Computational Optimization and Applications (2021) 78:1–42
https://doi.org/10.1007/s10589-020-00228-5

An accelerated active-set algorithm for a quadratic semidefinite program with general constraints

Chungen Shen[1] · Yunlong Wang[2] · Wenjuan Xue[3] · Lei-Hong Zhang[4]

Received: 9 September 2019 / Accepted: 12 September 2020 / Published online: 27 September 2020

Abstract

In this paper, we are concerned with efficient algorithms for solving the least squares semidefinite programming which contains many equalities and inequalities constraints. Our proposed method is built upon its dual formulation and is a type of active-set approach. In particular, by exploiting the nonnegative constraints in the dual form, our method first uses the information from the Barzlai–Borwein step to estimate the active/inactive sets, and within an adaptive framework, it then accelerates the convergence by switching the L-BFGS iteration and the semi-smooth Newton iteration dynamically. We show the global convergence under mild conditions, and furthermore, the local quadratic convergence under the additional nondegeneracy condition. Various types of synthetic as well as real-world examples are tested, and preliminary but promising numerical experiments are reported.

Keywords Semidefinite programs · Active set · Barzlai–Borwein step · L-BFGS · Semi-smooth Newton

Mathematics Subject Classification 65K05 · 95C55 · 90C30

Wenjuan Xue: The work of this author was supported in part by the National Natural Science Foundation of China NSFC-11601318. Lei-Hong Zhang: The work of this author was supported in part by the National Natural Science Foundation of China (NSFC-11671246, NSFC-12071332), the National Key R&D Program of China (No. 2018YFB0204404) and Double Innovation Program of Jiangsu Province, Year 2018.

✉ Chungen Shen
shenchungen@usst.edu.cn

Lei-Hong Zhang
longzlh@suda.edu.cn

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

[2] Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China

[3] School of Mathematics and Physics, Shanghai University of Electric Power, Shanghai 200090, China

[4] School of Mathematical Sciences, Soochow University, Suzhou 215006, Jiangsu, China

1 Introduction

Let $\mathcal{S}^n_+$ be the cone of positive semidefinite matrices in the space $\mathcal{S}^n$ of $n \times n$ symmetric matrices, and $\langle A, B\rangle := \mathrm{tr}(A^T B)$ with $A, B \in \mathcal{S}^n$. In this paper, we consider the least squares semidefinite programming (LSSDP) [2, 10, 12, 17, 27] of the following form

$$\begin{array}{ll} \min & \frac{1}{2}\|X - G\|_F^2 \\ s.t. & \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, p, \\ & \langle A_i, X\rangle \geq b_i, \quad i = p+1, \ldots, m, \\ & X \in \mathcal{S}^n_+, \end{array} \tag{1.1}$$

where $b = (b_1, \ldots, b_m)^T \in \mathbb{R}^m$, and $G, A_i \in \mathcal{S}^n$ for $i = 1, \ldots, m$ are all given.

To solve the problem (1.1) numerically, in the literature, a couple of methods have been proposed and can be used in different situations. For instance, for small- to medium-size n and m, interior point methods implemented in, for example, SeDuMi [31] and SDPT3 [34], can solve efficiently the dual problem (1.2)

$$\begin{array}{ll} \min & t \\ s.t. & \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, p, \\ & \langle A_i, X\rangle \geq b_i, \quad i = p+1, \ldots, m, \\ & t \geq \frac{1}{2}\|X - G\|_F^2, \\ & X \in \mathcal{S}^n_+, \end{array} \tag{1.2}$$

which is just to minimize a simple linear function subject to the constraints of linear equalities/inequalities, the second order cone, and the positive semidefinite cone. However, since a linear system associated with a dense Schur complement matrix of the size $(m + 1 + \bar{n}) \times (m + 1 + \bar{n})$ with $\bar{n} := \frac{1}{2}n(n+1)$ needs to be solved at each iteration, the efficiency of interior point methods decreases as n gets moderately larger (say, 2000). In the literature, therefore, methods other than interior point methods have been proposed and applied. For example, Malick [17], Boyd and Xiao [2], and Gao and Sun [10] proposed the BFGS method, the projected gradient method, and semi-smooth Newton method respectively, to solve the Lagrangian dual problem of (1.1). Along the dual framework, Li and Li [15] introduced a projected semi-smooth Newton method. Unlike [2, 10, 17], He, Xu and Yuan [12] suggested the augmented Lagrangian function with an auxiliary matrix variable and then applied the alternating direction methods (ADM) [8, 9, 35] to (1.1). Very recently, Sun and Vandenberghe [33] considered decomposition methods for matrix nearness problems with some sparsity pattern.

We remark that each of the previously mentioned methods handles either the computational and memory costs per step or the fast local convergence. For example, the projected gradient method [2] and the ADM methods [12] usually are of economic computational costs at each step, but they have relatively slow convergence,

especially when the iterates reach in the vicinity of a minimizer. The BFGS method [17] and the semi-smooth Newton method [10] locally converge fast, but the approximate Hessian matrix in the BFGS method has to be stored at each step, while the generalized Newton equation has to be solved in the semi-smooth Newton method, which degrades the efficiency of the related method when m gets very large.

In this paper, relying on the dual technique, we attempt to handle both the computational and memory costs per step and the local fast convergence. In particular, we combine the advantages of limited-BFGS (L-BFGS) and the semi-smooth Newton method to propose an accelerated active-set method for the dual problem. In our first stage, the Barzlai–Borwein (BB) step is used to draw information of the index of the final active/inactive sets. These active/inactive sets correspond to free/fixed variables. The second stage employs the L-BFGS method and the semi-smooth Newton method to accelerate the convergence of these free variables. For a practical implementation, we propose an adaptive strategy to smoothly switch them so that it adaptively refines the estimated index of the final active/inactive sets, and also is able to use the full-power of the semi-smooth Newton method whenever the iterates are close to the minimizer. We establish the global convergence and the fast local convergence under some mild conditions.

The rest of this paper is organized as follows. In Sect. 2, we state the dual problem and give the first-order/second-order optimality conditions. In Sect. 3, we present the details of our accelerated active-set method by describing four main ingredients: identification of active/inactive sets, the L-BFGS acceleration, the semi-smooth Newton acceleration, and an adaptive acceleration strategy. The global convergence of the proposed algorithm is proved under some mild conditions in Sect. 4, and fast local convergence is established in Sect. 5. Numerical evaluation of the proposed method is conducted in Sect. 6, where LSSDP instances on both real and synthetic data are tested, and the performance of other solvers [10, 12, 14, 35] are reported and compared. Final remarks are drawn in Sect. 7.

2 The dual problem and optimality conditions

We first rewrite (1.1) into the following general form:

$$\begin{array}{ll} \min & \frac{1}{2}\|X-G\|_F^2 \\ s.t. & \mathcal{A}(X) \in b + \mathcal{Q}, \\ & X \in \mathcal{K}, \end{array} \tag{2.1}$$

where $\mathcal{Q} = \{0\} \times \mathbb{R}_+^{m-p}$, $\mathcal{K}$ is a closed and convex cone, and $\mathcal{A}(X)$ is defined as

$$\mathcal{A}(X) = \begin{pmatrix} \langle A_1, X\rangle \\ \langle A_2, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{pmatrix}.$$

Note that the primal form (2.1) is convex and the associated Lagrangian function is given by

$$\mathcal{L}(X, y) = \frac{1}{2}\|X - G\|_F^2 - \langle \mathcal{A}(X) - b, y\rangle,$$

where $\langle x, y\rangle := x^T y$ with $x, y \in \mathbb{R}^n$.

2.1 The dual problem

The Lagrangian dual of (2.1) (see [10]) is

$$\begin{aligned} \max \quad & -\frac{1}{2}\|\Pi_{\mathcal{K}}[G + \mathcal{A}^*(y)]\|_F^2 + \langle b, y\rangle + \frac{1}{2}\langle G, G\rangle \\ s.t. \quad & y \in \mathcal{Q}^*, \end{aligned} \tag{2.2}$$

where $\mathcal{Q}^* = \mathbb{R}^p \times \mathbb{R}_+^{m-p}$ is the dual cone of $\mathcal{Q}$, $\mathcal{A}^*$ is the adjoint operator of $\mathcal{A}$ defined by $\mathcal{A}^*(y) = \sum_{i=1}^m y_i A_i$, and $\Pi_{\mathcal{K}}[Y]$ denotes the metric projection of $Y \in \mathcal{S}^n$ onto $\mathcal{K}$, i.e.,

$$\Pi_{\mathcal{K}}[Y] = \text{argmin}_{X \in \mathcal{K}} \langle X - Y, X - Y\rangle.$$

Denote

$$f(y) = \frac{1}{2}\|\Pi_{\mathcal{K}}[(G + \mathcal{A}^*(y))]\|_F^2 - \langle b, y\rangle - \frac{1}{2}\langle G, G\rangle.$$

Obviously, the dual problem (2.2) is equivalent to the problem

$$\begin{aligned} \min \quad & f(y) \\ s.t. \quad & y \in \mathcal{Q}^*. \end{aligned} \tag{2.3}$$

For $Y \in \mathcal{S}^n$, let the spectral decomposition be $Y = P\Lambda P^T$ with $\Lambda := \text{diag}(\lambda_1, \ldots, \lambda_n)$, where $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of Y and P is a corresponding orthogonal matrix of orthonormal eigenvectors of Y. Note that this decomposition needs $9n^3$ multiplications or so. From [26], we have

$$\Pi_{\mathcal{S}_+^n}[Y] = P\text{diag}(\max(0, \lambda_1), \ldots, \max(0, \lambda_n))P^T.$$

Hence, for (1.1), $\mathcal{K} = \mathcal{S}_+^n$, $\mathcal{A}^*(y) = \sum_{i=1}^m y_i A_i$, and

$$f(y) = \frac{1}{2}\left\|[G + \mathcal{A}^*(y)]_+\right\|_F^2 - \langle b, y\rangle - \frac{1}{2}\langle G, G\rangle,$$

where $[\cdot]_+ = \Pi_{\mathcal{S}_+^n}[\cdot]$. The function $f(y)$ is convex, continuously differentiable and coercive (i.e., $f(y) \to \infty$ as $\|y\| \to \infty$) [25] under the Slater condition. The gradient is given by

$$\nabla f(y) = \mathcal{A}[G + \mathcal{A}^*(y)]_+ - b, \quad y \in \mathbb{R}^m, \tag{2.4}$$

which is also Lipschitz continuous. Moreover, under the Slater constraint qualification, it has zero duality gap between the primal problem (1.1) and its dual (2.3), which provides a possibility to solve (1.1) from the dual. We point out that, due to the projection $\Pi_{S_+^n}[\cdot]$, $\nabla f(y)$ generally is not differentiable, and thus, the classical Newton method is not applicable for (2.3).

2.2 Optimality conditions

For convenience, we denote the index sets of equalities and inequalities of (1.1) by $\mathcal{E} := \{1, \ldots, p\}$ and $\mathcal{I} := \{p+1, \ldots, m\}$, respectively. Also, we denote $\nabla f(y)$ by $g(y)$ and the subvector of $g(y)$ with components $g_i(y), i \in \mathcal{E}$ by $g_\varepsilon(y)$. The meanings of $g_\mathcal{I}(y)$ and $y_\mathcal{I}$ follow similarly.

Due to the special structure of the constraints $y_i \geq 0$ with $i \in \mathcal{I}$, the Mangasarian-Fromovitz constraint qualification (MFCQ) is satisfied at any feasible point of (2.3). The first-order condition (i.e., the KKT condition) of (2.3) is

$$\begin{cases} g_\varepsilon(y) = 0, \\ g_\mathcal{I}(y) - \mu = 0, \\ \mu \geq 0, \quad \mu^T y_\mathcal{I} = 0, \quad y_\mathcal{I} \geq 0, \end{cases} \tag{2.5}$$

where $\mu = (\mu_1, \ldots, \mu_{m-p})^T \in \mathbb{R}^{m-p}$ is the multiplier vector corresponding to the constraints $y_i \geq 0, i \in \mathcal{I}$. If a pair (y, μ) satisfies (2.5), we call it a KKT pair. One way to measure the KKT error is the KKT residual

$$\Psi(y, \mu) = \left\| \begin{pmatrix} g_\varepsilon(y) \\ g_\mathcal{I}(y) - \mu \\ \min(y_\mathcal{I}, \mu) \end{pmatrix} \right\|,$$

where $\min(y_\mathcal{I}, \mu)$ is the componentwise minimum function of the vectors μ and $y_\mathcal{I}$. It is easy to verify that $\Psi(y^*, \mu^*) = 0$ if and only if (y^*, μ^*) is a KKT pair for (2.3).

We now suppose $\Psi(y^*, \mu^*) = 0$, and let

$$\begin{aligned} \mathcal{I}_0(y^*) &= \{i \in \mathcal{I} | y_i^* = 0\}, \\ \mathcal{I}_+(y^*) &= \{i \in \mathcal{I}_0 | \mu_i^* > 0\}, \\ \mathcal{I}_{00}(y^*) &= \{i \in \mathcal{I}_0 | \mu_i^* = 0\}, \\ \mathcal{H}_\theta(y^*) &= \{h \in \mathbb{R}^m | h_i = 0, \quad i \in \mathcal{I}_+(y^*); \quad h_i \geq 0, \quad i \in \mathcal{I}_{00}(y^*)\}, \end{aligned}$$

and

$$\bar{\mathcal{H}}_\theta(y^*) = \{h \in \mathbb{R}^m | h_i = 0, \quad i \in \mathcal{I}_0(y^*)\}.$$

The (strong) second-order sufficient condition holds at y^* if $h^T V h > 0$ for all nonzero vectors $h \in \mathcal{H}_\theta(y^*)$ ($h \in \bar{\mathcal{H}}_\theta(y^*)$) and all matrices $V \in \partial_B(\nabla f(y^*))$, where $\partial_B(\nabla f(y^*))$ denotes the B-subdifferential of $\nabla f(y)$ at y^* in the sense of Qi [21]. In [22,

Theorem 2.2], it is shown that y^* is a strict local minimum of (2.3) if the (strong) second-order sufficient condition holds.

3 Algorithm

In this section, we will present our accelerated active set algorithm for solving the dual problem (2.3). Our approach consists of a procedure of active set detection by the BB step [1], and a smoothly-switching search direction procedure between L-BFGS and the semi-smooth Newton methods. The BB step aims at estimating the index of the active/inactive sets (free/fixed variables), and whenever this stage is fulfilled, the L-BFGS method or the semi-smooth Newton method will be triggered to accelerate the convergence of the free variables.

3.1 The BB step

Originated by Barzlai and Borwein [1], the BB method has been discussed by many other researchers and applied in a broad area of applications. In our case, at the current iterate y^k, we first use the BB step to generate a trial iterate $\tilde{y}^k$ which is called a BB point. There are two commonly used formula of the stepsize (see [1]), namely,

$$\alpha_l^k = \frac{\|s^{k-1}\|_2^2}{(s^{k-1})^T w^{k-1}}, \quad \text{(the long BB stepsize)}$$
$$\alpha_s^k = \frac{(s^{k-1})^T w^{k-1}}{\|w^{k-1}\|_2^2}, \quad \text{(the short BB stepsize)}$$

where $s^{k-1} = y^k - y^{k-1}$, $w^{k-1} = g(y^k) - g(y^{k-1})$. It has been proved in theory that the long stepsize α_l^k can guarantee the reduction of the objective function, and the short stepsize α_s^k can induce a good descent direction for the next iteration. Based on this fact, Zhou, Gao and Dai [37] present an adaptive stepsize selection strategy and report numerical results for its efficiency. Specifically, such adaptive stepsize with some safeguards is defined as

$$\alpha_B^k = \begin{cases} \min(\alpha_{\max}, \max(\alpha_{\min}, \alpha_s^k)), & \alpha_s^k/\alpha_l^k \le \kappa; \\ \min(\alpha_{\max}, \max(\alpha_{\min}, \alpha_l^k)), & \text{otherwise}, \end{cases} \tag{3.1}$$

where $\kappa \in (0, 1)$, and $\alpha_{\max} > \alpha_{\min} > 0$. The adaptive stepsize can be viewed as a combination of the long BB stepsize and the short BB stepsize, and the parameter κ determines the trade-off between α_l^k and α_s^k. Generally, κ is around 0.5. Other adaptive strategies are discussed in [5, 37].

In our case, we use the strategy in [37] to generate the BB point. In particular, for the dual problem (2.3), we generate the trial BB point as

$$\tilde{y}^k = \Pi_{\mathcal{Q}}[y^k - \alpha_B^k g^k], \tag{3.2}$$

where $g^k := g(y^k)$, and the stepsize is determined by the adaptive stepsize α_B^k in (3.1).

3.2 Identification of the active set

In this subsection, we will discuss how to detect the active/inactive sets at the trial BB point $\tilde{y}^k$. Assume that y^* is a minimizer of the problem (2.3). As the linear independence constraint qualification (LICQ) holds at y^*, the uniqueness of the associated multiplier μ^* is ensured. Moreover, when the second-order sufficient condition holds, y^* is isolated (see [22, Theorem 2.2]). According to [7, Theorem 3.6], we have the following theorem.

Theorem 3.1 *Assume that the second-order sufficient condition holds at y^*. Then there exist two scalars $\kappa_1, \kappa_2 > 0$ such that*

$$\kappa_1 \left\| \begin{pmatrix} y - y^* \\ \mu - \mu^* \end{pmatrix} \right\| \le \Psi(y, \mu) \le \kappa_2 \left\| \begin{pmatrix} y - y^* \\ \mu - \mu^* \end{pmatrix} \right\|$$

for any pair (y, μ) sufficiently close to (y^*, μ^*).

Using [7, Theorems 3.7 and 2.2], the index set

$$\bar{\mathcal{B}}(y, \mu) = \{i \in \mathcal{I} | y_i \le \sqrt{\Psi(y, \mu)}\}$$

correctly identifies all active constraints if (y, μ) is sufficiently close to (y^*, μ^*). To apply this fact, by canceling the multiplier μ in (2.5), we rewrite the KKT condition (2.5) as

$$\begin{cases} g_{\mathcal{E}}(y) = 0, \\ g_{\mathcal{I}}(y) \ge 0, \quad g_{\mathcal{I}}(y)^T y_{\mathcal{I}} = 0, \quad y_{\mathcal{I}} \ge 0, \end{cases} \tag{3.3}$$

and by $\mu = g_{\mathcal{I}}(y)$, we have

$$\Psi(y, \mu) = \Psi(y, g_{\mathcal{I}}(y)) = \|y - \Pi_{\mathcal{Q}}[y - g(y)]\| := \varrho(y). \tag{3.4}$$

Here $\Pi_{\mathcal{Q}}[x]$ denotes the metric projection of $x \in \mathbb{R}^m$ onto $\mathcal{Q}$, which can be easily calculated as

$$\Pi_{\mathcal{Q}}[y] = \begin{cases} 0, & i \in \mathcal{I} \text{ with } y_i < 0, \\ y_i, & \text{otherwise}. \end{cases} \tag{3.5}$$

In our algorithm, at the BB point $\tilde{y}^k$ in (3.2), we define two index sets $\mathcal{B}^k$ and $\mathcal{F}^k$ associated with the point y^k as

$$\begin{cases} \mathcal{B}^k = \mathcal{B}(y^k) = \{i \in \mathcal{I} | 0 \le \tilde{y}_i^k \le \nu^k, \text{ or } g_i^k \le -\nu^k/(2\alpha_B^k) \quad \& \quad y_i^k \le \nu^k/2\} \text{ and} \\ \mathcal{F}^k = \mathcal{F}(y^k) = \{1, \ldots, m\} \backslash \mathcal{B}^k, \end{cases} \tag{3.6}$$

where $\nu^k = \min\{\epsilon_0, \sqrt{\varrho^k}\}$, $\epsilon_0 \in (0, 0.1)$ (generally, ϵ_0 is a small parameter of order, e.g., 10^{-6}), and $\varrho^k = \varrho(y^k)$. One can see from the definition of $\mathcal{B}^k$ that components of $\tilde{y}^k$ and y^k that approach the boundary are taken as the potential fix variables, and therefore, the corresponding indices are put in $\mathcal{B}^k$. More explanations will be stated in Sect. 4.

As $\mathcal{B}^k$ differs slightly from $\bar{\mathcal{B}}(y^k, \mu^k)$, we find that [7, Theorems 3.7, 2.2] is not directly applicable to $\mathcal{B}^k$ to detect the active set correctly. However, such a desired conclusion still holds as shown in the following theorem.

Theorem 3.2 *Assume that the second-order sufficient condition holds at y^*. Then $\mathcal{B}^k$ and $\mathcal{F}^k$ can correctly identify all active/inactive constraints respectively if y^k is sufficiently close to y^*.*

Proof It suffices to prove that $\mathcal{B}^k$ can correctly identifies all active/inactive constraints if y^k is close to y^* enough. By the definition of ϱ^k and (3.5),

$$\varrho^k = \varrho(y^k) = \left\| \begin{pmatrix} g_{\mathcal{E}}(y^k) \\ \min(y^k_{\mathcal{I}}, g_{\mathcal{I}}(y^k)) \end{pmatrix} \right\|. \tag{3.7}$$

From (3.2) and (3.5), we obtain that for $i \in \mathcal{I}$,

$$\tilde{y}^k_i = \max(y^k_i - \alpha^k_B g^k_i, 0) = y^k_i - \min(y^k_i, \alpha^k_B g^k_i). \tag{3.8}$$

Suppose $y^*_i = 0$ for some $i \in \mathcal{I}$. Due to Theorem 3.1 and $\varrho(y) = \Psi(y, \mu)$, it holds

$$\|y^k - y^*\| = o(\sqrt{\varrho^k}). \tag{3.9}$$

It follows with the definition of ν^k that $y^k_i \le \|y^k - y^*\| \le \nu^k/2$ for y^k close to y^* enough. By (3.7) and the definition of ν^k, it holds $|\min(y^k_i, \alpha^k_B g^k_i)| \le \nu^k/2$. This together with (3.8) and $y^k_i \le \nu^k/2$ yields $\tilde{y}^k_i \le \nu^k$. So $i \in \mathcal{B}^k$.

On the other hand, if $y^*_i > 0$ for some $i \in \mathcal{I}$, then $g_i(y^*) = 0$ due to the KKT condition (3.3). By (3.8), $\tilde{y}^k_i > \nu^k$ if y^k is sufficiently close to y^*. By (3.9) and the definition of ν^k, we have

$$\|y^k - y^*\| < \frac{\nu^k}{2L_g \alpha^k_B}$$

for y^k sufficiently close to y^*, where $\alpha^k_B \in [\alpha_{\min}, \alpha_{\max}]$ is the BB stepsize from (3.1) and $L_g > 0$ is the Lipschitz constant of $g(y)$. This combining with the Lipschitz continuity of $g(y)$ gives

$$|g^k_i| = |g^k_i - g_i(y^*)| \le L_g \|y^k - y^*\| < \nu^k/(2\alpha^k_B).$$

Thus, $-g^k_i < \nu^k/(2\alpha^k_B)$ for any y^k sufficiently close to y^*. So, $i \notin \mathcal{B}^k$ and consequently, $\mathcal{B}^k$ identifies all active constraints accurately. □

Theorem 3.2 reveals that the problem (2.3), with the second-order sufficient condition (not the strict complementarity condition), locally can be reduced to an unconstrained minimization problem. This provides an important foundation for establishing the fast local convergence in Sect. 5.

3.3 The search direction

With the BB point $\tilde{y}^k$ computed by (3.2) at hand, we can determine the working sets $\mathcal{B}^k$ and $\mathcal{F}^k$ by (3.6). Let J^k be a generalized Hessian matrix of $f(y)$ at y^k. Without loss of generality, we assume $\mathcal{F}^k = \{1, 2, \ldots, |\mathcal{F}^k|\}$ with $|\mathcal{F}^k|$ being the cardinality of $\mathcal{F}^k$, and then partition J^k and g^k as

$$J^k = \begin{pmatrix} J^k_{\mathcal{F}^k\mathcal{F}^k} & J^k_{\mathcal{F}^k\mathcal{B}^k} \\ J^k_{\mathcal{B}^k\mathcal{F}^k} & J^k_{\mathcal{B}^k\mathcal{B}^k} \end{pmatrix} \text{ and } g^k = \begin{pmatrix} g^k_{\mathcal{F}^k} \\ g^k_{\mathcal{B}^k} \end{pmatrix}. \tag{3.10}$$

As a result, we present the entry d^k_i of the search direction $d^k = d(y^k)$ associated with y^k as

$$d^k_i = d_i(y^k) = \begin{cases} \tilde{y}^k_i - y^k_i, & i \in \mathcal{B}^k, \\ -(Z\bar{H}^k Z^T g^k)_i, & i \in \mathcal{F}^k, \end{cases} \tag{3.11}$$

where Z is a matrix of columns consisting of $\{e_i | i \in \mathcal{F}^k\}$ with e_i being the ith column of the identity matrix I_n (i.e., $Z = (I_{|\mathcal{F}^k|}, 0)^T \in \mathbb{R}^{n\times|\mathcal{F}^k|}$ when $\mathcal{F}^k = \{1, 2, \ldots, |\mathcal{F}^k|\}$), and $\bar{H}^k$ is $(J^k_{\mathcal{F}^k\mathcal{F}^k})^{-1}$ or a certain approximation of $(J^k)^{-1}_{\mathcal{F}^k\mathcal{F}^k}$; here $(J^k)^{-1}_{\mathcal{F}^k\mathcal{F}^k}$ denotes the top-left block of $(J^k)^{-1}$ with $(J^k)^{-1}$ being partitioned similar as J^k.

The matrix $\bar{H}^k$ is a key to define the search direction in (3.11). In the next subsection, we will specify two ways to compute $\bar{H}^k$: the way based on the L-BFGS formula and the one based on the semi-smooth Newton method. Here, we only mention that if the approximation H^k of $(J^k)^{-1}$ is updated by the L-BFGS formula implicitly, then its top-left block $H^k_{\mathcal{F}^k\mathcal{F}^k}$ can be considered as an approximation of $(J^k)^{-1}_{\mathcal{F}^k\mathcal{F}^k}$, and thus $\bar{H}^k = H^k_{\mathcal{F}^k\mathcal{F}^k} = Z^T H^k Z$, whereas if the semi-smooth Newton acceleration is applied, then $\bar{H}^k = (J^k_{\mathcal{F}^k\mathcal{F}^k})^{-1}$. The detailed information on computation of $Z\bar{H}^k Z^T g^k$ will be given in the next subsection.

3.4 The L-BFGS and semi-smooth Newton search direction

3.4.1 The L-BFGS direction

The BFGS method [20, Chapter 7] is a well-known quasi-Newton method for the unconstrained minimization $\min f(y)$. A basic iteration of BFGS is

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

where α^k is a stepsize, and H^k is the inverse Hessian approximation updated by

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

where $\rho^k = 1/(w^k)^T s^k$, $V^k = I - \rho^k w^k (s^k)^T$, and $s^k = y^{k+1} - y^k$, $w^k = g^{k+1} - g^k$.

The L-BFGS algorithm (see [16, 20]) is an improvement of BFGS for large-scale problems which alleviates the memory requirement in storing H^k in each step. In particular, the matrix H^k in the L-BFGS algorithm is formed implicitly by the l most recent vectors pairs $\{s^i, w^i\}$, $i = k-l, \ldots, k-1$. Starting from an initial Hessian approximation[1] $(H^k)^0$, at the kth iteration, it uses the l most recent vector pairs $\{s^i, w^i\}$, $i = k-l, \ldots, k-1$ to have

$$\begin{aligned} H^k =& ((V^{k-1})^T \cdots (V^{k-l})^T)(H^k)^0 (V^{k-l} \cdots V^{k-1}) \\ &+ \rho^{k-l}((V^{k-1})^T \cdots (V^{k-l+1})^T) s^{k-l}(s^{k-l})^T (V^{k-l+1} \cdots V^{k-1}) \\ &+ \rho^{k-l+1}((V^{k-1})^T \cdots (V^{k-l+2})^T) s^{k-l+1}(s^{k-l+1})^T (V^{k-l+2} \cdots V^{k-1}) \\ &+ \cdots \\ &+ \rho^{k-1} s^{k-1}(s^{k-1})^T. \end{aligned} \tag{3.12}$$

According to this formula, an efficient recursive procedure (see [20, Algorithm 7.4]) can be derived to compute $H^k g^k$. In the case of the L-BFGS acceleration, the term $Z\bar{H}^k Z^T g^k$ in (3.11) is

$$Z\bar{H}^k Z^T g^k = Z H^k_{\mathcal{F}^k \mathcal{F}^k} Z^T g^k = \begin{pmatrix} H^k_{\mathcal{F}^k \mathcal{F}^k} g^k_{\mathcal{F}^k} \\ 0 \end{pmatrix}.$$

With the partition $H^k = \begin{pmatrix} H^k_{\mathcal{F}^k \mathcal{F}^k} & H^k_{\mathcal{F}^k \mathcal{B}^k} \\ H^k_{\mathcal{B}^k \mathcal{F}^k} & H^k_{\mathcal{B}^k \mathcal{B}^k} \end{pmatrix}$, we know that the first sub-vector $H^k_{\mathcal{F}^k \mathcal{F}^k} g^k_{\mathcal{F}^k}$ is the same as that of $H^k \begin{pmatrix} g^k_{\mathcal{F}^k} \\ 0 \end{pmatrix}$, and therefore, the part of search direction of d^k in (3.11) for the variables in $\mathcal{F}^k$ can be calculated by the recursive procedure with (3.12) (see [20, Algorithm 7.4]). Note that this recursive procedure requires about $4ml$ multiplications.

3.4.2 The semi-smooth Newton direction

Qi and Sun [24] introduced a semi-smooth Newton method to solve the correlation matrix approximation problem, and further extended it to the problem with general linear equality constraints. Numerical experiments show that the semi-smooth Newton method has better performance than methods using first-order information (for example, the projected gradient method and BFGS method).

For our case, we find that the semi-smooth Newton method can serve as a good acceleration for the solution of the dual problem. Let $\partial g(y^k)$ denote the generalized Jacobian of $g(y)$ at y^k in the sense of Clarke [4]. It is also the convex hull of $\partial_B g(y^k)$ [21]. For (2.4), let $G(y^k) = G + \mathcal{A}^*(y^k)$ and its spectral decomposition be $G(y^k) = P\Lambda(y^k)P^T$, where $\Lambda(y^k) := \mathrm{diag}(\lambda_1(y^k), \ldots, \lambda_n(y^k))$, $\lambda_1(y^k) \geq \cdots \geq \lambda_n(y^k)$ are the eigenvalues and $P \in \mathbb{O}_{G(y^k)} := \{P \in \mathbb{R}^{n\times n} | G(y^k) = P\Lambda(y^k)P^T, P^T P = I_n\}$ is a corresponding orthogonal matrix. Define three index sets associated with $\lambda_i(y^k)$, $i = 1, \ldots, n$ as follows:

[1] The initial inverse Hessian approximation in L-BFGS is generally set to be $\zeta^k I$ with $\zeta^k = \frac{(s^{k-1})^T w^{k-1}}{(w^{k-1})^T w^{k-1}}$.

$$\Gamma_1^k := \Gamma_1(y^k) = \{i|\lambda_i(y^k) > 0\}, \quad \Gamma_2^k := \Gamma_2(y^k) = \{i|\lambda_i(y^k) = 0\}, \quad \Gamma_3^k := \Gamma_3(y^k) = \{i|\lambda_i(y^k) < 0\},$$

and define

$$\mathcal{M} = \left\{ M \in \mathbb{R}^{n\times n} \middle| M = \begin{pmatrix} E_{\Gamma_1^k,\Gamma_1^k} & E_{\Gamma_1^k,\Gamma_2^k} & (\tau_{ij})_{i\in\Gamma_1^k, j\in\Gamma_3^k} \\ E_{\Gamma_2^k,\Gamma_1^k} & (\omega_{ij})_{i,j\in\Gamma_2^k} & 0 \\ (\tau_{ji})_{i\in\Gamma_1^k, j\in\Gamma_3^k} & 0 & 0 \end{pmatrix} \begin{matrix} w_{ij} = w_{ji} \in [0,1], \\ \text{for } i,j \in \Gamma_2^k, \\ \tau_{ij} = \lambda_i^k/(\lambda_i^k - \lambda_j^k), \\ \text{for } i \in \Gamma_1^k, j \in \Gamma_3^k \end{matrix} \right\}.$$

Then we have the following results.

Lemma 3.1 *For any $h \in \mathbb{R}^m$,*

$$\partial_B(g(y^k))h \subseteq \{\mathcal{A}(WT) : \ W \in \mathcal{W}\}, \tag{3.13}$$

where $T = \mathcal{A}^*(h)$, $\mathcal{W}$ is a set of operators given by

$$\mathcal{W} = \{W|WT = P\big(M\circ(P^TTP)\big)P^T, M \in \mathcal{M}, P \in \mathbb{O}_{G(y^k)}, \quad h \in \mathbb{R}^m\}, \tag{3.14}$$

and $\circ$ is the Hadamard product.

Proof The proof follows similarly to that of [24, Lemma 3.5]. □

At y^k, let $J^k \in \partial_B(g(y^k))$ be partitioned as (3.10) assuming, w.l.g., $\mathcal{F}^k = \{1, 2, \ldots, |\mathcal{F}^k|\}$. Our choice of semi-smooth Newton search direction $d^k_{\mathcal{F}^k}$ in d^k for the variables in $\mathcal{F}^k$ satisfies the linear system

$$J^k_{\mathcal{F}^k\mathcal{F}^k} d^k_{\mathcal{F}^k} = -g^k_{\mathcal{F}^k}. \tag{3.15}$$

Thus, for the semi-smooth Newton acceleration, we can compute the part of search direction $-Z\bar{H}^kZ^Tg^k$ in (3.11) as

$$-Z\bar{H}^kZ^Tg^k = \begin{pmatrix} d^k_{\mathcal{F}^k} \\ 0 \end{pmatrix},$$

where $d^k_{\mathcal{F}^k}$ is from (3.15). Practically, by taking advantage of the convexity of the problem, we invoke a certain Krylov subspace method (for example CG) for this system to obtain approximations of $d^k_{\mathcal{F}^k}$, in which the main costs lie in the matrix-vector products of the form $J^k_{\mathcal{F}^k\mathcal{F}^k}h_{\mathcal{F}^k}$ for $h_{\mathcal{F}^k} \in \mathbb{R}^{|\mathcal{F}^k|}$. If N_{CG} CG steps are used to solve for $d^k_{\mathcal{F}^k}$, excluding the multiplications in $\mathcal{A}(X)$ and $\mathcal{A}^*(y)$, we require $(4n^3 + n^2 + 2|\mathcal{F}^k|)N_{CG}$ multiplications.

Note the matrix-vector product $J^k_{\mathcal{F}^k\mathcal{F}^k}h_{\mathcal{F}^k}$ is indeed the first sub-vector of length $|\mathcal{F}^k|$ of

$$J^k\begin{pmatrix} h_{\mathcal{F}^k} \\ 0 \end{pmatrix} = \begin{pmatrix} J^k_{\mathcal{F}^k\mathcal{F}^k}h_{\mathcal{F}^k} \\ J^k_{\mathcal{B}^k\mathcal{F}^k}h_{\mathcal{F}^k} \end{pmatrix},$$

and thus, $J^k_{\mathcal{F}^k\mathcal{F}^k} h_{\mathcal{F}^k} \in \{\mathcal{A}_{\mathcal{F}^k}(WH) : W \in \mathcal{W}\}$, where the linear mapping $\mathcal{A}_{\mathcal{F}^k}$: $\mathcal{S}^n \to \mathbb{R}^{|\mathcal{F}^k|}$ is given by $\mathcal{A}_{\mathcal{F}^k}(X) = [\mathcal{A}(X)]_{\mathcal{F}^k}$ and $H = \mathcal{A}^*\left(\begin{pmatrix} h_{\mathcal{F}^k} \\ 0 \end{pmatrix}\right) = \sum_{i\in\mathcal{F}^k} h_i A_i$. This leads to the following corollary.

Corollary 3.1 *For the index set $\mathcal{F}^k$, define the linear mapping $\mathcal{A}_{\mathcal{F}^k}$: $\mathcal{S}^n \to \mathbb{R}^{|\mathcal{F}^k|}$ by $\mathcal{A}_{\mathcal{F}^k}(X) = [\mathcal{A}(X)]_{\mathcal{F}^k}$, and the function $F(y_{\mathcal{F}^k}) : \mathbb{R}^{|\mathcal{F}^k|} \to \mathbb{R}^{|\mathcal{F}^k|}$ by $F(y_{\mathcal{F}^k}) = g_{\mathcal{F}^k}(y)$ in which the elements of y in $\mathcal{B}^k$ are fixed at $y^k_{\mathcal{B}^k}$, i.e., $y_{\mathcal{B}^k} = y^k_{\mathcal{B}^k}$, where $\mathcal{B}^k = (\mathcal{E}\cup\mathcal{I})\backslash\mathcal{F}^k$. Then for any $h_{\mathcal{F}^k} \in \mathbb{R}^{|\mathcal{F}^k|}$, it follows*

$$\partial_B(F(y^k_{\mathcal{F}^k}))h_{\mathcal{F}^k} \subseteq \{\mathcal{A}_{\mathcal{F}^k}(WH) : W \in \mathcal{W}\},$$

where $H = \mathcal{A}^*(Zh_{\mathcal{F}^k})$, the columns of $Z \in \mathbb{R}^{n\times|\mathcal{F}^k|}$ are $\{e_i | i \in \mathcal{F}^k\}$, and $\mathcal{W}$ is defined in (3.14).

3.5 Algorithm

With the previous preparation, we now can summarize the procedure of our algorithm. Let y^k be the current iteration for (2.3). After computing the search direction d^k in (3.11), we determine a stepsize α^k by the backtracking line search technique to fulfill the following nonmonotone reduction condition on $f(y)$ [36]:

$$f(\Pi_{\mathcal{Q}}(y^k + \alpha^k d^k)) \le f^k_p + \delta\alpha^k (g^k)^T d^k, \tag{3.16}$$

where f^k_p is a weighted average of the past function values that is no less than $f(y^k)$, and $\delta \in (0, \frac{1}{2})$. We mention that such condition (3.16) is more relaxable than the standard Armijo condition. The framework of our proposed algorithm is as follows:

Algorithm 1: Accelerated Active Set Algorithm (AASA)

Given $\varepsilon > 0$, $\epsilon_0 > 0$, $\delta \in (0, \frac{1}{2})$, $\sigma \in (0,1)$, $\xi \in (0,1)$, $\epsilon_a > \epsilon_b > 0$;
Initialization: $y^0 \in \mathbb{R}^m$, $Q^0 = 1$, $f^0_p = f(y^0)$, $d^0 = -g^0$, $k = 0$, `switch= "Q"`;
while $\varrho^k > \varepsilon$ **do**
  Set $\alpha = 1$;
  while $f(\Pi_{\mathcal{Q}}[y^k + \alpha d^k]) > f^k_p + \delta\alpha(g^k)^T d^k$ **do**
    $\alpha = \sigma\alpha$;
  end
  $\alpha^k = \alpha$, $y^{k+1} = \Pi_{\mathcal{Q}}[y^k + \alpha^k d^k]$, $Q^{k+1} = \xi Q^k + 1$, and $f^{k+1}_p = (\xi Q^k f^k_p + f(y^{k+1}))/Q^{k+1}$;
  Compute the next search direction d^{k+1} in (3.11) adaptively by L-BFGS or the semi-smooth Newton iteration according to Algorithm 2 (stated in the next subsection);
  Set $k = k + 1$;
end

Note that the generated sequence $\{f(y^k)\}$ by Algorithm 1 is nonmonotonically descent (see Lemma 4.2 and (4.24)) and the objective function $f(y)$ is coercive, $\{y^k\}$ is bounded, and thereby, it has a convergent subsequence.

The final execution of Algorithm 1 needs the specification of the search direction d^{k+1} in step 9 of Algorithm 1. There are several possible ways for adaptively switching the directions (indexed by variable `switch`) between the one of L-BFGS (denoted by `switch`="`Q`") and the semi-smooth Newton (denoted by `switch`="`N`"). In the next subsection, we shall present an adaptive strategy.

3.6 An adaptive acceleration strategy

This subsection is to specify step 9 of Algorithm 1. In particular, we introduce a two-stage adaptive acceleration strategy: "A-stage acceleration" (triggered by $\varrho^{k+1} \leq \epsilon_a$) and "B-stage acceleration" (triggered by $\varrho^{k+1} \leq \epsilon_b$).

For "A-stage acceleration", we adaptively use the generalized Newton step to accelerate the inactive part of d^{k+1} while the active part of d^{k+1} still goes a BB step. For convenience, we use "`N`" (or N-step) and "`Q`" (or Q-step) to represent the generalized Newton acceleration step (denoted by d_N^{k+1}) and the L-BFGS acceleration step (denoted by d_Q^{k+1}), respectively. Generally, computing d_N^{k+1} is more expensive than d_Q^{k+1}, but d_N^{k+1} can usually bring more improvements on reduction of the residual ϱ^k in (3.4). To measure the improvement at kth iteration, if the previous two steps, i.e., $(k-1)$th and kth are both the generalized Newton steps, we introduce the 3-N-step improvement factor as

$$\chi_N^{k+1} = \frac{\varrho^{k-1} - \varrho^{k+1}}{\varrho^{k+1}},$$

where the generalized Newton steps are used for ϱ^{k-1}, ϱ^k and ϱ^{k+1}. The reason behind our 3-N-step improvement rate is that the generalized Newton step may not provide great reductions on the residual ϱ^k at each iteration, but it usually is able to reduce the residual in several successive steps. Our numerical experiments show that using three successive steps turns out a good choice in general. The "A-stage acceleration" of the adaptive acceleration strategy is stated in Lines 16–27 of Algorithm 2.

To achieve the fast local convergence speed, we introduce "B-stage acceleration" of the adaptive acceleration strategy. Notice from (3.11) that $d_i^k, i \in \mathcal{B}^k$ only involves the gradient information, and the stepsize can be cut to be very small in order to meet the nonmonotone reduction condition (3.16) on f. This implies that the fast local convergence may not be generally ensured in "A-stage acceleration". Fortunately, Theorem 3.2 shows that $\mathcal{B}^k$ and $\mathcal{F}^k$ can correctly identify all active/inactive constraints respectively in the final stage under the constraint nondegeneracy condition [10]. Thus, in "B-stage acceleration", we force $y_i^{k+1}, i \in \mathcal{B}^{k+1}$ to be zero, i.e., restrict the fix variables to the boundary. To distinguish the iterate y^{k+1} in Algorithm 1, we denote the full successive generalized Newton iterates by z^j for $j = 1, 2, \ldots$ with $z^1 = y^{k+1}$ used in this adaptive strategy. This means that the semi-smooth Newton acceleration is a first try at the $(k+1)$th iteration. Whenever the acceleration fails to meet our criterion to be specified, we shrink ϵ_b and go back to y^{k+1}, and then apply the L-BFGS acceleration to Algorithm 1 instead. The "B-stage acceleration" of the adaptive acceleration strategy is described in Lines 2–14 of Algorithm 2.

Algorithm 2: An adaptive acceleration strategy for Algorithm 1

1 Given $\epsilon_a > \epsilon_b > 0$;
2 **if** $\varrho^{k+1} \le \epsilon_b$ **then**
3 Initialize $z_i^1 := \begin{cases} 0, & i \in \mathcal{B}(y^{k+1}) \\ y_i^{k+1}, & i \in \mathcal{F}(y^{k+1}) \end{cases}$ where $\mathcal{B}(y^{k+1})$ and $\mathcal{F}(y^{k+1})$ are defined by (3.6);
4 Set $j = 1$, and `switch`="N";
5 Set $[\bar{d}_N^j]_i := \begin{cases} 0, & i \in \mathcal{B}(z^j) \\ [d_N(z^j)]_i, & i \in \mathcal{F}(z^j) \end{cases}$ where $d_N(z^j)$ is defined by (3.11) associated with z^j;
6 **while** $f(\Pi_{\mathcal{Q}}(z^j + \bar{d}_N^j)) < f(z^j) + \delta \nabla f(z^j)^T d_N(z^j)$ & $\mathcal{F}(z^j) = \mathcal{F}(y^{k+1})$ **do**
7 Set $z^{j+1} = \Pi_{\mathcal{Q}}(z^j + \bar{d}_N^j)$;
8 **if** $\varrho(z^{j+1}) \le \varepsilon$ **then**
9 **return** z^{j+1} as an approximation solution to (2.3);
10 **end**
11 Set $j = j + 1$;
12 Set $[\bar{d}_N^j]_i := \begin{cases} 0, & i \in \mathcal{B}(z^j) \\ [d_N(z^j)]_i, & i \in \mathcal{F}(z^j) \end{cases}$ where $d_N(z^j)$ is defined by (3.11) associated with z^j;
13 **end**
14 Set `switch`="Q" and $d^{k+1} = d_Q^{k+1}$;
15 Set $\epsilon_b = \epsilon_b/5$;
16 **else**
17 **if** $\varrho^{k+1} \le \epsilon_a$ **then**
18 Set `switch`="N";
19 **if** 3-N-step fails to be accepted **then**
20 Set `switch`="Q";
21 **end**
22 **end**
23 **if** `switch`="N" **then**
24 Set $d^{k+1} = d_N^{k+1}$;
25 **else**
26 Set $d^{k+1} = d_Q^{k+1}$;
27 **end**
28 **end**

Remark 3.1 The condition in Line 6 of Algorithm 2 means that during iterations the sufficient decrease condition

$$f(\Pi_{\mathcal{Q}}(z^j + \bar{d}_N^j)) < f(z^j) + \delta \nabla f(z^j)^T d_N(z^j) \tag{3.17}$$

holds and the working set $\mathcal{F}(z^j)$ keeps unchanged.

Remark 3.2 The statement of *'3-N-step fails to be accepted'* in Line 19 of Algorithm 2 means that one of the following conditions holds:

(i) $f(\Pi_{\mathcal{Q}}(y^{k+1} + d_N^{k+1})) > f_p^{k+1} + \delta(g^{k+1})^T d_N^{k+1}$; that is, the generalized Newton step d_N^{k+1} is not sufficiently descent for f in the sense of (3.16) with $\alpha^{k+1} = 1$;
(ii) in case that the previous two steps $(k-1)$th and kth are both the generalized Newton steps, the 3-N-step improvement factor $\chi_N^{k+1} \le \epsilon_\chi$. (In our numerical testing, $\epsilon_\chi = 0.3$)

Remark 3.3 For the computational costs of Algorithm 1 with Algorithm 2 embedded at each iteration, we first note that about $9n^3$ multiplications are required to compute the spectral decomposition of an n-by-n matrix; second, for the search direction, about $4lm$ multiplications are used in L-BFGS for computing d_Q^k, while $(4n^3 + n^2 + 2|\mathcal{F}^k|)N_{CG}$ multiplications for the generalized Newton step d_N^k (or $\bar{d}_N^j$). Therefore, by excluding the multiplications in computing $\mathcal{A}(X)$ and $\mathcal{A}^*(y)$, the total computational amount is about $9n^3 + 4lm$ multiplications for d_Q^k and $(4n^3 + n^2 + 2|\mathcal{F}^k|)N_{CG}$ multiplications for d_N^k (or $\bar{d}_N^j$). The cost in computing $\mathcal{A}(X)$ and $\mathcal{A}^*(y)$ is depend on the structure of the matrices $A_i, i = 1, \ldots, m$, and in particular, in our numerical experiments, $\mathcal{A}(X)$ and $\mathcal{A}^*(y)$ can be calculated efficiently due to the special structure of the matrices $A_i, i = 1, \ldots, m$.

4 Convergence analysis

We now turn to the issue of the global convergence of Algorithm 1. Define four subsets for the index set $\mathcal{I} = \{p + 1, \ldots, m\}$:

$$\mathcal{C}_1^k = \{i \in \mathcal{I} | \tilde{y}_i^k = 0, y_i^k = 0\}, \tag{4.1a}$$

$$\mathcal{C}_2^k = \{i \in \mathcal{I} | \tilde{y}_i^k = 0, y_i^k > 0\}, \tag{4.1b}$$

$$\mathcal{C}_3^k = \{i \in \mathcal{I} | 0 < \tilde{y}_i^k \le \nu^k\}, \tag{4.1c}$$

$$\mathcal{C}_4^k = \{i \in \mathcal{I} | g_i^k \le -\nu^k/(2\alpha_B^k), y_i^k \le \nu^k/2, \tilde{y}_i^k > \nu^k\}. \tag{4.1d}$$

Remark 4.1 The index sets $\mathcal{C}_i^k, i = 1, 2, 3, 4$ are mutually exclusive and $\mathcal{B}^k = \bigcup_{i=1}^4 \mathcal{C}_i^k$. By (4.1), $\mathcal{C}_1^k$ indeed represents the set corresponding to variables on the constraint boundary. The index set $\mathcal{C}_2^k$ contains indices associated with variables that are within the boundary but will get to the boundary after the BB step. The index set $\mathcal{C}_3^k$ points to the indices corresponding to variables that do not reach the boundary after the BB step. The variables y_i^k associated with $\mathcal{C}_4^k$ are very close to the boundary but might not be good "active" candidates due to $g_i^k \le -\nu^k/(2\alpha_B^k) \le 0$; nevertheless, in the limit, $\mathcal{C}_4^k$ might reduce to be empty or identify a part of active set which is said be degenerate (i.e., both g_i^k and y_i^k vanish in the limit).

The index $\mathcal{F}^k$ in (3.6) of the estimated inactive set now can be expressed as

$$\mathcal{F}^k = (\mathcal{D}^k \cap (\mathcal{C}_4^k)^0) \cup \mathcal{E}, \tag{4.2}$$

where

$$\mathcal{D}^k = \{i \in \mathcal{I} | \tilde{y}_i^k > \nu^k\} \tag{4.3}$$

and

$$(\mathcal{C}_4^k)^0 = \{i \in \mathcal{I} | -g_i^k < \nu^k/(2\alpha_B^k) \text{ or } y_i^k > \nu^k/2\}. \tag{4.4}$$

Here, $\mathcal{D}^k$ is actually the complementary set of $\bigcup_{i=1}^3 \mathcal{C}_i^k$ in $\mathcal{I}$. Recalling $\tilde{y}^k = \Pi_{\mathcal{Q}}[y^k - \alpha_B^k g^k]$, we can rewrite d^k in (3.11) equivalently as

$$d_i^k = \begin{cases} 0, & i \in \mathcal{C}_1^k, \\ -y_i^k, & i \in \mathcal{C}_2^k, \\ -\alpha_B^k g_i^k, & i \in \mathcal{C}_3^k \cup \mathcal{C}_4^k, \\ -(Z\bar{H}^k Z^T g^k)_i, & i \in \mathcal{F}^k. \end{cases} \tag{4.5}$$

This expression of d^k facilitates our convergence analysis.

Our global convergence is based on the following assumption.

Assumption 4.1 There exist two scalars $\gamma_1, \gamma_2 > 0$ such that

$$\gamma_1 \|Z^T g^k\| \le (g^k)^T Z \bar{H}^k Z^T g^k, \tag{4.6}$$

$$\|Z\bar{H}^k Z^T\| \le \gamma_2, \tag{4.7}$$

are satisfied for all k.

Remark 4.2 We remark that Assumption 4.1 holds when $\lambda_{\min}(\bar{H}^k) \ge \gamma_1$ and $\|\bar{H}^k\| \le \gamma_2$ for all k, where $\lambda_{\min}(\bar{H}^k)$ denotes the minimum eigenvalue of $\bar{H}^k$. Note that if the strong second-order sufficient condition for the problem (2.3) holds at a minimizer y^*, then $h^T V h > 0$ for all nonzero vectors $h \in \bar{\mathcal{H}}_\theta(y^*)$ and all matrices $V \in \partial_B(\nabla f(y^*))$. Equivalently, for all $V \in \partial_B(\nabla f(y^*))$, all matrices $(\mathcal{A}V\mathcal{A}^*)_{\mathcal{F}\mathcal{F}}$ are positive definite, where $\mathcal{F} = (\mathcal{E} \cup \mathcal{I}) \backslash \mathcal{I}_0(y^*)$. Therefore, if $\bar{H}^k$ is approaching $(\mathcal{A}V\mathcal{A}^*)^{-1}_{\mathcal{F}\mathcal{F}}$, then Assumption 4.1 holds for some scalars $\gamma_1, \gamma_2 > 0$. We will prove in Sect. 5 (refer to Theorem 5.2) that if the constraint nondegenerecy condition [10] for the problem (1.1) holds at X^* (X^* is associated with y^*), then the strong second-order sufficient condition for the problem (2.3) holds at a minimizer y^*. In this sense, Assumption 4.1 is consistent with the constraint nondegenerecy condition for the problem (1.1).

Lemma 4.1 *Suppose that Assumption 4.1 holds. Then*

$$(d^k)^T g^k \le 0. \tag{4.8}$$

Furthermore, equality holds if and only if $d^k = 0$.

Proof According to (4.5),

$$(d^k)^T g^k = -\sum_{i \in \mathcal{C}_2^k} y_i^k g_i^k - \sum_{i \in \mathcal{C}_3^k \cup \mathcal{C}_4^k} \alpha_B^k |g_i^k|^2 - \sum_{i \in \mathcal{F}^k} (Z\bar{H}^k Z^T g^k)_i g_i^k. \tag{4.9}$$

For $i \in \mathcal{C}_2^k$, due to (4.1b) and (3.2), $y_i^k - \alpha_B^k g_i^k \le 0$ and then $\alpha_B^k g_i^k \ge y_i^k > 0$, and therefore $y_i^k g_i^k \ge (y_i^k)^2/\alpha_B^k > 0$ for $i \in \mathcal{C}_2^k$. Thus, the first term in the right hand side of (4.9) is no greater than 0. The second term is obviously non-positive. Also, the third term is also non-positive by (4.6). Consequently, (4.8) is true. From (4.9), we know that $d^k = 0$ if and only if all the terms in the right-hand side of (4.9) vanish. □

Lemma 4.2 *Let* $\{d^k\}$ *and* $\{y^k\}$ *be generated by Algorithm* 1. *If Assumption* 4.1 *holds, then*

$$f(y^k) \le f_p^k \le \chi^k, \tag{4.10}$$

where $f_p^k = (\xi Q^k f_p^{k-1} + f(y^k))/Q^k$, *and* $\chi^k = \frac{1}{k+1}\sum_{i=0}^k f(y^i)$.

Proof The proof follows analogously from [36, Lemma 1.1]. □

Proposition 4.1 *Given* d^k *defined in* (4.5), *we have*

$$y_i^k + d_i^k \ge 0, \quad \text{for all} \quad i \in \bigcup_{i=1}^4 \mathcal{C}_i^k, \tag{4.11}$$

where $y^k \in \mathcal{Q}$.

Proof Since $\bigcup_{i=1}^4 \mathcal{C}_i^k = \mathcal{B}^k$, the conclusion follows from (3.11) and (3.2). □

Proposition 4.2 *The index sets* $\mathcal{C}_i^k, i = 1, 2, 3$ *can be described as follows*

$$\mathcal{C}_1^k = \{i \in \mathcal{I} | \tilde{y}_i^k = 0, y_i^k = 0, g_i^k \ge 0\}, \tag{4.12a}$$

$$\mathcal{C}_2^k = \{i \in \mathcal{I} | \tilde{y}_i^k = 0, y_i^k > 0, g_i^k > 0\}, \tag{4.12b}$$

$$\mathcal{C}_3^k = \{i \in \mathcal{I} | 0 < \tilde{y}_i^k \le \nu^k, \alpha_B^k g_i^k < y_i^k \le \nu^k + \alpha_B^k g_i^k\}, \tag{4.12c}$$

$$\mathcal{D}^k = \{i \in \mathcal{I} | y_i^k > \nu^k + \alpha_B^k g_i^k\}. \tag{4.12d}$$

Proof According to (4.1a), if $i \in \mathcal{C}_1^k$, then $\tilde{y}_i^k = \max(0, y_i^k - \alpha_B^k g_i^k) = \max(0, -\alpha_B^k g_i^k) = 0$ and therefore (4.12a) is true. If $i \in \mathcal{C}_2^k$, then

$$y_i^k - \alpha_B^k g_i^k \le \tilde{y}_i^k = 0 \tag{4.13}$$

because of (3.2) and (4.1b). Thus, $g_i^k \ge y_i^k/\alpha_B^k > 0$ which together with (4.1b) implies (4.12b). By (3.2) and (4.1c), $0 < y_i^k - \alpha_B^k g_i^k = \tilde{y}_i^k \le \nu^k$ for $i \in \mathcal{C}_3^k$ and $\alpha_B^k g_i^k < y_i^k \le \nu^k + \alpha_B^k g_i^k$. Use (4.1c) to have (4.12c). Finally, by (4.3), if $i \in \mathcal{D}^k$, then $\tilde{y}_i^k = y_i^k - \alpha_B^k g_i^k > \nu^k$ which yields (4.12d). □

The reformulation of $\mathcal{C}_i^k, i = 1, 2, 3$ in (4.12) facilitates our subsequent characterization of optimality conditions.

Lemma 4.3 *Let Assumption 4.1 hold, and* $\{d^k\}$ *and* $\{y^k\}$ *be generated by Algorithm 1. Then* y^k *is a KKT point of the problem (2.3) if and only if* $d^k = 0$.

Proof If y^k is a KKT point of the problem (2.3), then $\nu^k = \min\{\epsilon_0, \sqrt{\varrho^k}\} = \sqrt{\varrho^k} = 0$ which implies

$$\min(y_i^k, g_i^k) = 0, i \in \mathcal{I} \text{ and } g_i^k = 0, i \in \mathcal{E}. \tag{4.14}$$

By (4.2), (4.3) and (4.4), if $-g_i^k < \nu^k/(2\alpha_B^k)$ with $i \in \mathcal{F}^k \cap \mathcal{I}$ $(= \mathcal{D}^k \cap (\mathcal{C}_4^k)^0)$, then

$$\nu^k < \tilde{y}_i^k = y_i^k - \alpha_B^k g_i^k < y_i^k + \nu^k/2,$$

and therefore $y_i^k > \nu^k/2 = 0, i \in \mathcal{F}^k \cap \mathcal{I}$. Thus use (4.14) to get $g_i^k = 0, i \in \mathcal{F}^k$ and $d_i^k = 0, i \in \mathcal{F}^k$. If $\mathcal{C}_3^k \neq \emptyset$, from (4.12c) and $\nu^k = 0$, we have $0 < \tilde{y}_i^k \leq \nu^k = 0, i \in \mathcal{C}_3^k$, which is a contradiction. Therefore $\mathcal{C}_3^k = \emptyset$. In view of $\nu_k = 0$, (4.1d) and (4.14), $y_i^k = g_i^k = 0$ and $\tilde{y}_i^k > 0$ for $i \in \mathcal{C}_4^k$. Using (4.5) gives $d_i^k = 0$ and $\tilde{y}_i^k = 0$ for $i \in \mathcal{C}_4^k$, which contradicts $\tilde{y}_i^k > 0$. So $\mathcal{C}_4^k$ is empty. By (4.12b) and (4.14), the index set $\mathcal{C}_2^k$ is empty, too. Hence, $d^k = 0$.

Conversely, assume $d^k = 0$. From (4.5) and (4.12a), we have $y_i^k = 0, g_i^k \geq 0, i \in \mathcal{C}_1^k$. By (4.5) and (4.12b), we know $y_i^k = d_i^k = 0$ indicating that $\mathcal{C}_2^k$ is empty. For $i \in \mathcal{C}_3^k$, due to (4.5) and (4.12c), it is true that $y_i^k > 0, g_i^k = 0$. Also, based on (4.5) and (4.1d), it holds that $\mathcal{C}_4^k = \emptyset$ or $\nu^k = 0$, where the latter implies that y^k is a KKT point of the problem (2.3). For the case $\mathcal{C}_4^k = \emptyset$, from (4.5), we have $(Z\bar{H}^k Z^T g^k)_i = 0, i \in \mathcal{F}^k$ which together with (4.6) gives $g_i^k = 0, i \in \mathcal{F}^k$. Overall, by (3.3), y^k is a KKT point of the problem (2.3). □

Lemma 4.4 *Let* $\{d^k\}$ *and* $\{y^k\}$ *be generated by Algorithm 1. If* $d^k \neq 0$, *then*

$$\beta^k := \sup_{0 \leq \gamma \leq 1} \{\gamma | y^k + \gamma d^k \in \mathcal{Q}\} \geq \min\left\{1, \frac{\nu^k}{2\|d^k\|_\infty}\right\}, \tag{4.15}$$

where $\mathcal{Q}$ *is defined in (2.1).*

Before proving Lemma 4.4, we make some comments on the implication of this lemma. From the current iterate y^k, the trial point $y^k + \alpha d^k$ is searched along d^k. It can be seen that the largest step size that makes this trial point remain in the feasible region $\mathcal{Q}$ must be equal or greater than $\bar{\beta}^k := \min\{1, \frac{\nu^k}{2\|d^k\|_\infty}\}$. Therefore, Lemma 4.4 says that if $\alpha \leq \bar{\beta}^k \leq \beta^k$, by (4.15), we have $y^k + \alpha d^k \in \mathcal{Q}$, and

$$f(y^k + \alpha d^k) = f\big(\Pi_{\mathcal{Q}}[y^k + \alpha d^k]\big). \tag{4.16}$$

Proof It suffices to prove that $y_i^k + \bar{\beta}^k d_i^k \geq 0$ for all $i \in \mathcal{I}$. For $i \in \mathcal{B}^k$ and $0 < \alpha \leq 1$, $y_i^k + \alpha d_i^k = (1 - \alpha) y_i^k + \alpha(y_i^k + d_i^k) \geq 0$ due to $y_i^k \geq 0$ and Proposition 4.1. Hence,

$y_i^k + \bar{\beta}^k d_i^k \geq 0, i \in \mathcal{B}^k$. For $i \in \mathcal{F}^k \cap \mathcal{I}$ $(= \mathcal{D}^k \cap (\mathcal{C}_4^k)^0)$, $y_i^k > \nu^k + \alpha_B^k g_i^k$ due to (4.12d). This combining with (4.4) gives $y_i^k > \nu^k/2, i \in \mathcal{F}^k \cap \mathcal{I}$. Recall $\bar{\beta}^k \leq \frac{\nu^k}{2\|d^k\|_\infty}$. Thus, $y_i^k + \bar{\beta}^k d_i^k \geq \frac{\nu^k}{2} - \frac{\nu^k}{2\|d^k\|_\infty}|d_i^k| \geq 0, i \in \mathcal{F}^k \cap \mathcal{I}$. □

Lemma 4.5 *Suppose that Assumption 4.1 holds. Let* $\{d^k\}$ *and* $\{y^k\}$ *be generated by Algorithm* 1. *Then there exists a scalar* $u_d > 0$ *such that*

$$\|d^k\|_\infty \leq u_d$$

for all k, *i.e., the sequence* $\{d^k\}$ *is bounded.*

Proof From (4.5), we have

$$\|d^k\|^2 = \sum_{i \in \mathcal{C}_2^k} |y_i^k|^2 + \sum_{i \in \mathcal{C}_3^k \cup \mathcal{C}_4^k} (\alpha_B^k)^2 |g_i^k|^2 + \|Z\bar{H}^k Z^T g^k\|^2, \tag{4.17}$$

and together with (4.7) and (4.13), it follows

$$\begin{aligned} \|d^k\|^2 \leq & \sum_{i \in \mathcal{C}_2^k} (\alpha_B^k)^2 |g_i^k|^2 + \sum_{i \in \mathcal{C}_3^k \cup \mathcal{C}_4^k} (\alpha_B^k)^2 |g_i^k|^2 + \sum_{i \in \mathcal{F}^k} \gamma_2^2 |g_i^k|^2 \\ \leq & ((\alpha_B^k)^2 + \gamma_2^2)\|g^k\|^2. \end{aligned} \tag{4.18}$$

Since $g(y)$ is continuous and $\{y^k\}$ is bounded, inequality (4.18) with $\alpha_B^k \leq \alpha_{\max}$ implies $\{\|d^k\|^2\}$ is bounded. Hence, there exists a scalar $u_d > 0$ such that $\|d^k\|_\infty \leq u_d$ for all k. □

The boundedness of $\{d^k\}$ and Lemma 4.4 lead to

$$\beta^k \geq \tilde{\beta}^k := \min\left\{1, \frac{\nu^k}{2u_d}\right\}, \quad \forall k.$$

The next lemma gives a lower bound of the step size α^k, which also implies that Algorithm 1 is well-defined (i.e., the inner loop (from Line 5 to Line 7) terminates finitely).

Lemma 4.6 *Suppose that Assumption 4.1 holds. Let* $\{\alpha^k\}$ *be generated by Algorithm* 1. *Then there exists a scalar* $\bar{\alpha} > 0$ *such that*

$$\alpha^k \geq \min\left\{\sigma\bar{\beta}^k, \bar{\alpha}\right\}, \tag{4.19}$$

where $\bar{\beta}^k$ *is defined after Lemma 4.4, and* $\sigma \in (0, 1)$ *is from Algorithm* 1.

Proof From (4.17), (4.7) and (4.13), we obtain

$$\begin{aligned}\|d^k\|^2 &\le \sum_{i\in\mathcal{C}_2^k} \alpha_B^k y_i^k g_i^k + \sum_{i\in\mathcal{C}_3^k\cup\mathcal{C}_4^k} (\alpha_B^k)^2 |g_i^k|^2 + \gamma_2 (g^k)^T Z \bar{H}^k Z^T g^k \\ &\le \max(\alpha_B^k, \gamma_2)\left(\sum_{i\in\mathcal{C}_2^k} y_i^k g_i^k + \sum_{i\in\mathcal{C}_3^k\cup\mathcal{C}_4^k} \alpha_B^k |g_i^k|^2 + (g^k)^T Z \bar{H}^k Z^T g^k\right) \\ &\le -\max(\alpha_{\max}, \gamma_2)(d^k)^T g^k,\end{aligned} \tag{4.20}$$

where the last inequality follows from (4.9) and $\alpha_B^k \le \alpha_{\max}$. We now prove that the step size α^k generated by Algorithm 1 is no less than either $\sigma\bar{\beta}^k$ or some constant $\bar{\alpha} > 0$. If $\alpha^k < \sigma\bar{\beta}^k$, then the trial step size $\alpha_t^k := \frac{\alpha^k}{\sigma}$ cannot be accepted in Algorithm 1. As $\alpha_t^k < \bar{\beta}^k$,

$$f(\Pi_{\mathcal{Q}}[y^k + \alpha_t^k d^k]) = f(y^k + \alpha_t^k d^k) > f_p^k + \delta\alpha_t^k (g^k)^T d^k, \tag{4.21}$$

where the first equality follows from (4.16). By the mean value theorem,

$$\begin{aligned}f(y^k + \alpha_t^k d^k) &= f(y^k) + \alpha_t^k (g^k)^T d^k + (d^k)^T \int_0^{\alpha_t^k} [g(y^k + \check{\alpha} d^k) - g(y^k)] \mathrm{d}\check{\alpha} \\ &\le f(y^k) + \alpha_t^k (g^k)^T d^k + \frac{1}{2} L_g (\alpha_t^k)^2 \|d^k\|^2,\end{aligned} \tag{4.22}$$

where L_g is the Lipschitz constant of $g(y)$. By (4.10), (4.21) and (4.22),

$$\alpha_t^k (g^k)^T d^k + \frac{1}{2} L_g (\alpha_t^k)^2 \|d^k\|^2 \ge \delta\alpha_t^k (g^k)^T d^k,$$

and together with (4.20), we get

$$\alpha_t^k \ge -\frac{2(1-\delta)(g^k)^T d^k}{L_g \|d^k\|^2} \ge \frac{2(1-\delta)}{L_g} \frac{1}{\max(\alpha_{\max}, \gamma_2)}.$$

As $\alpha^k = \sigma\alpha_t^k$, it is true that $\alpha^k \ge \bar{\alpha}$ in the case $\alpha^k < \sigma\bar{\beta}^k$, where $\bar{\alpha} = \frac{2\sigma(1-\delta)}{L_g \max(\alpha_{\max}, \gamma_2)}$. Therefore,

$$\alpha^k \ge \min\left\{\sigma\bar{\beta}^k, \bar{\alpha}\right\}$$

holds in all cases. □

Theorem 4.1 *Suppose that Assumption 4.1 holds. Let $\{y^k\}$ be generated by Algorithm 1. Then any accumulation point of the sequence $\{y^k\}$ is a KKT point of the problem (2.3).*

Proof The update rule of Q^{k+1} given by Algorithm 1 yields

$$Q^{k+1} = 1 + \xi Q^k = 1 + \sum_{j=1}^{k+1} \xi^j \le \frac{1}{1-\xi}. \tag{4.23}$$

By the nonmonotone reduction condition of f and (4.10), we have

$$\begin{aligned} f_p^{k+1} =& \frac{\xi Q^k f_p^k + f(y^{k+1})}{Q^{k+1}} \\ \leq& \frac{(\xi Q^k + 1) f_p^k + \alpha^k \delta (d^k)^T g^k}{Q^{k+1}} \\ =& f_p^k + \frac{\alpha^k \delta (d^k)^T g^k}{Q^{k+1}} \\ \leq& f_p^k + \alpha^k (1-\xi) \delta (d^k)^T g^k, \end{aligned} \tag{4.24}$$

where the last inequality follows from (4.8) and (4.23). Since $\{y^k\}$ is bounded, $\{f(y^k)\}$ and $\{f_p^k\}$ are bounded below by (4.10). Combine with (4.24) to have

$$-\sum_{k=1}^{\infty} \alpha^k (1-\xi) \delta (d^k)^T g^k \leq \sum_{k=1}^{\infty} \left(f_p^k - f_p^{k+1} \right) < \infty.$$

Using (4.8), we obtain that

$$\lim_{k \to \infty} \alpha^k (d^k)^T g^k = 0. \tag{4.25}$$

We now prove that any accumulation point of $\{y^k\}$ is a KKT point of the problem (2.3). Let y^* be an accumulation point of $\{y^k\}$, and the subsequence $\{y^{k_i}\}$ of $\{y^k\}$ converges to y^*. It follows from the continuity of $\nabla f(y)$ that $g^{k_i} \to g^* := \nabla f(y^*)$ as $k_i \to \infty$. Similarly, due to (3.4) of the KKT error ϱ^{k_i}, we can assume, without loss of generality, that $\{\varrho^{k_i}\}$ converges to some limit point ϱ^*. If $\varrho^* = 0$, then y^* is already a KKT point. Otherwise, for all sufficiently large k_i, $\varrho^{k_i} \geq \frac{\varrho^*}{2}$. By (4.19) and Lemmas 4.5 and 4.6, α^{k_i} is bounded away from zero, and according to (4.25),

$$\lim_{k_i \to \infty} (d^{k_i})^T g^{k_i} = 0. \tag{4.26}$$

As k increases, we can assume without loss of generality that $\mathcal{C}_i^k, i = 1, 2, 3, 4$ and $\mathcal{F}^k$ are constants for all sufficiently large $k = k_i$, and thereby, we drop the superscripts k by simply denoting $\mathcal{C}_i, i = 1, 2, 3, 4$ and $\mathcal{F}$. Using (4.9) and (4.6), we have that

$$\begin{aligned} y_i^* g_i^* &= 0, \ i \in \mathcal{C}_2, \\ g_i^* &= 0, \quad i \in \mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{F}. \end{aligned}$$

Recall the definitions of d^k and $\mathcal{C}_i^k, i = 1, 2$ to have

$$\begin{aligned} y_i^* = 0, g_i^* \geq 0, \ i \in \mathcal{C}_1, \\ g_i^* \geq 0, y_i^* \geq 0 \ \ i \in \mathcal{C}_2. \end{aligned}$$

Putting above equalities and inequalities together, we conclude that y^* is a KKT point of the problem (2.3). □

5 Local quadratic convergence

We now investigate the convergence behavior of Algorithm 1 with Algorithm 2 embedded in Line 9 for adaptively computing the search direction. For simplicity, we call such an implementation of Algorithm 1 as AASA(Adaptive), for which the sequences $\{y^k\}$ and $\{z^j\}$ are called the outer-loop and inner-loop sequence, respectively. It should be noticed from Algorithm 2 that a particular inner-loop sequence $\{z^j\}$ starts from a certain outer-loop iterate y^{k+1}, and has one of the two mutually exclusive scenarios: (a) $\{z^j\}$ stops at some z^{j_k} and then the iteration of AASA(Adaptive) enters back to y^{k+1}, and continuously produces y^{k+2}, and (b) $\{z^j\}$ is an infinite sequence satisfying $\varrho(z^j) \to 0$ as $j \to \infty$. For the former case (a), the intermediate $\{z^j\}_{j=1}^{j_k}$ are only trials which do not change current outer y^{k+1}, whereas for (b), AASA(Adaptive) converges. Therefore, one and only one of the following situations occurs in AASA(Adaptive):

(i) an infinite outer-loop sequence $\{y^k\}_{k=1}^{\infty}$ is generated, or
(ii) from some y^{k+1}, a sequence $\{z^j\}_{j=1}^{\infty}$ is generated by Algorithm 2 and AASA(Adaptive) converges.

Theorem 5.1 *Suppose that Assumption 4.1 holds, then any accumulation point of the sequence $\{y^k\}$ from case (i) or $\{z^j\}$ from case (ii) is a KKT point of the problem (2.3).*

Proof If case (i) occurs, we know that all the inner-loop sequences $\{z^j\}$ are finite sequences, and the convergence analysis of Theorem 4.1 is true for the outer-loop sequence $\{y^k\}_{k=1}^{\infty}$. Indeed, for a y^{k+1}, if AASA(Adaptive) generates a finite sequence $\{z^j\}_{j=1}^{j_k}$, the search direction at y^{k+1} will be reset as the L-BFGS direction d_Q^{k+1} (cf. Line 14 of Algorithm 2), and thus Theorem 4.1 applies for this case.

For the case of (ii), the condition in Line 6 of Algorithm 2 ensures that

$$f(z^{j+1}) < f(z^j) + \delta \nabla f(z^j)^T d_N(z^j), \quad \forall j. \tag{5.1}$$

By Assumption 4.1 and Lemma 4.1,

$$\nabla f(z^j)^T d_N(z^j) \le 0,$$

and use (5.1) to conclude $\{f(z^j)\}$ is monotonically decreasing. The coercivity of f ensures the boundedness of $\{f(z^j)\}$ and further the convergence of $\{f(z^j)\}$. Thus by (5.1) and Lemma 4.1, $\lim_{j\to\infty} \nabla f(z^j)^T d_N(z^j) = 0$, a result similar to (4.26). With this and following analogously the proof of Theorem 4.1, we know any accumulation point of the sequence $\{z^j\}$ is a KKT point of (2.3). □

Let y^* be an accumulation point of the sequence $\{y^k\}$ (or $\{z^j\}$) generated by AASA(Adaptive). According to the convexity of (2.3), y^* is also a minimizer. From [22, Theorem 2.2], we know that a minimizer y^* is isolated under the second-order

sufficient condition, which, by Theorem 5.2 below, is true under the constraint nondegeneracy condition for the primal problem (1.1). Therefore, we can assume that the sequence $\{y^k\}$ (or $\{z^j\}$) converges to a minimizer y^* with the constraint nondegeneracy condition, and define accordingly

$$X^* = \Pi_{\mathcal{S}^n_+}[G + \mathcal{A}^*(y^*)]. \tag{5.2}$$

The constraint nondegenerecy condition holds at X^* if

$$\begin{pmatrix} \mathcal{A} \\ \mathcal{I}_e \end{pmatrix} \mathcal{S}^n + \begin{pmatrix} \mathrm{lin} T_{\mathcal{Q}}(\mathcal{A}(X^*) - b) \\ \mathrm{lin} T_{\mathcal{S}^n_+}(X^*) \end{pmatrix} = \begin{pmatrix} \mathbb{R}^m \\ \mathcal{S}^n \end{pmatrix},$$

where $T_{\mathcal{K}}(X)$ denotes the tangent cone of $\mathcal{K}$ at $X \in \mathcal{K}$, $\mathrm{lin} T_{\mathcal{K}}(X)$ denotes the largest linear space contained in $T_{\mathcal{K}}(X)$, and $\mathcal{I}_e$ is the identity mapping from $\mathcal{S}^n$ to $\mathcal{S}^n$. Let the primal index set of active constraints at X^* be

$$\mathcal{I}^p(X^*) = \{i | \langle A_i, X^* \rangle = b_i, \quad i = p+1, \ldots, m\}. \tag{5.3}$$

Then the linear mapping $\mathcal{A}_{\bar{\mathcal{F}}}$: $\mathcal{S}^n \to \mathbb{R}^{p+\bar{s}}$ given by $\mathcal{A}_{\bar{\mathcal{F}}}(X) = [\mathcal{A}(X)]_{\bar{\mathcal{F}}}$ has its adjoint $\mathcal{A}^*_{\bar{\mathcal{F}}}$, where $\bar{\mathcal{F}} = \mathcal{E} \cup \mathcal{I}^p(X^*)$, and $\bar{s}$ denotes the cardinality of $\mathcal{I}^p(X^*)$.

Theorem 5.2 *If the constraint nondegeneracy condition for the problem* (1.1) *holds at X^* given in* (5.2)*, then the strong second-order sufficient condition for the dual problem* (2.3) *holds at y^*.*

Proof By [10, Lemma 4.3], for any $V \in \partial_B(\Pi_{\mathcal{S}^n_+}[X^*])$,

$$\langle h_{\bar{\mathcal{F}}}, \mathcal{A}_{\bar{\mathcal{F}}} V \mathcal{A}^*_{\bar{\mathcal{F}}}(h_{\bar{\mathcal{F}}}) \rangle > 0$$

for any $h_{\bar{\mathcal{F}}} \neq 0$ with $h \in \mathbb{R}^m$. Because of the complementarity between $\langle A_i, X^* \rangle = b_i, i \in \mathcal{I}$ and y^*_i, $\bar{\mathcal{F}} \cup \mathcal{I}_0(y^*)$ equals $\mathcal{E} \cup \mathcal{I}$, but $\bar{\mathcal{F}} \cap \mathcal{I}_0(y^*)$ may not be empty. So, if $i \notin \bar{\mathcal{F}}$, then $i \in \mathcal{I}_0(y^*)$. For any $h \in \bar{\mathcal{H}}_\theta(y^*) \backslash \{0\}$, $h_i = 0, i \in \mathcal{I}_0(y^*)$, and then $\mathcal{A}^*(h) = \sum_{i \in \bar{\mathcal{F}}} h_i A_i + \sum_{i \notin \bar{\mathcal{F}}} h_i A_i = \mathcal{A}^*_{\bar{\mathcal{F}}}(h_{\bar{\mathcal{F}}})$. Therefore, for any $h \in \bar{\mathcal{H}}_\theta(y^*) \backslash \{0\}$ and any $V \in \partial_B(\Pi_{\mathcal{S}^n_+}[X^*])$,

$$\begin{aligned} \langle h, \mathcal{A} V \mathcal{A}^*(h) \rangle &= \langle \mathcal{A}^*(h), V \mathcal{A}^*(h) \rangle \\ &= \langle \mathcal{A}^*_{\bar{\mathcal{F}}}(h_{\bar{\mathcal{F}}}), V \mathcal{A}^*_{\bar{\mathcal{F}}}(h_{\bar{\mathcal{F}}}) \\ &= \langle h_{\bar{\mathcal{F}}}, \mathcal{A}_{\bar{\mathcal{F}}} V \mathcal{A}^*_{\bar{\mathcal{F}}}(h_{\bar{\mathcal{F}}}) \rangle > 0, \end{aligned}$$

which implies the strong second-order sufficient condition. □

Under the constraint nondegeneracy condition for the primal problem (1.1), the following theorem shows that our algorithm converges to a minimizer of the dual problem (2.3) quadratically.

Theorem 5.3 *Let y^* be any limit point from case (i) or (ii) of AASA(Adaptive). If the strict complementarity condition for the dual problem* (2.3) *holds at y^* and the*

constraint nondegeneracy condition holds at X^* given by (5.2), then y^* must be a limit point from case (ii); that is, y^* is a limit point of $\{z^j\}_{j=1}^{\infty}$ generated at Line 7 of Algorithm 2 from some outer iterate y^{k+1}. Moreover, $\{z^j\}_{j=1}^{\infty}$ converges to y^* quadratically.

Proof By assumption, Theorem 5.2 ensures that the strong second-order sufficient condition holds at the KKT point y^*, which is a minimizer by the convexity. Using Theorem 3.2, for sufficiently large k, $\mathcal{F}(y^k)$ (or $\mathcal{F}(z^j)$) and $\mathcal{B}(y^k)$ (or $\mathcal{B}(z^j)$) can correctly identify the inactive and active sets $\mathcal{F}$ and $\mathcal{B}$, respectively.

Let X^* and $\mathcal{I}^p(X^*)$ be defined by (5.2) and (5.3), respectively. Then the strict complementarity condition ensures $\mathcal{F} = \bar{\mathcal{F}} := \mathcal{E} \cup \mathcal{I}^p(X^*)$. Therefore, with the active (inactive) set identified by y^k which is sufficiently close to y^*, solving (2.3) is equivalent to $\min_{z\in\mathcal{Z}} f(z)$, where $\mathcal{Z} = \{z | z_i = 0, i \in \mathcal{B}\}$. At such y^k, define an auxiliary function $\hat{f}(z_{\mathcal{F}})$ with the variables $z_{\mathcal{F}}$ extracted from z by $\mathcal{F}$ and $z_{\mathcal{B}} = 0$ fixed, and $\hat{f}(z_{\mathcal{F}}) = f_{|_{z\in\mathcal{Z}}}(z)$. Thus

$$\min_{z\in\mathcal{Z}} f(z) \Longleftrightarrow \min_{z_{\mathcal{F}}\in\mathbb{R}^{|\mathcal{F}|}} \hat{f}(z_{\mathcal{F}}). \tag{5.4}$$

Due to the convexity of $\hat{f}(z_{\mathcal{F}})$, (5.4) can be solved from

$$F(z_{\mathcal{F}}) := \nabla\hat{f}(z_{\mathcal{F}}) = [\nabla f_{|_{z\in\mathcal{Z}}}(z)]_{\mathcal{F}} = 0,$$

where $\nabla f(z)$ is given by (2.4).

Let $\{\hat{z}_{\mathcal{F}}^j\}_{j=1}^{\infty}$ be the sequence from the generalized Newton iteration for $F(z_{\mathcal{F}}) = 0$ via

$$\hat{z}_{\mathcal{F}}^{j+1} = \hat{z}_{\mathcal{F}}^j + \hat{d}_{\mathcal{F}}^j,\ j = 1, 2, \ldots, \tag{5.5}$$

where $\hat{J}^j\hat{d}_{\mathcal{F}}^j = -F(\hat{z}_{\mathcal{F}}^j)$ and $\hat{J}^j \in \partial_B F(\hat{z}_{\mathcal{F}}^j)$. For the index set $\mathcal{F}$, define the linear mapping $\mathcal{A}_{\mathcal{F}}\colon \mathcal{S}^n \to \mathbb{R}^{|\mathcal{F}|}$ by $\mathcal{A}_{\mathcal{F}}(X) = [\mathcal{A}(X)]_{\mathcal{F}}$, whose adjoin mapping $\mathcal{A}_{\mathcal{F}}^*\colon \mathbb{R}^{|\mathcal{F}|} \to \mathcal{S}^n$ is given by $\mathcal{A}_{\mathcal{F}}^*(h_{\mathcal{F}}) = \sum_{i\in\mathcal{F}} h_i A_i$. By (2.4),

$$F(z_{\mathcal{F}}) = [\nabla f(z)]_{\mathcal{F}} = \mathcal{A}_{\mathcal{F}}[G + \mathcal{A}_{\mathcal{F}}^*(z_{\mathcal{F}})]_+ - b$$

is locally Lipschitz continuous and also strongly semi-smooth on $\mathbb{R}^{|\mathcal{F}|}$ [3, 32]. From [10, Lemma 4.3], for all $V \in \partial_B(\Pi_{\mathcal{S}_+^n}[G + \mathcal{A}_{\mathcal{F}}^*(y_{\mathcal{F}}^*)])$, every $\mathcal{A}_{\mathcal{F}} V \mathcal{A}_{\mathcal{F}}^* \in \partial_B F(y_{\mathcal{F}}^*)$ is positive definite. Consequently, by [23, Theorems 3.2 and 3.3], $\{\hat{z}_{\mathcal{F}}^j\}$ converges to $y_{\mathcal{F}}^*$ globally and quadratically provided that $\hat{z}_{\mathcal{F}}^1$ is sufficiently close to $y_{\mathcal{F}}^*$.

Recall y^* is a limit point from case (i) or (ii). By Theorem 5.1, we can assume, without loss of generality, that z^1 (see Line 3 of Algorithm 2) is sufficiently close to the limit point y^* of either case (i) or (ii). Let $z_{\mathcal{F}}^1$ be the subvector of z^1 indexed by $\mathcal{F}$. Starting from $\hat{z}_{\mathcal{F}}^1 = z_{\mathcal{F}}^1$, the sequence $\{\hat{z}_{\mathcal{F}}^j\}$ generated by (5.5) converges to $y_{\mathcal{F}}^*$ globally and quadratically. Augment $\hat{z}_{\mathcal{F}}^j$ to $\hat{z}^j$ by setting $\hat{z}_{\mathcal{B}}^j = 0$ accordingly to have an auxiliary sequence $\{\hat{z}^j\}$, and we know it converges to y^* globally and quadratically. Our final task is to show $\{\hat{z}^j\}$ is the same as that generated by Algorithm 2 (cf. Line 7).

The proof is by induction on j. First, $z^1 = \hat{z}^1$. Suppose the conclusion holds for $j > 0$ and we will show $z^{j+1} = \hat{z}^{j+1}$. In fact, as $\{\hat{z}^j\}$ converges to y^* and $\hat{z}^1 (= z^1)$ is sufficiently close to y^*, by the inductive hypothesis, we know $\mathcal{F}(z^j) = \mathcal{F}$ and $\mathcal{B}(z^j) = \mathcal{B}$. This ensures that the later condition in Line 6 of Algorithm 2 is fulfilled. By (3.11) and (3.15), $[\bar{d}_N^j]_{\mathcal{F}} (= [d_N(z^j)]_{\mathcal{F}})$ is exactly $\hat{d}_{\mathcal{F}}^j$ in (5.5). Because of $\hat{z}^j \to y^*$ and $[\bar{d}_N^j]_{\mathcal{B}} = 0$ (see Lines 5 and 12 of Algorithm 2), $d_N(\hat{z}^j) \to 0$. Second, due to $y^*_{\mathcal{F}\cap\mathcal{I}} > 0$, assume, without loss of generality, $\hat{z}^j_{\mathcal{F}\cap\mathcal{I}} > 0$. By induction, we further have $\Pi_{\mathcal{Q}}(z^j + \bar{d}_N^j) = \Pi_{\mathcal{Q}}(\hat{z}^j + \bar{d}_N^j) = \hat{z}^j + \bar{d}_N^j = z^j + \bar{d}_N^j$. Note that the strict complementarity condition implies $y^*_{\mathcal{B}} = 0$ and $g^*_{\mathcal{B}} > 0$. Assume $g_{\mathcal{B}}(\hat{z}^j) > 0$, and thus $g_{\mathcal{B}}(z^j) > 0$. By (3.11) and (3.2),

$$[d_N(z^j)]_{\mathcal{B}} = \max(0, z^j_{\mathcal{B}} - \alpha^j_B g_{\mathcal{B}}(z^j)) - z^j_{\mathcal{B}}, \quad \alpha^j_B > 0.$$

Hence, $z^j_{\mathcal{B}} = 0$ and $g_{\mathcal{B}}(z^j) > 0$ imply $[d_N(z^j)]_{\mathcal{B}} = 0$. Due to the generalized Newton equation $J^j_{\mathcal{F}\mathcal{F}}[\bar{d}_N^j]_{\mathcal{F}} = -g_{\mathcal{F}}(z^j)$ with $J^j_{\mathcal{F}\mathcal{F}} \in \partial_B F(z^j_{\mathcal{F}})$, it holds that

$$g_{\mathcal{F}}(z^j)^T[\bar{d}_N^j]_{\mathcal{F}} = -[\bar{d}_N^j]_{\mathcal{F}}^T J^j_{\mathcal{F}\mathcal{F}}[\bar{d}_N^j]_{\mathcal{F}}$$

and $\|z^j_{\mathcal{F}} + [\bar{d}_N^j]_{\mathcal{F}} - y^*_{\mathcal{F}}\| = \mathcal{O}(\|z^j_{\mathcal{F}} - y^*_{\mathcal{F}}\|^2)$. By [10, Lemma 4.3], $J^j_{\mathcal{F}\mathcal{F}}$ is uniformly positive definite for all $z^j_{\mathcal{F}}$, i.e., $\exists \bar{\rho} > 0$ such that

$$g_{\mathcal{F}}(z^j)^T[\bar{d}_N^j]_{\mathcal{F}} \le -\bar{\rho}\|[\bar{d}_N^j]_{\mathcal{F}}\|^2, \quad \forall z^j_{\mathcal{F}}.$$

Applying [6, Lemma 3.2 and Theorem 3.3], we have

$$\hat{f}(z^j_{\mathcal{F}} + [\bar{d}_N^j]_{\mathcal{F}}) < \hat{f}(z^j_{\mathcal{F}}) + \delta g_{\mathcal{F}}(z^j)^T[d_N(z^j)]_{\mathcal{F}}. \tag{5.6}$$

As $z^j_{\mathcal{B}} = 0$ and $[d_N(z^j)]_{\mathcal{B}} = 0$, it holds that

$$f(z^j + \bar{d}_N^j) = \hat{f}(z^j_{\mathcal{F}} + [\bar{d}_N^j]_{\mathcal{F}}), \quad f(z^j) = \hat{f}(z^j_{\mathcal{F}}), \text{ and } g(z^j)^T d_N(z^j) = g_{\mathcal{F}}(z^j)^T[d_N(z^j)]_{\mathcal{F}},$$

which together with (5.6) yields that

$$f(z^j + \bar{d}_N^j) < f(z^j) + \delta g(z^j)^T d_N(z^j).$$

Consequently, the sufficient decrease condition (3.17) is true due to $\Pi_{\mathcal{Q}}(z^j + \bar{d}_N^j) = z^j + \bar{d}_N^j$. According to Algorithm 2 (see Lines 6-7), the new iterate is

$$z^{j+1} = \Pi_{\mathcal{Q}}(z^j + \bar{d}_N^j) = z^j + \bar{d}_N^j = \hat{z}^{j+1},$$

where the last equality follows from $[d_N(z^j)]_{\mathcal{F}} = \hat{d}^j_{\mathcal{F}}$, $[d_N(z^j)]_{\mathcal{B}} = 0$, and the inductive hypothesis. This completes the proof. □

6 Numerical experiments

In this section, we conduct numerical evaluation of Algorithm 1 with the adaptive acceleration of Algorithm 2 (denoted by AASA(Adaptive)) on various problems. Our numerical experiments are obtained by comparing with several other approaches or implementations, including the inexact smoothing Newton method (denoted by ISNM) [10], the projected BFGS method (denoted by P-BFGS) [14], Algorithm 1 accelerated by the pure limited memory BFGS (denoted by AASA(L-BFGS)), and the alternating direction method (ADM) [12, 35]. The codes of ISNM and P-BFGS are available online,[2] while the ADM is coded by ourselves. For comparison purpose, we report the number of iterations, CPU time and the KKT residuals ϱ^k. The numerical comparison was carried out on a PC under Windows 8 (64bits) system with Inter(R) Core(TM) i5-4590 CPU @ 2.4GHz and 4GB memory, in the Matlab environment (R2019a).

For the parameters involved, we terminate our algorithm whenever $\varrho^k \le \varepsilon = 10^{-6}$; other parameters in our implementation are given as follows:

$$\sigma = 0.2, \quad \delta = 0.02, \quad \xi = 0.85, \quad \epsilon_a = 0.2, \quad \epsilon_b = 0.005, \quad \epsilon_\chi = 0.3, \quad y^0 = 0.$$

All the parameters for AASA(L-BFGS) are the same as AASA(Adaptive). For ISNM, P-BFGS and ADM, we set $\varepsilon = 10^{-6}$ as tolerance and use default values for other parameters. We remark that the stopping conditions for AASA(Adaptive), ISNM, P-BFGS and ADM are not completely the same, due to their different optimality measures involved.

Our numerical examples are in the following form:

$$\begin{cases} \min & \frac{1}{2}\|X - G\|_F^2 \\ \text{s.t.} & X_{ij} = b_{ij}, \quad (i,j) \in \mathcal{B}_e \\ & X_{ij} \ge l_{ij}, \quad (i,j) \in \mathcal{B}_l, \\ & X_{ij} \le u_{ij}, \quad (i,j) \in \mathcal{B}_u, \\ & X \in \mathcal{S}_+^n, \end{cases} \tag{6.1}$$

where $\mathcal{B}_e$, $\mathcal{B}_l$ and $\mathcal{B}_u$ are three index subsets of $\{(i,j)|1 \le i \le j \le n\}$; in particular, the values of l_{ij} for $(i,j) \in \mathcal{B}_l$ and u_{ij} for $(i,j) \in \mathcal{B}_u$ are lower and upper bounds, respectively, satisfying $l_{ij} < u_{ij}$ for all $(i,j) \in \mathcal{B}_l \cap \mathcal{B}_u$.

We choose various specific problems for testing. The set of test problems includes synthetic data as well as real world data. Numerical results from synthetic problems are reported in Sect. 6.1, where the data matrix G is generated randomly with medium size problems ($n < 1000$) and large-scale problems ($1000 \le n \le 2000$). Numerical results from real world data are shown in Sect. 6.2, where two data matrices are from financial markets (Shenzhen Stock Exchange and Shanghai Stock

[2] Codes of the approaches of ISNM and P-BFGS are available online at http://www.math.nus.edu.sg/~matsundf/, and https://ctk.math.ncsu.edu/matlab_darts.html, respectively.

Exchange in China) and constraint positions and constraint levels are specified to simulate some stress testing scenarios in financial risk management.

6.1 Synthetic examples

Similar to [10, 12], we randomly generate six synthetic examples (denoted by **E1**–**E6**) as follows:

E1: The matrix G is generated by Matlab built-in command `rand` via `G=2.0*rand(n,n)-ones(n,n);` `G=triu(G)*triu(G,1)';` `for i=1:n; G(i,i)=1; end;` The index sets are

$$\mathcal{B}_e = \{(i,i)|i = 1,\dots,n\} \quad \text{and} \quad \mathcal{B}_l = \mathcal{B}_u = \{(i, \min(i+j,n))|i = 1,\dots,n, \quad j = 1,\dots,n_r\},$$

where $n_r \le n$ is an positive integer. Moreover, $b_{ii} = 1$ for $(i,i) \in \mathcal{B}_e$, $l_{ij} = -0.1$ for $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.1$ for $(i,j) \in \mathcal{B}_u$.

E2: G and $\mathcal{B}_e$ are the same as in **E1**. The index sets $\mathcal{B}_l, \mathcal{B}_u \subset \{(i,j)|1 \le i < j \le n\}$ consist of $\min(n_r, n-i)$ pairs (i, j) with j randomly generated at the ith row of X, $i = 1,\dots,n$. Similar to **E1**, $b_{ii} = 1$ for $(i,i) \in \mathcal{B}_e$, $l_{ij} = -0.1$ for $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.1$ for $(i,j) \in \mathcal{B}_u$.

E3: The settings are the same as in **E1** except that $l_{ij} = -0.5$ for $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.5$ for $(i,j) \in \mathcal{B}_u$.

E4: The settings are the same as in **E2** except that $l_{ij} = -0.5$ for $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.5$ for $(i,j) \in \mathcal{B}_u$.

E5: The settings are the same as in **E1** except that $l_{ij} = -0.5 * \texttt{rand}$ for $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.5 * \texttt{rand}$ for $(i,j) \in \mathcal{B}_u$.

E6: The settings are the same as in **E2** except that $l_{ij} = -0.5 * \texttt{rand}$ for $(i,j) \in \mathcal{B}_l$, and $u_{ij} = 0.5 * \texttt{rand}$ for $(i,j) \in \mathcal{B}_u$.

6.1.1 Choice of the parameter *l*

Generally, the performance of the L-BFGS method is dependent on the parameter l in (3.12). Also, since AASA(Adaptive) and AASA(L-BFGS) both involve the L-BFGS update, a good choice l is desired practically. For that purpose, we particularly test AASA(L-BFGS) using various $l = 2, 3, 5, 8$ on **E3** with $n = 1000, 1500, 2000$ and $n_r = 300, 600, 800, 1200$. The numerical results are listed in Table 1, where the label 'Iter' stands for the number of iterations, and $m = 2(n-1-n_r)n_r + (1+n_r)n_r + n$ is the number of constraints.

From Table 1, overall, we observed that $l = 3$ turns out to be a good choice for AASA(L-BFGS). In particular, the algorithm with $l = 3$ converges (in CPU time) fastest in general. Since AASA(Adaptive) generates the same iteration in the earlier stage as AASA(L-BFGS), we also set $l = 3$ for both AASA(Adaptive) and AASA(L-BFGS) in the following numerical experiments.

Table 1 Numerical results for AASA(L-BFGS) with different l on E3,

n	n_r	m	Iter				CPU time (s)			
			$l=2$	$l=3$	$l=5$	$l=8$	$l=2$	$l=3$	$l=5$	$l=8$
1000	300	510,700	930	**793**	880	1039	258	**233**	280	371
1000	600	840,400	1636	1197	**1174**	1453	571	**447**	489	694
1000	800	960,200	1917	1447	**1278**	1651	726	590	**586**	859
1500	300	811,200	941	852	**832**	907	692	**646**	664	779
1500	600	1,440,900	1718	**1405**	1534	1645	1497	**1284**	1509	1760
1500	800	1,760,700	1927	**1420**	1638	1870	1806	**1406**	1811	2223
1500	1200	2,160,300	2670	1880	**1772**	2646	2751	**2118**	**2178**	3704
2000	300	1,111,700	**787**	**787**	952	1115	**1268**	1299	1618	1997
2000	600	2,041,400	1772	1427	**1369**	1891	3195	**2700**	2723	4159
2000	800	2,561,200	2097	1697	**1555**	2358	4072	3404	**3335**	5582
2000	1200	3,360,800	3231	2144	**2138**	3186	6771	**4798**	5192	8477

The best cases in terms of number of iterations and cpu time are bold

6.1.2 Fast local convergence

We now demonstrate the fast local convergence of AASA(Adaptive) in practice. For illustration, we run AASA (Adaptive) on two instances of **E2**: one is a medium instance with $n = 500$, $n_r = 200$ and $m = 90{,}400$, and the other is a large-scale instance with $n = 2000$, $n_r = 500$ and $m = 1{,}751{,}500$. The tolerance[3] ϵ is set to be 10^{-8} in order to observe the quadratic convergence.

In this experiment, AASA(Adaptive) solves the first instance within 160 iterations, where the generalized Newton steps d_N^k and $\bar{d}_N^j$ are used 15 times ($\bar{d}_N^j$ is rejected for once time, accepted 3 times, and d_N^k is used for the remaining 11 times). For the second instance, it takes 311 iterations, where the generalized Newton steps d_N^k and $\bar{d}_N^j$ are used 17 times ($\bar{d}_N^j$ is rejected for 2 times, accepted 2 times, and d_N^k is used for the remaining 13 times). The detailed information of iterations on the two instances is given in Tables 2 and 3, where the label 'Res' represents the KKT residual. The column of 'Step d^k' gives information about the search direction, and 'Active set' tells if the true active set is detected or not; moreover, the column labeled by 'Full step $\bar{d}_N^j$' shows if the full semi-smooth Newton step $\bar{d}_N^j$ (in Algorithm 2) is accepted or not. From Tables 2 and 3, we know that the residuals in the last 3 or 2 iterations drop very rapidly as long as the active set is identified. The semi-Newton direction $\bar{d}_N^j$ is used at the final several iterations and fast local convergence is observed.[4]

[3] We remark that ϵ is set to be 10^{-8} only in this subsubsection because it is beneficial to observe the quadratic convergence rate from numerical results. In the rest numerical experiments, ϵ is still set to be 10^{-6}.

[4] One may observe that the convergence in the final stage in Tables 2 and 3 is not indeed quadratic. This is due to the use of CG for the generalized Newton system where only a properly accurate approximation solution is computed.

Table 2 The information of intermediate iterations generated by AASA(Adaptive) on the medium instance of E2

Iter	Res	Step d^k	Active set	Full step $\bar{d}_N^j$
...	...	...	...	...
145	8.13E−02	d_Q^k	Undetected	–
146	3.10E−01	d_N^k	Undetected	–
147	1.74E−01	d_Q^k	Undetected	–
148	1.07E−01	d_N^k	Undetected	–
...	...	...	...	...
154	1.12E−03	d_N^k	Undetected	–
155	1.33E−02	$\bar{d}_N^j$	Undetected	Rejected
156	1.25E−03	d_Q^k	Undetected	–
157	**2.33E−04**	d_N^k	Detected	–
158	**2.31E−06**	$\bar{d}_N^j$	Detected	Accepted
159	**2.14E−08**	$\bar{d}_N^j$	Detected	Accepted
160	**2.09E−10**	$\bar{d}_N^j$	Detected	Accepted

The boldfaced indicates the occurrence of the superlinear convergence

Table 3 The information of intermediate iterations generated by AASA(Adaptive) on the large instance of E2

Iter	Res	Step d^k	Active set	Full step $\bar{d}_N^j$
...	...	...	...	...
300	1.03E−01	d_N^k	Undetected	–
301	6.72E−02	d_N^k	Undetected	–
302	2.75E−02	d_N^k	Undetected	–
303	2.72E−02	d_N^k	Undetected	–
304	2.85E−03	d_N^k	Undetected	–
305	8.36E−03	$\bar{d}_N^j$	Undetected	Rejected
306	6.86E−03	d_Q^k	Undetected	–
307	2.21E−04	d_N^k	Undetected	–
308	3.35E−05	$\bar{d}_N^j$	Undetected	Rejected
309	**3.27E−05**	d_Q^k	Detected	–
310	**3.08E−07**	$\bar{d}_N^j$	Detected	Accepted
311	**2.98E−09**	$\bar{d}_N^j$	Detected	Accepted

The boldfaced indicates the occurrence of the superlinear convergence

6.1.3 Results from medium size problems

We test **E1**–**E6** with n varying from 400 to 900 and n_r from 100 to 500. In this testing, the maximum number of iterations for ISNM, AASA(L-BFGS), AASA(Adaptive), P-BFGS and ADM are set to be 1000, 2000, 2000, 2000 and 3000, respectively, and the maximal allowed CPU consuming time is 1500 seconds

Table 4 Numerical results on **E1** for medium problems

n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)					ADM		P-BFGS		
			Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u	Iter	t	Iter	t	Res
400	100	70,300	71	32	2.5E−08	340	11	9.1E−07	99	5	4.1E−07	12	2	3000†	65	1552	88	9.3E−07
400	200	120,200	78	53	9.5E−09	489	20	8.8E−07	151	10	1.2E−08	21	2	3000†	64	1920	131	9.5E−07
400	300	150,100	92	83	3.1E−08	586	30	9.4E−07	190	15	4.8E−08	30	1	3000†	64	2001†	171	5.6E−06
500	100	90,400	66	31	5.6E−09	313	14	8.9E−07	116	11	8.4E−08	27	3	3000†	94	2001†	247	1.7E−06
500	200	160,300	111	102	1.3E−08	490	30	9.9E−07	161	15	7.6E−07	21	1	3000†	95	2001†	212	5.6E−06
500	300	210,200	120	167	2.2E−08	591	45	9.7E−07	201	24	1.3E−08	21	3	3000†	97	2001†	247	7.3E−05
500	400	240,100	95	133	5.2E−09	671	56	8.1E−07	207	24	2.4E−07	17	2	3000†	97	2001†	265	3.4E−05
600	100	110,500	57	27	1.3E−07	307	19	9.1E−07	97	10	1.4E−08	12	2	3000†	144	2001†	230	3.0E−06
600	200	200,400	129	165	1.4E−08	491	43	1.0E−06	144	19	3.3E−07	14	2	3000†	144	2001†	298	1.2E−05
600	300	270,300	101	164	1.5E−07	664	71	8.3E−07	206	34	6.6E−08	28	2	3000†	146	2001†	355	2.0E−04
600	400	320,200	103	199	1.2E−07	775	92	9.7E−07	199	34	5.5E−07	17	2	3000†	147	2001†	385	2.4E−04
600	500	350,100	139	349	9.7E−08	739	94	9.9E−07	227	40	9.1E−08	20	2	3000†	148	2001†	405	2.7E−04
700	100	130,600	72	54	3.2E−08	298	26	7.3E−07	87	13	1.7E−08	9	2	3000†	207	2001†	334	5.8E−06
700	200	240,500	152	246	9.5E−09	494	58	8.9E−07	160	27	2.8E−08	15	2	3000†	209	2001†	404	3.6E−05
700	300	330,400	111	229	2.7E−08	675	95	9.6E−07	187	47	2.7E−07	28	3	3000†	211	2001†	471	4.1E−04
700	400	400,300	176	548	7.7E−08	748	120	8.1E−07	230	58	3.1E−08	34	1	3000†	212	2001†	525	6.7E−04
700	500	450,200	174	645	7.2E−09	847	146	9.8E−07	226	62	6.5E−08	25	3	3000†	215	2001†	564	6.5E−04
800	100	150,700	81	80	7.0E−08	298	37	9.1E−07	106	21	3.0E−07	13	2	3000†	294	2001†	461	9.6E−06
800	200	280,600	114	201	2.6E−08	516	81	8.9E−07	216	80	7.5E−07	56	5	3000†	297	2001†	561	1.6E−04
800	300	390,500	289	1004	1.2E−07	678	127	9.9E−07	205	63	4.6E−08	24	3	3000†	299	2001†	635	7.3E−04
800	400	480,400	204	834	4.0E−07	866	182	8.3E−07	236	70	9.4E−07	22	2	3000†	303	2001†	698	1.4E−03
800	500	550,300	234	1141	3.3E−07	827	190	9.9E−07	288	104	2.0E−07	30	3	3000†	306	2001†	750	1.3E−03
900	100	170,800	88	107	2.2E−07	329	53	9.7E−07	104	29	1.1E−07	12	5	3000†	397	2001†	601	7.0E−06

Table 4 (continued)

n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)					ADM		P-BFGS		
			Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u	Iter	t	Iter	t	Res
900	200	320,700	167	422	7.3E−09	496	99	9.5E−07	166	57	5.5E−08	29	2	3000†	399	2001†	708	4.2E−04
900	300	450,600	183	635	1.9E−08	716	167	9.8E−07	192	68	4.2E−07	17	2	3000†	402	2001†	792	1.6E−03
900	400	560,500	197	865	3.6E−07	807	212	8.7E−07	249	89	3.7E−07	15	2	3000†	406	2001†	876	3.6E−03
900	500	650,400	217	1137	2.2E−08	854	244	9.1E−07	292	123	2.4E−08	29	2	3000†	413	2001†	940	1.7E−03

The dagger symbol (†) indicates the failure

Table 5 Numerical results on **E2** for medium size problems

n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)					ADM		P-BFGS		
			Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u	Iter	t	Iter	t	Res
400	100	70,300	73	36	1.6E−08	371	12	9.3E−07	104	5	7.1E−07	13	2	3000†	64	2001†	191	3.5E−06
400	200	120,200	88	67	3.3E−08	531	22	8.6E−07	173	11	1.5E−08	20	1	3000†	65	2001†	140	2.6E−06
400	300	150,100	63	47	1.6E−07	575	30	8.5E−07	199	14	3.1E−08	22	1	3000†	65	2001†	171	1.5E−05
500	100	90,400	88	51	4.1E−07	354	16	8.2E−07	108	9	2.7E−08	17	1	3000†	96	2001†	159	1.9E−06
500	200	160,300	70	58	6.7E−08	522	33	8.8E−07	169	17	1.2E−08	18	4	3000†	96	2001†	215	2.1E−05
500	300	210,200	146	200	7.1E−09	619	47	9.3E−07	203	23	6.4E−08	29	2	3000†	97	2001†	246	3.6E−05
500	400	240,100	126	216	5.8E−09	707	59	9.9E−07	215	25	4.6E−08	20	2	3000†	97	2001†	270	5.7E−05
600	100	110,500	68	48	1.1E−07	369	24	9.9E−07	132	18	1.9E−08	34	2	3000†	145	2001†	234	1.4E−05
600	200	200,400	116	168	9.1E−08	543	48	9.0E−07	178	22	2.1E−07	14	2	3000†	147	2001†	304	3.3E−05
600	300	270,300	108	184	3.1E−08	717	77	1.0E−06	232	36	4.2E−08	27	2	3000†	148	2001†	352	1.5E−04
600	400	320,200	121	259	1.5E−08	761	91	8.2E−07	219	38	7.2E−08	22	2	3000†	148	2001†	389	1.8E−04
600	500	350,100	141	339	1.3E−08	741	94	9.3E−07	227	44	2.2E−07	24	3	3000†	150	2001†	403	2.3E−04
700	100	130,600	88	84	3.7E−07	370	33	9.3E−07	108	17	5.6E−08	15	2	3000†	211	2001†	338	3.9E−05
700	200	240,500	151	266	4.5E−07	590	71	8.8E−07	198	45	1.3E−08	39	5	3000†	214	2001†	412	2.0E−04
700	300	330,400	183	541	1.6E−08	691	100	9.1E−07	219	46	9.2E−07	23	2	3000†	214	2001†	481	3.1E−04
700	400	400,300	173	543	1.6E−08	796	128	8.3E−07	209	49	3.0E−07	17	2	3000†	215	2001†	522	6.5E−04
700	500	450,200	157	497	3.2E−08	815	142	9.3E−07	260	66	1.2E−07	23	2	3000†	217	2001†	566	9.4E−04
800	100	150,700	160	250	5.6E−08	345	43	9.2E−07	114	25	5.9E−08	15	2	3000†	299	2001†	467	5.4E−05
800	200	280,600	135	292	4.1E−08	583	93	8.4E−07	156	41	4.0E−07	17	2	3000†	301	2001†	557	5.0E−04
800	300	390,500	167	533	5.0E−09	738	139	9.6E−07	204	56	1.8E−08	17	2	3000†	304	2001†	636	4.9E−04
800	400	480,400	185	714	2.9E−08	747	157	8.9E−07	244	82	4.8E−08	32	3	3000†	316	2001†	702	7.6E−04
800	500	550,300	201	928	8.9E−09	898	206	9.9E−07	248	85	1.1E−07	25	2	3000†	307	2001†	749	1.8E−03
900	100	170,800	91	116	9.2E−08	385	63	9.4E−07	125	36	9.5E−07	19	3	3000†	402	2001†	611	6.5E−05

Table 5 (continued)

n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)					ADM		P-BFGS		
			Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u	Iter	t	Iter	t	Res
900	200	320,700	153	449	1.1E−07	595	121	7.6E−07	198	62	2.1E−08	25	2	3000†	407	2001†	724	1.1E−03
900	300	450,600	236	1028	3.0E−08	674	159	9.8E−07	239	95	4.6E−07	39	2	3000†	410	2001†	805	1.6E−03
900	400	560,500	197	890	1.4E−08	994	273	9.2E−07	257	98	2.2E−07	19	2	3000†	411	2001†	891	1.7E−03
900	500	650,400	217	1241	1.1E−08	920	298	8.8E−07	259	113	7.3E−07	18	2	3000†	429	2001†	871	3.0E−03

The dagger symbol (†) indicates the failure

for all solvers. The numerical results on **E1** and **E2** are listed in Tables 4 and 5, respectively, where the labels 't', 'N' and 'N_u' represent CPU time (in seconds), the number of generalized Newton steps (both d_N^k and $\bar{d}_N^j$), the number of rejected generalized Newton steps ($\bar{d}_N^j$), respectively. It should be pointed out that the KKT residuals associated with each solver are recalculated by (3.4) for measuring the accuracy of the computed solutions; moreover, since ADM only gives the primal solution, which cannot be used to calculate the KKT residual ϱ^k directly, we omit the final residuals associated with ADM in Tables 4 and 5.

We observed from Tables 4 and 5 that ISNM, AASA(L-BFGS) and AASA(Adaptive) succeed in solving all the instances of **E1** and **E2**, while the other two solvers fail (labeled by †) in some instances. Particularly, ADM fails on all the cases as it cannot find approximate solutions within the prescribed accuracy in 3000 iterations. P-BFGS fails in almost all the instances due to either exceeding the maximum number (2000) of iterations. Also, we can see that the number N_u in Tables 4 and 5 is no more than 5, i.e., no more than 5 generalized Newton steps are rejected for each instance. The number N in Tables 4 and 5 is less than 40, which means that no more than 40 generalized Newton equations are solved for each instance. The numerical results in Tables 4 and 5 show the efficiency of AASA(Adaptive) in terms of CPU time.

The numerical results on **E3**–**E6** are presented on Table 6. Again, we can see that ASSA(adaptive) successfully solves all the problems efficiently, while P-BFGS and ADM both fail in most cases. ISNM generally terminates in a reasonable number of iterations but needs more CPU consuming times comparing with ASSA(adaptive) and ASSA(L-BFGS). Also, the magnitudes of N and N_u here are similar to those in Tables 4 and 5.

6.1.4 Results from large-scale problems

Next, we increase n and n_r to use various n ranging from 1000 to 2000, and n_r from 300 to 800. In this case, we also extend the maximum CPU time to 3600 seconds. The numerical results for **E1** and **E2** are omitted due to its similarity to those of **E3**–**E6** in Table 7. Also, results from ADM and P-BFGS are exclusive due to their poor performance. It can be seen from Table 7 that, as n and n_r increase, the efficiency of AASA(Adaptive) gets more manifest.

Without the restriction on the consuming CPU time and the number of iterations, we provide two cases for medium-scale problems on **E5** and **E6**: one (called case (i)) is $(n, n_r) = (1000, 500)$ and the other (called case (ii)) is $(n, n_r) = (2000, 500)$. Figures 1 and 2 demonstrate the performances of three solvers in terms of the number of iterations, the residuals and the CPU time.

Table 6 Numerical results on **E3–E6** with $n_r = 500$

Type	n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)					ADM		P-BFGS		
				Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u	Iter	t	Iter	t	Res
E3	600	500	350,100	313	348	1.4E−07	878	111	9.5E−07	207	29	9.4E−09	6	1	3000†	153	819	156	8.4E−07
	700	500	450,200	583	939	2.0E−08	979	167	9.3E−07	247	46	7.4E−07	6	1	3000†	216	873	228	9.4E−07
	800	500	550,300	476	948	4.9E−07	1113	256	9.6E−07	230	60	2.5E−08	6	1	3000†	307	2001†	> 1500	5.5E−06
	900	500	650,400	566	> 1500	1.7E+00	1109	313	9.9E−07	257	81	6.9E−08	6	1	3000†	415	894	404	9.0E−07
E4	600	500	350,100	308	346	4.7E−09	1032	129	8.9E−07	237	33	2.1E−07	5	0	3000†	150	996	191	9.6E−07
	700	500	450,200	606	984	2.0E−07	1064	182	9.5E−07	251	46	9.2E−07	5	0	3000†	217	819	211	8.2E−07
	800	500	550,300	514	1043	4.6E−09	1157	261	9.3E−07	254	65	1.3E−08	7	1	3000†	313	2001†	> 1500	1.4E−06
	900	500	650,400	555	> 1500	1.9E+00	1148	327	9.8E−07	274	85	7.1E−07	6	1	3000†	420	948	418	7.9E−07
E5	600	500	350,100	176	320	1.1E−07	901	115	8.1E−07	275	52	1.7E−07	32	2	3000†	151	2001†	402	1.0E−04
	700	500	450,200	254	657	4.1E−07	892	157	9.4E−07	275	62	8.3E−08	19	2	3000†	219	2001†	557	4.6E−04
	800	500	550,300	287	1021	2.2E−08	1089	251	1.0E−06	286	100	3.0E−08	33	4	3000†	320	2001†	742	7.3E−04
	900	500	650,400	328	1483	4.3E−09	963	278	9.7E−07	317	116	4.8E−07	19	1	3000†	421	2001†	938	1.3E−03
E6	600	500	350,100	218	425	4.9E−08	802	103	9.8E−07	256	43	4.2E−07	28	1	3000†	152	2001†	403	1.2E−04
	700	500	450,200	250	649	3.8E−07	838	146	8.8E−07	245	54	6.1E−07	17	2	3000†	219	2001†	559	3.6E−04
	800	500	550,300	300	1056	6.2E−09	973	226	7.6E−07	281	85	4.2E−07	21	2	3000†	317	2001†	747	1.0E−03
	900	500	650,400	323	1436	1.9E−08	1102	322	8.2E−07	312	131	6.2E−08	26	2	3000†	418	2001†	934	1.1E−03

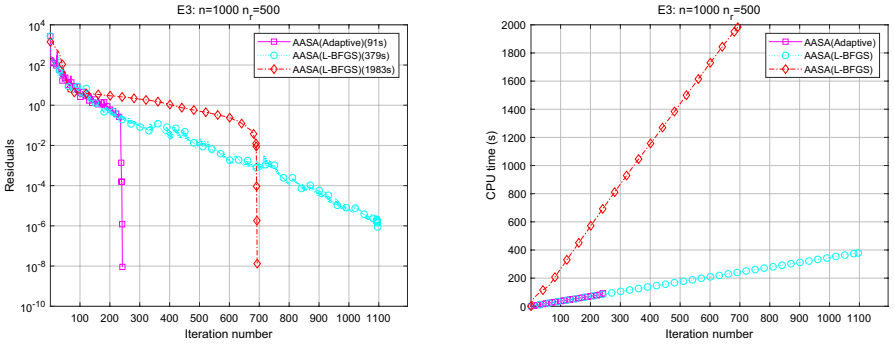
The dagger symbol (†) indicates the failure

Table 7 Numerical results on **E3–E6** for large-scale problems

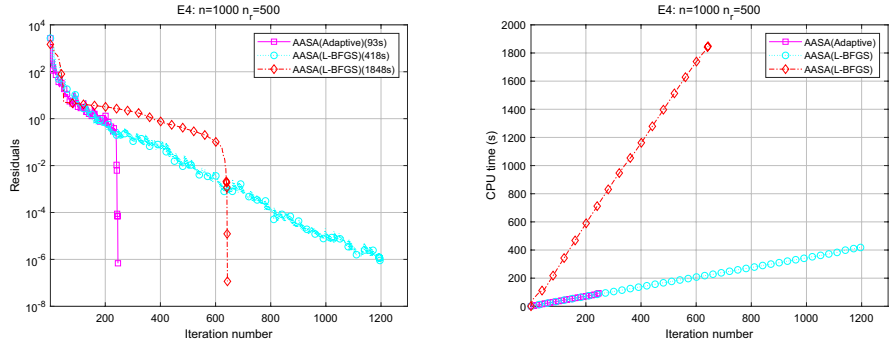
Type	n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)				
				Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u
E3	1000	300	510,700	724	1768	3.7E−07	1012	295	9.9E−07	191	65	5.6E−07	5	1
	1000	600	840,400	796	2554	4.6E−08	1318	483	9.6E−07	218	99	6.1E−08	11	5
	1000	800	960,200	940	> 3600	6.3E−01	1310	523	9.0E−07	289	123	3.8E−07	5	0
	1500	300	811,200	648	> 3600	7.1E−01	880	659	9.7E−07	219	174	1.7E−08	5	1
	1500	600	1,440,900	467	> 3600	6.9E+00	1331	1197	9.1E−07	311	300	3.6E−07	5	0
	1500	800	1,760,700	415	> 3600	1.2E+01	1809	1758	9.8E−07	392	405	2.8E−07	5	1
	2000	300	1,111,700	307	> 3600	6.5E+00	734	1176	9.8E−07	175	307	2.7E−08	5	1
	2000	600	2,041,400	256	> 3600	1.7E+01	1261	2336	9.9E−07	316	624	1.6E−07	5	1
	2000	800	2,561,200	217	> 3600	2.2E+01	1643	3279	9.2E−07	381	834	3.8E−08	5	1
E4	1000	300	510,700	442	963	3.1E−08	850	246	9.4E−07	218	72	2.2E−08	5	1
	1000	600	840,400	1049	> 3600	2.6E−02	1162	426	9.8E−07	305	122	1.6E−07	5	0
	1000	800	960,200	820	> 3600	3.8E+00	1413	556	9.6E−07	331	147	4.9E−07	8	3
	1500	300	811,200	605	> 3600	1.4E+00	824	610	9.9E−07	217	177	9.2E−08	5	1
	1500	600	1,440,900	442	> 3600	9.0E+00	1407	1281	9.9E−07	322	322	4.8E−07	7	3
	1500	800	1,760,700	401	> 3600	1.2E+01	1738	1733	9.4E−07	328	339	1.2E−08	6	0
	2000	300	1,111,700	291	> 3600	7.0E+00	894	1458	9.0E−07	219	386	2.0E−08	5	1
	2000	600	2,041,400	246	> 3600	1.7E+01	1429	2677	8.8E−07	358	696	3.2E−07	5	1
	2000	800	2,561,200	215	> 3600	2.2E+01	1678	3373	9.6E−07	450	949	3.3E−07	5	0

Table 7 (continued)

Type	n	n_r	m	ISNM			AASA(L-BFGS)			AASA(Adaptive)				
				Iter	t	Res	Iter	t	Res	Iter	t	Res	N	N_u
E5	1000	300	510,700	656	2741	9.7E−09	705	206	9.7E−07	248	118	6.8E−08	36	4
	1000	600	840,400	368	2084	1.6E−07	1224	459	8.8E−07	310	146	7.4E−07	19	2
	1000	800	960,200	514	> 3600	1.3E−01	1084	442	9.1E−07	342	204	2.3E−08	37	3
	1500	300	811,200	465	> 3600	1.5E−01	888	667	8.8E−07	230	206	3.7E−07	13	2
	1500	600	1,440,900	315	> 3600	3.3E+00	1263	1149	9.4E−07	352	398	9.5E−07	27	2
	1500	800	1,760,700	270	> 3600	8.0E+00	1524	1513	9.6E−07	426	498	7.5E−07	21	2
	2000	300	1,111,700	267	> 3600	3.5E+00	727	1195	9.9E−07	223	429	6.0E−07	11	2
	2000	600	2,041,400	193	> 3600	1.3E+01	1575	2983	7.7E−07	400	1005	1.9E−07	38	3
	2000	800	2,561,200	171	> 3600	1.6E+01	1669	3378	9.9E−07	450	1163	1.6E−08	30	3
E6	1000	300	510,700	530	2259	3.8E−07	760	224	8.1E−07	279	114	1.4E−07	23	3
	1000	600	840,400	366	2134	1.3E−07	1230	469	9.7E−07	334	205	2.7E−07	47	2
	1000	800	960,200	489	3376	1.9E−07	1459	593	9.6E−07	385	215	9.5E−08	20	2
	1500	300	811,200	411	> 3600	1.0E+00	776	579	1.0E−06	368	359	2.0E−07	28	2
	1500	600	1,440,900	301	> 3600	3.7E+00	1204	1102	8.9E−07	356	447	1.3E−08	33	5
	1500	800	1,760,700	266	> 3600	8.3E+00	1309	1309	9.5E−07	420	568	2.8E−07	37	5
	2000	300	1,111,700	237	> 3600	3.9E+00	849	1399	9.4E−07	252	484	1.8E−07	13	2
	2000	600	2,041,400	188	> 3600	1.2E+01	1713	3369	9.8E−07	381	864	3.2E−08	19	2
	2000	800	2,561,200	170	> 3600	1.6E+01	1764	> 3600	1.1E−03	460	1282	5.5E−07	46	1



(a) Residuals on **E3** **(b)** CPU time on **E3**



(c) Residuals on **E4** **(d)** CPU time on **E4**

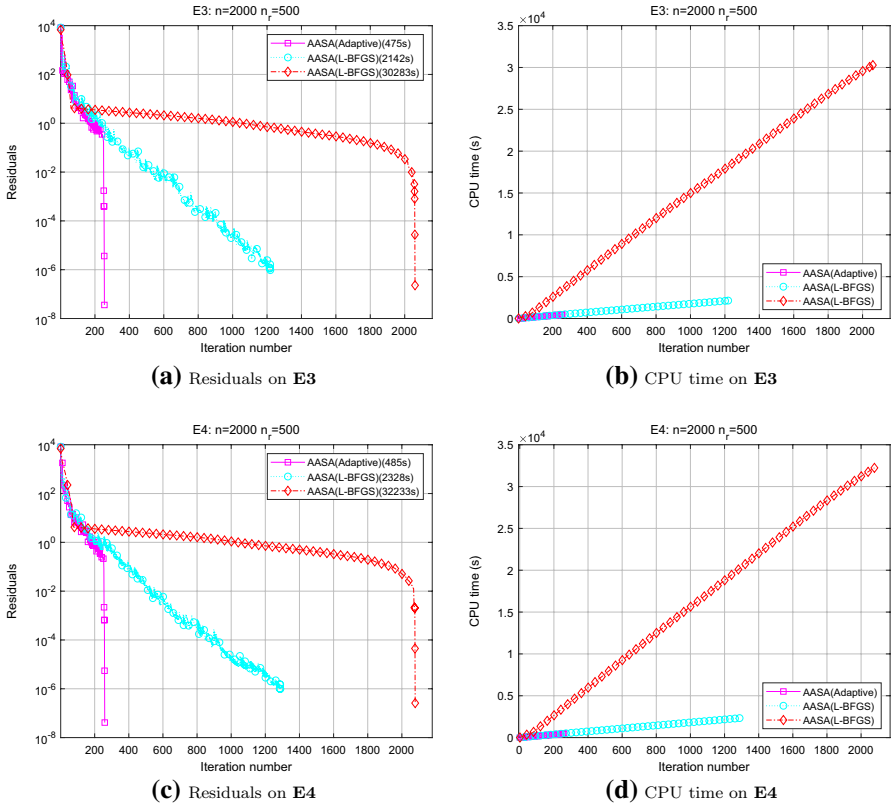
Fig. 1 Performance profile for residuals (left) and CPU time (right) on **E3** (top) and **E4** (bottom)

It can be seen from Fig. 1a that the residual corresponding to AASA(Adaptive) gradually decreases as the number of iterations increases from 1 to 240 (the L-BFGS acceleration works in this phase), and then rapidly drops to the preset tolerance (the semi-smooth Newton acceleration works at that stage). The residual corresponding to ISNM declines slowly at the beginning, and drops rapidly in the last period. This also verifies the fast convergence of the semi-smooth Newton method numerically. From Fig. 1b, the three curves of CPU time are nearly linear. Similar observation can be seen from Fig. 1c and d and from Fig. 2 for case (ii). Again, the comparison also indicates that AASA(Adaptive) is the most efficient one.

6.2 Real examples

In this subsection, we test two real world instances whose data are obtained from financial markets.

In calculating value at risk in financial markets, the correlation matrix [28] is a critical factor. For example, it is noticed that the market correlation structure serves as a reflection



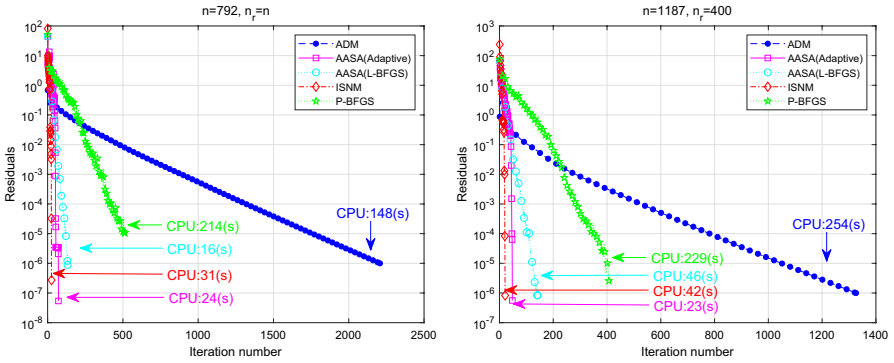
(a) Residuals on **E3** **(b)** CPU time on **E3**

(c) Residuals on **E4** **(d)** CPU time on **E4**

Fig. 2 Performance profile for residuals (left) and CPU time (right) on **E3** (top) and **E4** (bottom)

of the Great Crash, which changes around the Great Crash in several aspects [11] like abrupt regime changes [29, 30] or periodic economic cycle. In particular, the average correlation coefficient after the critical point of crash usually gets higher than that before the crash [18, 19] and will approach to a steady state gradually. Moreover, the correlation coefficients can be used in the stress testing: the correlation coefficients between certain underlying assets and others are adjusted largely to simulate the stress scenarios [28]; in financial industry, people usually try to find an approximation of the restricted correlation matrix in stress testing scenario in order to recover the adjusted matrix back to a correlation matrix. Such task can be mathematically formulated as the model (6.1).

In our numerical verification, the correlation matrices to be approximated are calculated from the sample data in the Shenzhen Stock Exchange and the Shanghai Stock Exchange in China. For the constraints of (6.1), we notice that particular restrictions may be associated with a historical stressful event (such as the 1987 stock market crash and 2008 economic crisis), or can be a set of hypothetical changes related with same possible future stressful market event [13]. Generally, identifying accurately the stress events set and restricting the correlation coefficients between stress events and other underlying events [13] are very difficult, and



(a) Real Example (**E7**) **(b)** Real Example (**E8**)

Fig. 3 Performance profile on **E7** (left) and **E8** (right)

therefore, following [27], in our experiments, we just randomly generate the constraints and their positions to imitate the stress testing scenarios.

Precisely, for the first real world example (denoted by **E7**), we follow the way in [27] and calculate the data matrix using the daily closing price of 792 stocks listed in Shanghai Stock Exchange (from September 2016 to September 2018). For the constraints, we restrict all entries of the correlation matrix to be no less than $-0.1 * \texttt{rand} + 0.6$ and there are no restrictions on the upper bounds, which can mimic some sort of scenarios in the period of economic depression or economic crisis.

For the second real world example (denoted by **E8**), the data matrix is obtained from 1187 stocks in Shenzhen Stocks Exchange (from September 2016 to September 2018). Again, we follow the way in [27] to generate the correlation matrix. For the constraints, here, we choose 400 random positions at each row of the matrix X, and set $-0.3 * \texttt{rand} + 0.1$ and $0.3 * \texttt{rand} + 0.1$ as the lower and upper bounds corresponding to the selected positions, respectively.

We plotted Fig. 3 to show how the residual changes against the number of iterations when applying the five solvers to solve **E7** and **E8**. It can be seen that all the solvers succeed in solving the two real data examples within the maximum number of iterations. We remark that the residual used in ADM is different from others but it is a default option in [12]. Figure 3 (both left and right) shows that the residuals corresponding to AASA(Adaptive) and ISNM drop very fast as the number of iterations increases. This is explainable because AASA(Adaptive) and ISNM both make use of the second-order information, while the other three solvers (AASA(L-BFGS), P-BFGS and ADM) are based on the first-order information which generally need less computational effort per iteration but converges (in the sense of the residuals) slower. Besides the behavior of the residual against the number of iterations, we also reported the consuming CPU times in Fig. 3 for the overall performance of these algorithms. One can see that AASA(Adaptive) in general is one of the best and can be an efficient and robust approach for solving the problem (6.1).

7 Conclusion

In this paper, we have presented a type of active-set algorithm for solving a generalization of least squares semidefinite programming. Treating it from its equivalent dual form, our algorithm begins with estimating the active/inactive sets using the BB step information, and then adaptively applies the L-BFGS and the semi-smooth Newton methods to accelerate the convergence of the free variables. Under some mild conditions, the proposed algorithm is proved to be globally convergent, and fast local convergence is guaranteed in a refined adaptive strategy. Numerical experiments on both synthetic and real world data problems are conducted. The reported numerical results are preliminary but very promising, indicating our proposed AASA(Adaptive) algorithm is an efficient and robust approach for this programming.

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

References

1. Barzilai, J., Borwein, J.M.: Two point step size gradient methods. IMA J. Numer. Anal. **8**, 141–148 (1988)
2. Boyd, S., Xiao, L.: Least squares covariance matrix adjustment. SIAM J. Matrix Anal. Appl. **27**, 532–546 (2005)
3. Chen, X., Qi, H., Tseng, P.: Analysis of nonsmooth symmetric-matrix-valued functions with applications to semidefinite complementarity problems. SIAM J. Optim. **13**, 960–985 (2003)
4. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)
5. Dai, Y.H.: Alternate step gradient method. Optimization **52**, 395–415 (2003)
6. Facchinei, F.: Minimization of SC^1 functions and the Maratos effect. Oper. Res. Lett. **17**(3), 131–137 (1995)
7. Facchinei, F., Fischer, A., Kanzow, C.: On the accurate identification of active constraints. SIAM J. Optim. **9**(1), 14–32 (1998)
8. Gabay, D.: 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, pp. 299–331. North-Holland, Amsterdam (1983)
9. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximations. Comput. Math. Appl. **2**, 17–40 (1976)
10. Gao, Y., Sun, D.F.: Calibrating least squares semidefinite programming with equality and inequality constraints. SIAM J. Matrix Anal. Appl. **31**, 1432–1457 (2009)
11. Han, R.Q., Xie, W.J., Xiong, X.: Market correlation structure changes around the great crash: a random matrix theory analysis of the Chinese stock market. Fluct. Noise Lett. **16**(02), 1750018 (2017)
12. He, B.S., Xu, M.H., Yuan, X.M.: Solving large-scale least squares covariance matrix problems by alternating direction methods. SIAM J. Matrix Anal. Appl. **32**, 136–152 (2011)
13. Kupiec, P.H.: Stress testing in a value-at-risk framework. J. Deriv. **6**, 7–24 (1998)
14. Kelley, C.T.: Iterative Methods for Optimization, pp. 102–104. SIAM, Philadelphia (1999)
15. Li, Q.N., Li, D.H.: A projected semi-smooth Newton method for problems of calibrating least squares covariance matrix. Oper. Res. Lett. **39**, 103–108 (2011)
16. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large-scale optimization. Math. Program. **45**, 503–528 (1989)
17. Malick, J.: A dual approach to semidefinite least squares problems. SIAM J. Matrix Anal. Appl. **26**, 272–284 (2004)
18. Nobi, A., Maeng, S.E., Ha, G.G., Lee, J.W.: Random matrix theory and cross-correlations in global financial indices and local stock market indices. J. Korean Phys. Soc. **62**(4), 569–574 (2013)

19. Nobi, A., Maeng, S.E., Ha, G.G., Lee, J.W.: Effects of global financial crisis on network structure in a local stock market. Phys. A **407**, 135–143 (2014)
20. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Berlin (2006)
21. Qi, L.Q.: Convergence analysis of some algorithms for solving nonsmooth equations. Math. Oper. Res. **18**, 227–244 (1993)
22. Qi, L.Q.: Superlinearly convergent approximate Newton methods for LC^1 optimization problems. Math. Program. **64**(1–3), 277–294 (1994)
23. Qi, L.Q., Sun, J.: A nonsmooth version of Newton's method. Math. Program. **58**, 353–367 (1993)
24. Qi, H.D., Sun, D.F.: A quadratically convergent Newton method for computing the nearest correlation matrix. SIAM J. Matrix Anal. Appl. **28**, 360–385 (2006)
25. Rockafellar, R.T.: Conjugate Duality and Optimization. SIAM, Philadelphia (1974)
26. Schwertman, N.C., Allen, D.M.: Smoothing an indefinite variance–covariance matrix. J. Stat. Comput. Simul. **9**, 183–194 (1979)
27. Shen, C.G., Fan, C.X., Wang, Y.L., Xue, W.J.: Limited memory BFGS algorithm for the matrix approximation problem in Frobenius norm. Comput. Appl. Math. **39**, 43 (2020)
28. So, M.K.P., Wang, J., Asai, M.: Stress testing correlation matrices for risk management. North Am. J. Econ. Finance **26**, 310–322 (2013)
29. Sornette, D.: Critical market crashes. Phys. Rep. **378**(1), 1–98 (2003)
30. Sornette, D.: Why Stock Markets Crash: Critical Events in Complex Financial Systems. Princeton University Press, Princeton (2017)
31. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11**, 625–653 (1999)
32. Sun, D.F., Sun, J.: Semi-smooth matrix valued functions. Math. Oper. Res. **27**, 150–169 (2002)
33. Sun, Y.F., Vandenberghe, L.: Decomposition methods for sparse matrix nearness problems. SIAM J. Matrix Anal. Appl. **36**, 1691–1717 (2015)
34. Tütüncü, R.H., Toh, K.C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using SDPT3. Math. Program. **95**, 189–217 (2003)
35. Ye, C.H., Yuan, X.M.: A descent method for structured monotone variational inequalities. Optim. Methods Softw. **22**, 329–338 (2007)
36. Zhang, H., Hager, W.W.: A nonmonotone line search technique and its application to unconstrained optimization. SIAM J. Optim. **14**, 1043–1056 (2004)
37. Zhou, B., Gao, L., Dai, Y.H.: Gradient methods with adaptive step-sizes. Comput. Optim. Appl. **35**, 69–86 (2006)