing price of 1187 stocks (from September 2016 to September 2018) in the Shenzhen Stock Exchange. We collected the data on the sources of Yahoo by the R-package **quantmod**, and omitted the abnormal data (such as NA or consecutive constant prices due to suspension). We used the remaining 500 observations (the log-daily-return rates) to compute the sample correlation matrix. We randomly specified the positions of entries of the matrix and the corresponding upper and lower bounds to simulate the stress testing scenarios. Specifically, we chose 300 random positions at each row, and set $-0.025 * rand() + 0.025$ and 0.025*rand()+0.025 as the lower and upper bounds for the entries corresponding to these positions, where Matlab built-in function rand is used to generate random numbers.

The data matrix for the second real example (denoted by E4) is obtained from another 792 stocks in the Shanghai Stock Exchange. We used the same method to calculate the log-daily-return rates and then generated the sample correlation matrix. The setting of the positions of entries of the matrix is also the same as the first real example. We restricted all the entries of the sample correlation matrix to be greater than or equal to $-0.10 * rand() + 0.6$.

In this experiment, all solvers except for TMP-BFGS solved both real data problems E3 and E4 successfully within the given tolerance 10^{-5} and their own maximum number of iterations though they took different number of iteration number and cost different CPU time. TMP-BFGS terminated early for some other reasons and its residuals for E3 and E4 are greater than 10^{-5}. To make a clear comparison, we plotted Fig. 6 to show how the residual changes against the number of iterations when applying the solvers to solve E3 and E4, where the residual used for ADM is different from the others but it is borrowed from He et al. (2011). Figure 6 (both the left and right) shows that the residual for ISNM drops very fast as the number of iterations increases. The possible reason is that ISNM is a second-order method which has the superlinear or quadratic rate of convergence. All the other four solvers only use first-order information and they generally require less computational effort per iteration than ISNM. As the iteration goes, the residual for N-L-BFGS decreases much faster than that for P-BFGS and ADM. The residual for TMP-BFGS drops as fast as that for N-L-BFGS, but eventually terminates early due to various reasons (for instance, the failure of the line search). However, the behavior of the residual against the number of iterations cannot fully reflect the performance since the solvers may take different computational amount and cost different CPU time at each iteration. From Fig. 6, we see that N-L-BFGS takes the least CPU time, followed by TMP-BFGS, ISNM, P-BFGS and ADM in turn. Surprisingly, P-BFGS performs much worse than N-L-BFGS although both of them are based on the L-BFGS update. It may be that the active set strategy of P-BFGS slows down the convergence speed. Overall, compared with ISNM, P-BFGS, ADM and TMP-BFGS, our algorithm is a good choice for solving the problem (4.1).

5 Conclusion

In this paper, we proposed an active-set L-BFGS algorithm to solve the matrix nearness problem with some linear equalities, inequalities and a positive semidefinite constraint. Due to the convexity of the problem, it has zero duality gap between the primal and dual problems if the Slater constraint qualification is satisfied. To deal with large-scale dual problems, we employed the active set technique to estimate the active set, and applied L-BFGS method to accelerate free variables. Under some mild conditions, we proved that the proposed algorithm is globally convergent. We finally conducted some preliminary numerical experiments on random and real data and compared our algorithm with the inexact smoothing Newton method Gao and Sun (2009), the projected BFGS method Kelley (1999), the alternating direction method (He et al. 2011; Ye and Yuan 2007) and the two-metric projection method based on the L-BFGS Gafni and Bertsekas (1984). The results on both random data and real data show that our algorithm has some advantages in terms of CPU time if m is large enough.

Acknowledgements The authors would like to thank the Associate Editor and anonymous referees for their helpful suggestions.

References

Borsdorf R, Higham NJ (2010) A preconditioned Newton algorithm for the nearest correlation matrix. IMA J Numer Anal 30:94–107

Boyd S, Xiao L (2005) Least squares covariance matrix adjustment. SIAM J Matrix Anal Appl 27:532–546

Byrd RH, Lu P, Nocedal J, Zhu C (1995) A limited memory algorithm for bound constrained optimization. SIAM J Sci Comput 16:1190–1208

Cristofari A, De Santis M, Lucidi S, Rinaldi F (2017) A two-stage active-set algorithm for bound-constrained optimization. J Optim Theory Appl 172(2):369–401

Dykstra RL (1983) An algorithm for restricted least squares regression. J Am Stat Assoc 78:837–842

FEAST solver (2009–2015). http://www.feast-solver.org/. Accessed 17 June 2018

Gabay D (1983) Application of the method of multipliers to variational inequalities. In: Fortin M, Glowinski R (eds) Augmented Lagrangian methods: application to the numerical solution of boundary-value problems. North-Holland, Amsterdam, pp 299–331

Gabay D, Mercier B (1976) A dual algorithm for the solution of nonlinear variational problems via finite element approximations. Comput Math Appl 2:17–40

Gafni EM, Bertsekas DP (1984) Two-metric projection methods for constrained optimization. SIAM J Control Optim 22(6):936–964

Gao Y, Sun DF (2009) Calibrating least squares semidefinite programming with equality and inequality constraints. SIAM J Matrix Anal Appl 31:1432–1457

Hager WW, Zhang H (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim 16:170–192

Hager WW, Zhang H (2006) A new active set algorithm for box constrained optimization. SIAM J Optim 17:526–557

Han RQ, Xie WJ, Xiong X, Zhang W, Zhou WX (2017) Market correlation structure changes around the great crash: a random matrix theory analysis of the Chinese stock market. Fluct Noise Lett 16(2):1750018

He BS, Xu MH, Yuan XM (2011) Solving large-scale least squares semidefinite programming by alternating direction methods. SIAM J Matrix Anal Appl 32:136–152

Higham NJ (2002) Computing the nearest correlation matrix-a problem from finance. IMA J Numer Anal 22:329–343

Kelley CT (1999) Iterative methods for optimization. SIAM, Philadelphia, pp 102–104

Kupiec PH (1998) Stress testing in a value-at-risk framework. J Deriv 6:7–24

Li QN, Li DH (2011) A projected semismooth Newton method for problems of calibrating least squares covariance matrix. Oper Res Lett 39:103–108

Lin C-J, Moré JJ (1999) Newtons method for large bound-constrained optimization problems. SIAM J Optim 9:1100C1127

Liu DC, Nocedal J (1989) On the limited memory BFGS method for large-scale optimization. Math Program 45:503–528

Malick J (2004) A dual approach to semidefinite least squares problems. SIAM J Matrix Anal Appl 26:272–284

Ni Q, Yuan Y (1997) A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization. Math Comp 66:1509–1520

Nocedal J, Wright SJ (2006) Numerical optimization, vol 2. Springer, New York

Polanco-Martínez JM, Fernández-Macho J, Neumann MB, Faria SH (2018) A pre-crisis vs. crisis analysis of peripheral EU stock markets by means of wavelet transform and a nonlinear causality test. Phys A 490:1211–1227

Polizzi E (2009) Density-matrix-based algorithms for solving eigenvalue problems. Phys Rev B 79:115112

Qi HD, Sun D (2006) A quadratically convergent Newton method for computing the nearest correlation matrix. SIAM J Matrix Anal Appl 28:360–385

Qi HD, Sun D (2010) Correlation stress testing for value-at-risk: an unconstrained convex optimization approach. Comput Optim Appl 45:427–462

Rahpeymaii F, Kimiaei M, Bagheri A (2016) A limited memory quasi-Newton trust-region method for box constrained optimization. J Comput Appl Math 303:105–118

Rebonato R, Jäckel P (1999) The most general methodology for creating a valid correlation matrix for risk management and option pricing purposes. J Risk 2:17–27

Rockafellar RT (1974) Conjugate duality and optimization. BMS-NSF Regional Conference Series in Applied Mathematics 16, SIAM, Philadelphia

Schwertman NC, Allen DM (1979) Smoothing an indefinite variance-covariance matrix. J Stat Comput Simul 9:183–194

So MKP, Wong J, Asai M (2013) Stress testing correlation matrices for risk management. N Am J Econ Finan 26:310–322

Sun YF, Vandenberghe L (2015) Decomposition methods for sparse matrix nearness problems. SIAM J Matrix Anal Appl 36:1691–1717

Werner R, Schöttle K (2007) Calibration of correlation matrices-SDP or not SDP. Technical report, Munich University of Technology, Munich

Xiao Y, Wei Z (2007) A new subspace limited memory BFGS algorithm for large-scale bound constrained optimization. Appl Math Comput 185(1):350–359

Xiao Y, Li DH (2008) An active set limited memory BFGS algorithm for large-scale bound constrained optimization. Math Methods Oper Res 67(3):443–454

Xue WJ, Shen CG (2018) Limited memory BFGS method for least squares semidefinite programming with banded structure. Technical Report, University of Shanghai for Science and Technology, Shanghai

Yuan G, Lu X (2011) An active set limited memory BFGS algorithm for bound constrained optimization. Appl Math Model 35(7):3561–3573

Yuan G, Wei Z (2010) Convergence analysis of a modified BFGS method on convex minimizations. Comput Optim Appl 47:237–255

Ye CH, Yuan XM (2007) A descent method for structured monotone variational inequalities. Optim Methods Softw 22:329–338

Zarantonello EH (1971) Projections on convex sets in Hilbert space and spectral theory I and II. In: Zarantonello EH (ed) Contributions to nonlinear functional analysis. Academic Press, New York, pp 237–424

Zhang HC, Hager WW (2004) A nonmonotone line search technique and its application to unconstrained optimization. SIAM J Optim 14:1043–1056