Journal of Optimization Theory and Applications (2025) 204:9
https://doi.org/10.1007/s10957-024-02592-2

An Accelerated Proximal Alternating Direction Method of Multipliers for Optimal Decentralized Control of Uncertain Systems

Bo Yang[1] · Xinyuan Zhao[1] · Xudong Li[2] · Defeng Sun[3]

Received: 29 January 2024 / Accepted: 5 October 2024

Abstract
To ensure the system stability of the $\mathcal{H}_2$-guaranteed cost optimal decentralized control (ODC) problem, we formulate an approximate semidefinite programming (SDP) problem that leverages the block diagonal structure of the decentralized controller's gain matrix. To minimize data storage requirements and enhance computational efficiency, we employ the Kronecker product to vectorize the SDP problem into a conic programming (CP) problem. We then propose a proximal alternating direction method of multipliers (PADMM) to solve the dual of the resulting CP problem. By using the equivalence between the semi-proximal ADMM and the (partial) proximal point algorithm, we identify the non-expansive operator of PADMM, enabling the use of Halpern fixed-point iteration to accelerate the algorithm. Finally, we demonstrate that the sequence generated by the proposed accelerated PADMM exhibits a fast convergence rate for the Karush–Kuhn–Tucker residual. Numerical experiments confirm that the accelerated algorithm outperforms the well-known COSMO, MOSEK, and SCS solvers in efficiently solving large-scale CP problems, particularly those arising from $\mathcal{H}_2$-guaranteed cost ODC problems.

Communicated by Nguyen Mau Nam.

✉ Xinyuan Zhao
xyzhao@bjut.edu.cn

Bo Yang
bbo_yang@163.com

Xudong Li
lixudong@fudan.edu.cn

Defeng Sun
defeng.sun@polyu.edu.hk

1 Beijing University of Technology, Beijing 100124, China

2 Fudan University, Shanghai 200433, China

3 The Hong Kong Polytechnic University, Hong Kong, China

Published online: 28 December 2024

Keywords Semidefinite programming · Conic programming · Acceleration · Halpern iteration · Decentralized control

Mathematics Subject Classification 90C22 · 90C25 · 90C06 · 49M29

1 Introduction

Numerous complex real-world systems, including aircraft formations, automated highways, and power systems, are comprised of a multitude of interconnected subsystems. Within these systems, the controller is limited to accessing only the status information of individual subsystems. To mitigate the computational and communication complexity inherent in the overall control process, the optimal decentralized control (ODC) problem with structural constraints is employed to stabilize the system while achieving optimal performance. Given its critical importance, the ODC problem has garnered significant research interest since the late 1970s [29, 42].

The design of a centralized controller without structural constraints for control problems such as the linear quadratic regulator, linear quadratic Gaussian, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems typically involves solving algebraic Riccati equations. However, extending this technique to decentralized control scenarios is challenging due to the specific sparsity constraints imposed on the gain matrix of the decentralized controller. Moreover, it is well-established that designing a globally optimal decentralized controller is generally an NP-hard problem [44, 45]. Significant efforts have been devoted to addressing this complex problem, particularly for special system types such as spatially distributed systems [32], dynamically decoupled systems [1], weakly connected systems [41], and strongly connected systems [20]. Early approaches focused on parameterization techniques [6, 15], which later evolved into matrix optimization methods [40, 47]. While these methods have certain advantages, they are often computationally intensive and do not always guarantee global optimality. Additionally, they can be particularly resource-demanding for large-scale systems and susceptible to numerical instability. These limitations underscore the need for alternative approaches in the design of decentralized controllers.

Recent advancements in convex optimization have shifted the focus of control synthesis towards finding convex formulations that can be solved efficiently. Various convex relaxation techniques, such as those based on linear matrix inequalities, semidefinite programming (SDP) [43], and second-order cone programming [26], have become popular for addressing the ODC problems. Additionally, quadratic invariance (QI) has been identified as a necessary and sufficient condition for ensuring that the set of Youla parameters is convex [21], thereby guaranteeing the existence of a sparse controller [13, 22, 28, 39]. Recently, the concept of sparse invariance was introduced by [12] as a method to design optimally distributed controllers and overcome the restrictions of QI. However, verifying these invariance conditions can be computationally expensive, if not infeasible, for practical applications.

To enhance computational effectiveness, more recently, by adapting the inexact symmetric Gauss–Seidel (sGS) decomposition technique developed in [23, 24] for solving multi-block convex problems, the sGS semi-proximal augmented Lagrangian

method was first used by [30, 49] to solve conic programming (CP) relaxation of the $\mathcal{H}_2$-guaranteed cost ODC problem. This method is effective in maintaining a decentralized structure while ensuring robust stability and performance. However, the computational efficiency remains suboptimal, potentially due to the fact that the algorithm has to compute the projections onto the semidefinite cones twice in each iteration. Therefore, there is a strong motivation to design a more efficient algorithm for solving the $\mathcal{H}_2$-guaranteed cost ODC problem.

Acceleration techniques play a crucial role in enhancing algorithmic efficiency and have gained significant attention in optimization and related fields. Inspired by Nesterov's accelerated gradient method [34], Kim [18] introduced an accelerated scheme of the proximal point algorithm (PPA) to solve the maximally monotone inclusion problem. This accelerated scheme exhibits a rapid $\mathcal{O}(1/k)$ convergence rate for the fixed-point residual of the corresponding maximally monotone operator, where k denotes the number of iterations. Notably, solving the maximal monotone inclusion problem can be equivalently reformulated as finding the fixed point of a non-expansive operator [3]. A more general Halpern iteration was introduced earlier [16] for approximating the fixed point of a non-expansive operator. Specifically, for a non-expansive mapping T on the Hilbert space $\mathcal{H}$ and a fixed $x_0 \in \mathcal{H}$, the Halpern fixed-point iteration is given by:

$$x_{k+1} := \lambda_k x_0 + (1 - \lambda_k) T(x_k),$$

where the stepsizes $\lambda_k \in (0, 1)$. Recently, Lieder [27] demonstrated that the Halpern iteration with stepsizes $\lambda_k = 1/(k + 2)$ achieves an accelerated convergence rate of $\mathcal{O}(1/k)$ for the norm of the fixed-point residual in Hilbert space. The connection between the Halpern iteration with stepsizes $\lambda_k = 1/(k + 2)$ [16, 27] and Kim's acceleration [18] was further explored in [4, Proposition 5]. Moreover, Contreras and Cominetti in [4, Proposition 2] established that the best possible rate for general Mann iterations, including the Halpern iteration, in normed spaces is bounded from below by $\mathcal{O}(1/k)$. Consequently, employing the Halpern iteration or Kim's acceleration appears natural for enhancing the convergence rate of the fixed-point residual. Following this line of research, an efficient Halpern-Peaceman-Rachford algorithm was proposed by Zhang et al. [48] for solving the two-block convex programming problems, including the Wasserstein barycenter problem. This algorithm achieved an appealing $\mathcal{O}(1/k)$ non-ergodic convergence rate concerning the Karush–Kuhn–Tucker (KKT) residual. Additionally, Kim [18] proposed an accelerated alternating direction method of multipliers (ADMM) and achieved an $\mathcal{O}(1/k)$ convergence rate concerning primal infeasibility. However, the convergence of these accelerated methods is contingent on specific assumptions, such as the full column rank of the coefficient matrix of the constraints or the strong convexity of the objective functions [10]. It is important to note that the CP problem derived from the $\mathcal{H}_2$-guaranteed cost ODC problem may not necessarily satisfy these assumptions. To tackle this issue, we introduce an innovative accelerated proximal ADMM (PADMM) based on the Halpern iteration for solving the CP problem. Specifically, proximal terms are integrated to ensure the existence of optimal solutions to the corresponding subproblems, and the Halpern iteration [27] is utilized as an acceleration strategy. In comparison to the non-ergodic $\mathcal{O}(1/\sqrt{k})$ iter-

ation complexity of the majorized ADMM presented in [5], this paper demonstrates that the proposed accelerated PADMM achieves a faster $\mathcal{O}(1/k)$ convergence rate for the fixed-point KKT residual.

In this paper, we approximate the ODC problem with an SDP problem by using a parameterized approach that preserves the block-diagonal structure of the feedback matrix. Additionally, the stability of the system with parameter uncertainties is guaranteed [30]. To enhance the algorithm's efficiency and minimize matrix operations, we transform the original SDP problem into an equivalent vector form, and the optimal solutions are efficiently obtained by using the accelerated PADMM. The contributions made in this paper to the current literature can be summarized as follows:

(i) We establish an equivalence between the semi-proximal ADMM and the (partial) PPA. Utilizing this equivalence, we propose an accelerated PADMM by applying the Halpern iteration to the (partial) PPA. The proposed accelerated PADMM demonstrates a convergence rate of $\mathcal{O}(1/k)$ with respect to the norm of the KKT residual.

(ii) By exploiting the sparsity of the problem data, we introduce a lifting technique to solve the linear systems of subproblems in the (accelerated) PADMM. This technique enhances the efficiency of the proposed accelerated PADMM in addressing large-scale CP problems arising from the ODC problem.

(iii) We implement the accelerated PADMM in C (using the MSVC compiler) and compare its performance to COSMO [14], MOSEK [31], and SCS [36] in solving small-medium and large-scale problems[1]. Extensive numerical results demonstrate the efficiency and robustness of the proposed algorithm.

The remainder of this paper is organized as follows: In Sect. 2, we first introduce the generic form of the $\mathcal{H}_2$-guaranteed cost ODC problem for uncertain systems. We then relax this problem to a CP formulation. We establish the equivalence between the semi-proximal ADMM and the (partial) PPA in Sect. 3. Building on this equivalence, we propose an accelerated PADMM for solving the CP problem. In Sect. 4, we provide a fast implementation of the proposed algorithm for solving the CP problem. Section 5 presents extensive numerical experiments that demonstrate the superiority of our algorithm compared to existing methods. Finally, we conclude the paper in Sect. 6.

Notation. The notations used in this paper are defined as follows:

- Let $\mathbb{R}^n$ denote the n-dimensional real space. The identity operator, denoted by $\mathcal{I}$, is the operator on $\mathbb{R}^n$ such that $\mathcal{I}v = v, \forall\, v \in \mathbb{R}^n$.
- $\mathbb{S}^n$ represents the set of all $n \times n$ real symmetric matrices; $\mathbb{S}^n_{++}$ ($\mathbb{S}^n_{+}$) is the cone of positive (semidefinite) matrices in $\mathbb{S}^n$ with the trace inner product $\langle \cdot, \cdot \rangle$ and the Frobenius norm $\| \cdot \|$. We may sometimes write $X \succ 0$ ($X \succeq 0$) to indicate that $X \in \mathbb{S}^n_{++}$ ($\mathbb{S}^n_{+}$). We use $\otimes$ to denote the Kronecker product of matrices.
- The block diagonal matrix with diagonal entries $A_1,\ A_2,\ \ldots,\ A_m$ is denoted by **blkdiag** $\{A_1,\ A_2,\ \ldots,\ A_m\}$, and **vec**$(A)$ denotes the column vector formed by stacking the columns of A sequentially..
- The vector $e_k \in \mathbb{R}^{p\times 1}$ is defined as having 1 in the kth position and 0 elsewhere.

[1] We say a problem is of small scale if the number of variables/constraints is less than 5000, medium scale if the number is between 5,000 and 150,000, and large scale if the number is larger than 150,000.

- For a set $\mathcal{X} \subseteq \mathbb{R}^n$, $\delta_{\mathcal{X}}$ is the indicator function for $\mathcal{X}$, i.e., $\delta_{\mathcal{X}}(x) = 0$ if $x \in \mathcal{X}$ and $\delta_{\mathcal{X}}(x) = +\infty$ if $x \notin \mathcal{X}$. For a closed convex set $\mathcal{D}$, the Euclidean projector onto $\mathcal{D}$ is defined by $\Pi_{\mathcal{D}}(x) := \underset{s\in\mathcal{D}}{\operatorname{argmin}}\{\|s - x\|\}$.
- The set of fixed points of an operator $\mathcal{T} : \mathcal{X} \to \mathcal{X}$ is denoted by $\operatorname{Fix}(\mathcal{T})$, i.e., $\operatorname{Fix}(\mathcal{T}) := \{x \in \mathcal{X} \mid x = \mathcal{T}(x)\}$.

2 Optimal Decentralized Control Problem

In this section, we first introduce the ODC problem and subsequently present an approximated CP model for the ODC problem.

2.1 Problem Statement

Consider a linear time-invariant (LTI) system with additive disturbance in the following compact form:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_2u(t) + B_1w(t), \\ z(t) &= Cx(t) + Du(t) \end{aligned} \tag{1}$$

with a static state feedback controller

$$u(t) = -Kx(t). \tag{2}$$

Here, $x(t) \in \mathbb{R}^n$ represents the state vector, $u(t) \in \mathbb{R}^m$ is the control input vector, $w(t) \in \mathbb{R}^l$ denotes the exogenous disturbance, and $z(t) \in \mathbb{R}^q$ is the controlled output. The matrices $A \in \mathbb{R}^{n\times n}$, $B_1 \in \mathbb{R}^{n\times l}$, $B_2 \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{q\times n}$, and $D \in \mathbb{R}^{q\times m}$ are system parameters. The feedback gain matrix $K \in \mathbb{R}^{m\times n}$ is assumed to have a decentralized structure, as defined by:

$$K = \begin{bmatrix} K_1 & & 0 \\ & \ddots & \\ 0 & & K_m \end{bmatrix}, \tag{3}$$

where $K_i \in \mathbb{R}^{1\times D_i}$, for $i = 1, \ldots, m$, and $\sum_{i=1}^m D_i = n$.

Assumption 2.1 The state matrix C and control matrix D satisfy $C^\top D = 0$, implying no cross-weighting between the state variables and control variables. And the control weighting matrix D satisfies $D^\top D \succ 0$, under the standard assumptions that (A, B_2) is stabilizable and (A, C) is detectable.

The objective function associated with the linear systems (1) and (2) is defined as

$$J := \int z(t)^\top z(t)dt. \tag{4}$$

We can convert the linear system (1) to the frequency domain by taking the Laplace transform [2, Appendix C] as follows:

$$sX(s) - sx(0) = AX(s) + B_2U(s) + B_1W(s),$$
$$Z(s) = CX(s) + DU(s).$$

Substituting the Eq. (2) into the above expressions, we obtain

$$X(s) = (sI - A + B_2K)^{-1}sx(0) + (sI - A + B_2K)^{-1}B_1W(s),$$
$$Z(s) = (C-DK)(sI-A+B_2K)^{-1}sx(0)+(C - DK)(sI - A+B_2K)^{-1}B_1W(s).$$

Hence, the transfer function of the state-space model (1) and (2) with $x(0) = 0$ is

$$H(s) = (C - DK)(sI - A + B_2K)^{-1}B_1.$$

Motivated by the optimal H_2 controller parametrization results in Doyle et al. [7], minimizing the objective function (4) can be converted to minimizing the following $\mathcal{H}_2$-guaranteed cost ODC problem:

$$\min_{K\in\mathscr{T}} \|H(s)\|_2^2 = \mathrm{Tr}((C - DK)W_c(C - DK)^\top), \tag{5}$$

where W_c is the controllability Gramian matrix related to the closed-loop system, and $\mathscr{T}$ is the set of all decentralized stabilizing controllers. To ensure the robustness of the closed-loop system, we assume that the system matrices (A, B_2) are unknown, but estimates $(\bar{A}, \bar{B}_2)$ are available to the control designer. Specifically, we consider

$$A = \bar{A} + \Delta_A, B_2 = \bar{B}_2 + \Delta_{B_2},$$

where the uncertainty matrices Δ_A and Δ_{B_2} are unknown and belong to the following convex compact set of the polytope type [37], represented as:

$$\mathcal{U} := \left\{(\Delta_A, \Delta_{B_2}) \mid (\Delta_A, \Delta_{B_2}) = \sum_{i=1}^{M} \xi_i(\Delta_A^i, \Delta_{B_2}^i), \sum_{i}^{M} \xi_i = 1,\ \xi_i \geq 0\right\}.$$

Here $\left\{\Delta_A^1, \ldots, \Delta_A^M\right\}$ and $\left\{\Delta_{B_2}^1, \ldots, \Delta_{B_2}^M\right\}$ are known vertex matrices with fixed $M > 0$.

2.2 An Approximated CP Model of the ODC Problem

For notational convenience, we define the data matrix Φ and parameter matrix W as follows:

$$\Phi := \begin{bmatrix} C^\top C & 0 \\ 0 & D^\top D \end{bmatrix} \in \mathbb{S}^p, \quad W := \begin{bmatrix} W_1 & W_2 \\ W_2^\top & W_3 \end{bmatrix} \in \mathbb{S}^p,$$

where $W_1 \in \mathbb{S}^n_{++}$, $W_2 \in \mathbb{R}^{n\times m}$, $W_3 \in \mathbb{S}^m$, $p = n + m$. The following lemma shows that under an appropriate convex restriction, the feedback gain matrix K derived from the parameter matrix W can preserve the decentralized structure, robust stability, and robust performance, even in the presence of parameter uncertainties.

Lemma 2.1 [30, Theorem 1] *For $i = 1, \ldots, M$, the function $\mathcal{F}_i : \mathbb{S}^p \to \mathbb{R}^{n\times n}$ is defined as*

$$\mathcal{F}_i(W) := F_i W E^\top + E W F_i^\top + B_1 B_1^\top,$$

where

$$E := \begin{bmatrix} I_{n\times n} & 0_{n\times m} \end{bmatrix} \in \mathbb{R}^{n\times p}, \quad F_i := \begin{bmatrix} A^i & -B_2^i \end{bmatrix} \in \mathbb{R}^{n\times p}.$$

Based on the partition of the decentralized structure of the feedback gain matrix K defined in (3), we define the set

$$\begin{aligned} \mathcal{W} := \Big\{ & W \in \mathbb{S}^p \mid W \succeq 0,\ \mathcal{F}_i(W) \preceq 0,\ \forall\, i = 1, 2, \ldots, M, \\ & W_1 = \mathbf{blkdiag}\left\{ W_{1,D_1},\ W_{1,D_2}, \ldots,\ W_{1,D_m} \right\}, \\ & W_2 = \mathbf{blkdiag}\left\{ W_{2,D_1},\ W_{2,D_2}, \ldots,\ W_{2,D_m} \right\}, \\ & W_{1,D_i} \in \mathbb{S}^{D_i},\ W_{2,D_i} \in \mathbb{R}^{D_i},\ \forall\, j = 1, 2, \ldots, m \Big\} \end{aligned} \tag{6}$$

and

$$\mathcal{K} := \left\{ W_2^\top W_1^{-1} \mid W \in \mathcal{W} \right\}.$$

Then,

(a) *$K \in \mathcal{K}$ maintains the decentralized structure as defined in (3).*
(b) *$K \in \mathcal{K}$ stabilizes the closed-loop system under parameter uncertainties.*
(c) *$K \in \mathcal{K}$ gives $\langle \Phi, W \rangle \geq \|H_i(s)\|_2^2$, $\forall\, i = 1, 2, \ldots, M$, where $\|H_i(s)\|_2$ represents the $\mathcal{H}_2$-norm with respect to i-th extreme system.*

On the one hand, we observe that $\langle \Phi, W \rangle$ serves as an upper bound for $\|H(s)\|_2^2$ according to Lemma 2.1. On the other hand, the block diagonal structure of W_1 and W_2 in (6) can be restricted by the following linear systems:

$$V_{j1} W V_{j2}^\top = 0, \quad j = 1, \ldots, N, \tag{7}$$

where $N = (3m(m-1))/2$, and the matrices $V_{j1} \in \mathbb{R}^{v_{j1}\times p}$ and $V_{j2} \in \mathbb{R}^{v_{j2}\times p}$ are provided in Appendix A. Therefore, by introducing slack variables $S_i \in \mathbb{S}^n_+$, $i = 1, \ldots, M$, the design of the controller K inherent in the $\mathcal{H}_2$-guaranteed cost decentralized control problem (5) can be relaxed into the following SDP problem:

$$\min_{W, S_i} \langle \Phi, W \rangle$$

$$\begin{aligned} \text{s.t. } & S_i + \mathcal{F}_i(W) = 0, \\ & V_{j1} W V_{j2}^\top \quad = 0, \quad j = 1, \ldots, N, \\ & W \in \mathbb{S}_+^p, \ S_i \in \mathbb{S}_+^n, i = 1, \ldots, M. \end{aligned} \tag{8}$$

To reduce the heavy cost of matrix multiplication and improve implementation performance, we consider rewriting the above problem in an equivalent vector form. For a symmetric matrix $S \in \mathbb{S}^n$, there exists a matrix $T_n \in \mathbb{R}^{\frac{n(n+1)}{2} \times n^2}$ such that

$$\mathbf{svec}(S) = T_n \mathbf{vec}(S) \quad \text{with} \quad T_n T_n^\top = I_{\frac{n(n+1)}{2}}, \tag{9}$$

where the symmetric vectorization operator $\mathbf{svec} : \mathbb{S}^n \to \mathbb{R}^{\frac{n(n+1)}{2}}$ is defined as:

$$\mathbf{svec}(S) := \left[s_{11}, \sqrt{2} s_{21}, \cdots, \sqrt{2} s_{n1}, \cdots, s_{nn} \right]^\top.$$

For ease of notation, we define the following quantities:

$$\begin{aligned} & w = \mathbf{svec}(W) \in \mathbb{R}^{\frac{p(p+1)}{2}}, \ r = \mathbf{svec}(\Phi) \in \mathbb{R}^{\frac{p(p+1)}{2}}, \quad b = \mathbf{svec}(B_1 B_1^\top) \in \mathbb{R}^{\frac{n(n+1)}{2}}, \\ & A_w^i = T_n(E \otimes E F_i + E F_i \otimes E) T_p^\top \in \mathbb{R}^{\frac{n(n+1)}{2} \times \frac{p(p+1)}{2}}, s_i = \mathbf{svec}(S_i) \in \mathbb{R}^{\frac{n(n+1)}{2}}, \\ & B_w^j = (V_{j2} \otimes V_{j1}) T_p^\top \in \mathbb{R}^{v_{j2} v_{j1} \times \frac{p(p+1)}{2}}, \text{ for } i = 1, \ldots, M, \ j = 1, \ldots, N, \end{aligned}$$

and

$$\begin{aligned} & s = \left[s_1 \ , \ \ldots \ , \ s_M \right]^\top \in \mathbb{R}^{M \frac{n(n+1)}{2}}, \quad A_w = \left[A_w^1 \ , \ \ldots \ , \ A_w^M \right]^\top \in \mathbb{R}^{M \frac{n(n+1)}{2} \times \frac{p(p+1)}{2}}, \\ & b_w = \left[b \ , \ \ldots \ , \ b \right]^\top \in \mathbb{R}^{M \frac{n(n+1)}{2}}, \quad B_w = \left[B_w^1 \ , \ \ldots \ , \ B_w^N \right]^\top \in \mathbb{R}^{\sum_{j=1}^N v_{j2} v_{j1} \times \frac{p(p+1)}{2}}. \end{aligned}$$

By combining the properties of the Kronecker product with the Eq. (9), we can reformulate the problem (8) into the following vectorized form:

$$\begin{aligned} \min_{w,s} \ & r^\top w \\ \text{s.t.} \ & A_w w + s + b_w = 0, \\ & B_w w = 0, \\ & w \in \Gamma^{\frac{p(p+1)}{2}}, s \in \mathcal{C}, \end{aligned} \tag{10}$$

where the cone $\Gamma^{\frac{n(n+1)}{2}} \subseteq \mathbb{R}^{\frac{n(n+1)}{2}}$ is defined as

$$\Gamma^{\frac{n(n+1)}{2}} := \left\{ \mathbf{svec}(X) \in \mathbb{R}^{\frac{n(n+1)}{2}} \mid X \in \mathbb{S}_+^n \right\},$$

and the convex set $\mathcal{C} \subseteq \mathbb{R}^{M \frac{n(n+1)}{2}}$ is denoted by $\mathcal{C} := \Gamma^{\frac{n(n+1)}{2}} \times \ldots \times \Gamma^{\frac{n(n+1)}{2}}$.

3 An Accelerated PADMM for Solving the CP Model

In this section, we first design the PADMM to solve the dual problem of the approximated CP model (10). We then establish the equivalence between the semi-proximal ADMM, including PADMM, and the (partial) PPA. This equivalence allows us to accelerate the PADMM by applying the Halpern iteration to the (partial) PPA.

3.1 PADMM for Solving the CP Model

The Lagrangian function of the problem (10) is defined by

$$\mathcal{L}(w_s; z, y, v_\Lambda) := -\langle z, b_w\rangle - \langle z + v, s\rangle + \langle r - \Lambda - A_w^\top z - B_w^\top y, w\rangle,$$

where the primal variable w_s, and multipliers z, y and v_Λ of the problem (10) are defined as

$$z := [z_1; \ldots; z_M] \in \mathcal{Z} := \mathbb{R}^{M\frac{n(n+1)}{2}}, \quad y := [y_1; \ldots; y_N] \in \mathcal{Y} := \mathbb{R}^{\sum_{j=1}^N v_{j2}v_{j1}},$$
$$v_\Lambda := [\Lambda; v] \in \mathcal{V} := \mathbb{R}^{\frac{p(p+1)}{2}} \times \mathcal{Z}, \qquad w_s := [w; s] \in \mathcal{V}.$$

The dual of (10), ignoring the minus sign in the objective, is given by

$$\begin{aligned} \min_{z,y,\Lambda,v} \ & \langle z, b_w\rangle \\ \text{s.t.} \ & A_w^\top z + B_w^\top y + \Lambda = r, \\ & z \qquad\quad + v = 0, \\ & \Lambda \in \Gamma^{\frac{p(p+1)}{2}}, v \in \mathcal{C}. \end{aligned} \tag{11}$$

To achieve a more compact representation of problem (11), we introduce the linear operators $\mathcal{A}$ and $\mathcal{B}$ as follows:

$$\mathcal{A} := \begin{bmatrix} A_w^1 & I & 0 & \cdots & 0 \\ A_w^2 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_w^M & 0 & 0 & \cdots & I \end{bmatrix} = \begin{bmatrix} A_w & \mathcal{I} \end{bmatrix}, \quad \mathcal{B} := \begin{bmatrix} B_w^1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ B_w^N & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} B_w & 0 \end{bmatrix}.$$

The adjoints of the linear operators $\mathcal{A}$ and $\mathcal{B}$ are denoted as

$$\mathcal{A}^* = \begin{bmatrix} A_w^\top \\ \mathcal{I} \end{bmatrix} \quad \text{and} \quad \mathcal{B}^* = \begin{bmatrix} B_w^\top \\ 0 \end{bmatrix}.$$

Let $\tilde{b} := [b_w; 0] \in \mathcal{Z} \times \mathcal{Y}$, $\tilde{r} := [r; 0] \in \mathcal{V}$, and $\tilde{\mathcal{A}}^* := \begin{bmatrix} \mathcal{A}^* & \mathcal{B}^* \end{bmatrix}$. Then, we can rewrite problem (11) as the following two-block problem:

$$\min_{\xi, v_\Lambda} \langle \tilde{b}, \xi\rangle + \delta_\Gamma(v_\Lambda)$$

$$\text{s.t. } \tilde{\mathcal{A}}^*\xi + v_\Lambda = \tilde{r}, \tag{12}$$

where $\Gamma = \Gamma^{\frac{p(p+1)}{2}} \times \mathcal{C}$ and $\xi = [z; y] \in \mathcal{Z} \times \mathcal{Y}$. Given $\sigma > 0$, for any $(v_\Lambda, \xi; w_s) \in \mathcal{U} := \mathcal{V} \times (\mathcal{Z} \times \mathcal{Y}) \times \mathcal{V}$, we can define the augmented Lagrangian function associated with the problem (12) as:

$$\mathcal{L}_\sigma(v_\Lambda, \xi; w_s) := \langle \tilde{b}, \xi \rangle + \delta_\Gamma(v_\Lambda) + \langle w_s, \tilde{\mathcal{A}}^*\xi + v_\Lambda - \tilde{r} \rangle + \frac{\sigma}{2}\|\tilde{\mathcal{A}}^*\xi + v_\Lambda - \tilde{r}\|^2.$$

The KKT system associated with (12) is given by:

$$\begin{aligned} 0 &= \tilde{b} + \tilde{\mathcal{A}} w_s, \\ 0 &\in \partial\delta_\Gamma(v_\Lambda) + w_s, \\ 0 &= \tilde{r} - \tilde{\mathcal{A}}^*\xi - v_\Lambda. \end{aligned}$$

Let Ω denote the solution set to the above KKT system. Assuming $\Omega \neq \emptyset$, we define the following monotone operator $\mathcal{T}$

$$\mathcal{T}(v_\Lambda, \xi; w_s) := \begin{pmatrix} \tilde{b} + \tilde{\mathcal{A}} w_s \\ \partial\delta_\Gamma(v_\Lambda) + w_s \\ \tilde{r} - \tilde{\mathcal{A}}^*\xi - v_\Lambda \end{pmatrix}, \quad \forall (v_\Lambda, \xi; w_s) \in \mathcal{U},$$

where $\partial\delta_\Gamma(\cdot)$ denotes the subdifferential of convex function $\delta_\Gamma(\cdot)$. It is evident that $\mathcal{T}$ is a maximally monotone operator [3, Example 20.26 and Corollary 25.5], and $\mathcal{T}^{-1}(0) = \Omega \neq \emptyset$.

Next, we focus on the design of the algorithm. To enhance computational efficiency, we categorize the problem into medium-scale problems (**Case 1**) and large-scale problems (**Case 2**), selecting distinct proximal terms accordingly.

Case 1. For medium-scale problems, let μ_0 and μ_1 be given positive parameters. Define

$$\tilde{\mathcal{S}}_0 := \sigma\mu_0 \begin{bmatrix} \mathcal{I}_\mathcal{Z} & 0 \\ 0 & \mathcal{I}_\mathcal{Y} \end{bmatrix},$$

where $\mathcal{I}_\mathcal{Z}$, $\mathcal{I}_\mathcal{Y}$ and $\mathcal{I}_\mathcal{V}$ denote the identity operators in $\mathcal{Z}$, $\mathcal{Y}$, and $\mathcal{V}$, respectively. The two-block PADMM for solving medium-scale problem (12) is provided as follows:

Case 2. For large-scale problems, let μ_2 and μ_3 be given positive parameters. Define

$$\mathcal{Q} := \sigma\tilde{\mathcal{A}}\tilde{\mathcal{A}}^* + \tilde{\mathcal{S}}_0' = \sigma \begin{bmatrix} \mathcal{A}\mathcal{A}^* + \mu_2\mathcal{I}_\mathcal{Z} & \mathcal{A}\mathcal{B}^* \\ \mathcal{B}\mathcal{A}^* & \mathcal{B}\mathcal{B}^* + \mu_3\mathcal{I}_\mathcal{Y} \end{bmatrix} = \mathcal{Q}_u + \mathcal{Q}_d + \mathcal{Q}_u^*,$$

where

$$\tilde{\mathcal{S}}_0' := \sigma \begin{bmatrix} \mu_2\mathcal{I}_\mathcal{Z} & 0 \\ 0 & \mu_3\mathcal{I}_\mathcal{Y} \end{bmatrix}, \quad \mathcal{Q}_u := \sigma \begin{bmatrix} 0 & \mathcal{A}\mathcal{B}^* \\ 0 & 0 \end{bmatrix},$$

Algorithm 1 PADMM_2blk

Require: Set $u^0 = (v_\Lambda^0, \xi^0, w_s^0) \in \mathcal{U}$ to be the initial point.
for $k = 0, 1, ...,$ **do**

$$
\begin{aligned}
v_\Lambda^{k+1} &= \underset{v_\Lambda \in \mathcal{V}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda, \xi^k; w_s^k) + \frac{1}{2}\|v_\Lambda - v_\Lambda^k\|^2_{\mu_1 \mathcal{I}_\mathcal{V}} \\
w_s^{k+1} &= w_s^k + \sigma(\tilde{\mathcal{A}}^* \xi^k + v_\Lambda^{k+1} - \tilde{r}) \\
\xi^{k+1} &= \underset{\xi \in \mathcal{Z} \times \mathcal{Y}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda^{k+1}, \xi; w_s^{k+1}) + \frac{1}{2}\|\xi - \xi^k\|^2_{\tilde{\mathcal{S}}_0}
\end{aligned}
$$

end for

and

$$
\mathcal{Q}_d := \sigma \begin{bmatrix} \mathcal{A}\mathcal{A}^* + \mu_2 \mathcal{I}_\mathcal{Z} & 0 \\ 0 & \mathcal{B}\mathcal{B}^* + \mu_3 \mathcal{I}_\mathcal{Y} \end{bmatrix}.
$$

Note that $\mathcal{Q}_d$ is positive definite. From [23, 24], the self-adjoint positive semi-definite linear operator $\mathrm{sGS}(\mathcal{Q}) : \mathcal{Z} \times \mathcal{Y} \to \mathcal{Z} \times \mathcal{Y}$ can be defined as

$$
\mathrm{sGS}(\mathcal{Q}) := \mathcal{Q}_u \mathcal{Q}_d^{-1} \mathcal{Q}_u^*. \tag{13}
$$

For large-scale problems, solving the linear system involving ξ poses a significant challenge. Utilizing the $\mathrm{sGS}(\mathcal{Q})$ operator enables the separation of ξ into z and y. Consequently, problem (12) can be solved using the multi-block PADMM provided in Algorithm 2:

Algorithm 2 PADMM_sGS

Require: Set $u^0 = (v_\Lambda^0, (z^0, y^0), w_s^0) \in \mathcal{U}$ to be the initial point.
for $k = 0, 1, ...,$ **do**

$$
\begin{aligned}
v_\Lambda^{k+1} &= \underset{v_\Lambda \in \mathcal{V}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda, (z^k, y^k); w_s^k) + \frac{1}{2}\|v_\Lambda - v_\Lambda^k\|^2_{\mu_1 \mathcal{I}_\mathcal{V}} \\
w_s^{k+1} &= w_s^k + \sigma(\tilde{\mathcal{A}}^*[z^k; y^k] + v_\Lambda^{k+1} - \tilde{r}) \\
y^{k+\frac{1}{2}} &= \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda^{k+1}, (z^k, y); w_s^{k+1}) + \frac{1}{2}\|y - y^k\|^2_{\sigma\mu_3 \mathcal{I}_\mathcal{Y}} \qquad (14) \\
z^{k+1} &= \underset{z \in \mathcal{Z}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda^{k+1}, (z, y^{k+\frac{1}{2}}); w_s^{k+1}) + \frac{1}{2}\|z - z^k\|^2_{\sigma\mu_2 \mathcal{I}_\mathcal{Z}} \qquad (15) \\
y^{k+1} &= \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda^{k+1}, (z^{k+1}, y); w_s^{k+1}) + \frac{1}{2}\|y - y^k\|^2_{\sigma\mu_3 \mathcal{I}_\mathcal{Y}} \qquad (16)
\end{aligned}
$$

end for

To simplify notation, we introduce the following proximal operators:

$$\mathcal{S}_1 = \mu_1 \mathcal{I}_{\mathcal{V}} \quad \text{and} \quad \mathcal{S}_2 := \begin{cases} \tilde{\mathcal{S}}_0, & \text{for PADMM\_2blk,} \\ \text{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0', & \text{for PADMM\_sGS.} \end{cases} \tag{17}$$

This allows us to consolidate Algorithm 1 and Algorithm 2 into the unified PADMM framework presented in Algorithm 3:

Algorithm 3 PADMM

Require: Set $u^0 = (v_\Lambda^0, \xi^0, w_s^0) \in \mathcal{U}$ to be the initial point.
for $k = 0, 1, \ldots,$ **do**

$$v_\Lambda^{k+1} = \underset{v_\Lambda \in \mathcal{V}}{\text{argmin}}\ \mathcal{L}_\sigma(v_\Lambda, \xi^k; w_s^k) + \frac{1}{2}\|v_\Lambda - v_\Lambda^k\|_{\mathcal{S}_1}^2 \tag{18}$$

$$w_s^{k+1} = w_s^k + \sigma(\tilde{\mathcal{A}}^* \xi^k + v_\Lambda^{k+1} - \tilde{r})$$

$$\xi^{k+1} = \underset{\xi \in \mathcal{Z} \times \mathcal{Y}}{\text{argmin}}\ \mathcal{L}_\sigma(v_\Lambda^{k+1}, \xi; w_s^{k+1}) + \frac{1}{2}\|\xi - \xi^k\|_{\mathcal{S}_2}^2 \tag{19}$$

end for

Remark 3.1 The selection of $\mathcal{S}_2$ is fundamentally dependent on the dimensionality of the problem and the available computational memory (RAM). In our numerical experiments, for example, the dual problem (12) and the primal problem (10) derived from $n = 300, m = 2, M = 4$ in the ODC problem (5) are classified as large-scale problems. For such cases, the proximal operator $\text{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$ is utilized within the PADMM framework. The large-scale subproblems associated with this proximal operator are efficiently solved by leveraging the sGS decomposition technique, as detailed in [23, 24].

3.2 Acceleration of the PADMM Scheme

We define a self-adjoint linear operator $\mathcal{S} : \mathcal{U} \to \mathcal{U}$ as follows:

$$\mathcal{S} := \begin{pmatrix} \mathcal{S}_1 & & \\ & \sigma \tilde{\mathcal{A}} \tilde{\mathcal{A}}^* + \mathcal{S}_2 & \tilde{\mathcal{A}} \\ & \tilde{\mathcal{A}}^* & \sigma^{-1} \mathcal{I}_{\mathcal{V}} \end{pmatrix}.$$

Since $\mathcal{S}_1$ and $\mathcal{S}_2$ are self-adjoint positive definite linear operators, by leveraging the equivalence between the semi-proximal ADMM and the (partial) proximal point method presented in Proposition B.1 in Appendix B, we can obtain the following equivalence:

Proposition 3.1 *Consider* $(v_\Lambda^0, \xi^0, w_s^0) \in \mathcal{U}$ *and* $\rho \in \mathbb{R}$. *Then the point* $(v_\Lambda^+, \xi^+, w_s^+)$ *generated by the following generalized PADMM (GPADMM) scheme*

$$\begin{aligned} \bar{v}_\Lambda^+ &= \underset{v_\Lambda \in \mathcal{V}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(v_\Lambda, \xi^0; w_s^0) + \frac{1}{2} \left\| v_\Lambda - v_\Lambda^0 \right\|_{\mathcal{S}_1}^2, \\ \bar{w}_s^+ &= w_s^0 + \sigma(\bar{v}_\Lambda^+ + \tilde{\mathcal{A}}^* \xi^0 - \tilde{r}), \\ \bar{\xi}^+ &= \underset{\xi \in \mathcal{Z} \times \mathcal{Y}}{\operatorname{argmin}} \, \mathcal{L}_\sigma(\bar{v}_\Lambda^+, \xi; \bar{w}_s^+) + \frac{1}{2} \left\| \xi - \xi^0 \right\|_{\mathcal{S}_2}^2, \\ (v_\Lambda^+, \xi^+, w_s^+) &= (1-\rho)(v_\Lambda^0, \xi^0, w_s^0) + \rho(\bar{v}_\Lambda^+, \bar{\xi}^+, \bar{w}_s^+) \end{aligned} \tag{20}$$

is equivalent to the one generated by the following proximal point scheme

$$\begin{aligned} 0 &\in \mathcal{T}(\bar{v}_\Lambda^+, \bar{\xi}^+, \bar{w}_s^+) + \mathcal{S}(\bar{v}_\Lambda^+ - v_\Lambda^0, \bar{\xi}^+ - \xi^0, \bar{w}_s^+ - w_s^0), \\ (v_\Lambda^+, \xi^+, w_s^+) &= (1-\rho)(v_\Lambda^0, \xi^0, w_s^0) + \rho(\bar{v}_\Lambda^+, \bar{\xi}^+, \bar{w}_s^+). \end{aligned} \tag{21}$$

Remark 3.2 For the special positive definite diagonal proximal terms $\mathcal{S}_1 = \mu_1 \mathcal{I}_\mathcal{V}$ and $\mathcal{S}_2 = \tilde{\mathcal{S}}_0$, Eckstein [11] demonstrated that the scheme (20) with $\rho = 1$ and a cyclically equivalent updating rule "$v_\Lambda \to \xi \to w$" can be interpreted as a Douglas-Rachford splitting method for solving the inclusion system $0 \in \mathcal{T}$. Consequently, according to [10, Theorem 6], this approach can be then viewed as an application of the proximal point algorithm corresponding to an inclusion system depending on the specific splitting form of $\mathcal{T}$. Here, we focus on the general scheme (20) by providing a simple yet rigorous proof of the equivalence between GPADMM and the proximal point algorithm.

To achieve a faster convergence rate, based on the above Proposition 3.1, we adopt the Halpern fixed-point method to accelerate the proximal point scheme (21). Let ρ be a given relaxation parameter in $(0, 2]$ and $u := (v_\Lambda, \xi, w_s) \in \mathcal{U}$. Then each iteration in the relaxed PPA (21) can be expressed abstractly as:

$$u^+ = F_\rho(u),$$

where the operator $F_\rho : \mathcal{U} \to \mathcal{U}$ is defined by

$$F_\rho(u) := (1-\rho)u + \rho(\mathcal{S} + \mathcal{T})^{-1} \mathcal{S} u = u - 2 \cdot \frac{\rho}{2} \cdot [\mathcal{I} - (\mathcal{S} + \mathcal{T})^{-1} \mathcal{S}] u. \tag{22}$$

Note that

$$u \in \mathrm{Fix}(F_\rho) \;\Leftrightarrow\; u = (\mathcal{S} + \mathcal{T})^{-1} \mathcal{S} u \;\Leftrightarrow\; 0 \in \mathcal{T}(u).$$

Thus,

$$\mathrm{Fix}(F_\rho) = \mathcal{T}^{-1}(0).$$

For notational convenience, the dependence of F on ρ is omitted in the remainder of this section.

Since $\mathcal{S}$ is a symmetric positive definite linear operator, $\mathcal{S}^{-1}$ can be decomposed as $\mathcal{S}^{-1} = LL^T$, where L is a real lower triangular matrix with positive diagonal entries. We now demonstrate that F is non-expansive with respect to the induced norm $\|\cdot\|_{\mathcal{S}}$. To this end, we define the normalized operator of F, denoted by $\widetilde{F} : \mathcal{U} \to \mathcal{U}$, as follows:

$$\widetilde{F}(\tilde{u}) := (1-\rho)\tilde{u} + \rho\left(\mathcal{I} + L^T\mathcal{T}L\right)^{-1}\tilde{u} = \tilde{u} - 2\cdot\frac{\rho}{2}\cdot\left[\mathcal{I} - (\mathcal{I}+\widetilde{\mathcal{T}})^{-1}\right]\tilde{u}, \quad (23)$$

where $\widetilde{\mathcal{T}} = L^T\mathcal{T}L$ is a maximally monotone operator by [3, Proposition 23.25]. Similar to [25, Lemma 2.1], it is not difficult to verify that

$$L^{-1}F(u) = \widetilde{F}(L^{-1}u), \quad \forall\, u \in \mathcal{U}. \quad (24)$$

Proposition 3.2 *Given the parameter $\rho \in (0, 2]$, the operator $\widetilde{F}$ defined in (23) is non-expansive. Furthermore, F is non-expansive with respect to the induced norm $\|\cdot\|_{\mathcal{S}}$, that is,*

$$\|F(u) - F(v)\|_{\mathcal{S}} \le \|u - v\|_{\mathcal{S}}, \quad \forall\, u, v \in \mathcal{U}.$$

Proof Define the resolvent operator of $\widetilde{\mathcal{T}}$ as $\widetilde{G} := (\mathcal{I}+\widetilde{\mathcal{T}})^{-1}$ and $\widetilde{\mathcal{Q}} := \mathcal{I} - \widetilde{G}$. It should be noted that $\widetilde{G}$ is a single-valued and firmly non-expansive operator. By [3, Proposition 4.4] and $\rho \in (0, 2]$, we have

$$\begin{aligned}\langle\widetilde{\mathcal{Q}}x - \widetilde{\mathcal{Q}}y, x - y\rangle &= \langle\widetilde{\mathcal{Q}}x - \widetilde{\mathcal{Q}}y, \widetilde{G}x - \widetilde{G}y\rangle + \|\widetilde{\mathcal{Q}}x - \widetilde{\mathcal{Q}}y\|^2 \\ &\ge \|\widetilde{\mathcal{Q}}x - \widetilde{\mathcal{Q}}y\|^2 \ge \frac{\rho}{2}\|\widetilde{\mathcal{Q}}x - \widetilde{\mathcal{Q}}y\|^2,\end{aligned}$$

which shows that $\widetilde{\mathcal{Q}}$ is $\frac{\rho}{2}$-co-coercive. Then, according to [3, Proposition 4.11], we know that the mapping $\widetilde{\widetilde{F}} := \mathcal{I} - 2(\frac{\rho}{2})\widetilde{\mathcal{Q}}$ is a non-expansive operator.

Now for all $u, v \in \mathcal{U}$, it holds from (24) that

$$\begin{aligned}\|F(u) - F(v)\|_{\mathcal{S}} &= \|L^{-1}F(u) - L^{-1}F(v)\| \\ &= \|\widetilde{F}(L^{-1}u) - \widetilde{F}(L^{-1}v)\| \\ &\le \|L^{-1}u - L^{-1}v\| = \|u - v\|_{\mathcal{S}},\end{aligned}$$

where the last inequality follows from the non-expansiveness of $\widetilde{F}$. □

Proposition 3.2 implies that the following Halpern fixed-point iteration can be used to find a fixed-point in $\mathrm{Fix}(F)$:

$$u^{k+1} = \frac{1}{k+2}u^0 + \left(1 - \frac{1}{k+2}\right)F(u^k), \quad \forall\, k \ge 0. \quad (25)$$

Moreover, by considering the definition of F in (22) along with Proposition 3.1, we can equivalently reformulate the update scheme (25) as the following accelerated PADMM algorithm.

Algorithm 4 the accelerated proximal ADMM (APADMM)

Require: Choose parameters $\sigma > 0$, $\mu_0 > 0$, $\mu_1 > 0$, $\mu_2 > 0$, $\mu_3 > 0$, and $\rho \in (0, 2]$. Let operators $\mathcal{S}_1$ and $\mathcal{S}_2$ be defined in (17). Select an initial point $u^0 = (v_\Lambda^0, \xi^0, w_s^0) \in \mathcal{U}$.

for $k = 0, 1, ...,$ **do**

$$\bar{v}_\Lambda^k = \underset{v_\Lambda \in \mathcal{V}}{\operatorname{argmin}} \mathcal{L}_\sigma(v_\Lambda, \xi^k; w_s^k) + \frac{1}{2}\|v_\Lambda - v_\Lambda^k\|_{\mathcal{S}_1}^2 \tag{26}$$

$$\bar{w}_s^k = w_s^k + \sigma(\tilde{\mathcal{A}}^*\xi^k + \bar{v}_\Lambda^k - \tilde{r})$$

$$\bar{\xi}^k = \underset{\xi \in \mathcal{Z}\times\mathcal{Y}}{\operatorname{argmin}} \mathcal{L}_\sigma(\bar{v}_\Lambda^k, \xi; \bar{w}_s^k) + \frac{1}{2}\|\xi - \xi^k\|_{\mathcal{S}_2}^2 \tag{27}$$

$$\hat{u}^{k+1} = \rho\bar{u}^k + (1-\rho)u^k$$

$$u^{k+1} = \frac{1}{k+2}u^0 + \frac{k+1}{k+2}\hat{u}^{k+1}$$

end for

Remark 3.3 Algorithm 4 incorporates both the relaxation and the acceleration techniques. Notably, the relaxation parameter ρ is permissible within the interval $(0, 2]$, which is a distinguishing feature compared to the classic generalized ADMM [10], where ρ is confined to the open interval $(0, 2)$. Moreover, to enhance the performance of Algorithm 4, our implementation incorporates the restarting strategy discussed in [33, Section 11.4] and [35, Section 5.1].

We now present the convergence and rate of convergence results for Algorithm 4.

Theorem 3.1 [46, Theorem 2] *Assume that* $\mathcal{T}^{-1}(0) \neq \emptyset$. *Let* $\{u^k\}_{k=0}^\infty$ *denote the infinite sequence generated by Algorithm 4. Then, it holds that* $\lim_{k\to\infty} u^k = \Pi_{\mathcal{T}^{-1}(0)}(u^0)$.

Utilizing the results from [27, Theorem 2.1], we establish the following on the rate of convergence for Algorithm 4.

Theorem 3.2 *Assume that* $\mathcal{T}^{-1}(0) \neq \emptyset$. *Let* $\{u^k\}_{k=0}^\infty$ *be the infinite sequence generated by Algorithm 4. Then, the following inequality holds*

$$\|u^k - F(u^k)\|_{\mathcal{S}} \leq \frac{2\|u^0 - u^*\|_{\mathcal{S}}}{k+1}, \ \forall\, k \geq 0 \text{ and } u^* \in \mathcal{T}^{-1}(0).$$

4 A Fast Implementation for Solving the Subproblems of APADMM

In this section, we delineate a fast implementation strategy for solving subproblems of the APADMM for solving problem (12). This strategy involves varying proximal operators $\mathcal{S}_2$ depending on the problem scale.

4.1 The Projection Onto the Cone Γ

Within the subproblem (26), the variable $\bar{\Lambda}^k$ is determined through the following projection:

$$\bar{\Lambda}^k = \Pi_{\Gamma^{\frac{p(p+1)}{2}}} \left[\frac{\mu_1 \Lambda^k - \sigma(A_w^\top z^k + B_w^\top y^k - r + \frac{1}{\sigma} w^k)}{\sigma + \mu_1} \right], \tag{28}$$

and the variable $\bar{v}^k = [\bar{v}_1^k; \ldots; \bar{v}_M^k]$ is computed as follows: for $i \geq 1$,

$$\bar{v}_i^k = \Pi_{\Gamma^{\frac{n(n+1)}{2}}} \left[\frac{\mu_1 v_i^k - (\sigma z_i^k + s_i^k)}{\sigma + \mu_1} \right]. \tag{29}$$

4.2 A Lifting Technique for Solving Large-Scale Sparse Linear Systems

To enhance the algorithm's efficiency, we introduce a lifting technique to solve large-scale sparse linear systems within the subproblem (27), which is applicable to both **Case 1** and **Case 2** as previously discussed.

Case 1. In the APADMM iteration scheme with the operator $\mathcal{S}_2 = \tilde{\mathcal{S}}_0$, the optimality condition for $\bar{\xi}^k$ in (27) is expressed as:

$$\left(\tilde{\mathcal{A}}\tilde{\mathcal{A}}^* + \frac{1}{\sigma}\tilde{\mathcal{S}}_0 \right) \bar{\xi}^k = \frac{1}{\sigma}\tilde{\mathcal{S}}_0 \xi^k - \frac{1}{\sigma}\left(\tilde{b} + \tilde{\mathcal{A}} w_s^k \right) - \tilde{\mathcal{A}}(v_\Lambda^k - \tilde{r}). \tag{30}$$

Note that the computation of $\tilde{\mathcal{A}}\tilde{\mathcal{A}}^*$ is time-consuming, and even if $\tilde{\mathcal{A}}$ is sparse, $\tilde{\mathcal{A}}\tilde{\mathcal{A}}^*$ becomes dense for problem (12). Therefore, solving the linear system (30) via direct methods becomes formidable. To mitigate this challenge, we introduce an auxiliary variable $\eta' = \tilde{\mathcal{A}}^* \bar{\xi}^k$ leading to the following augmented system

$$\begin{bmatrix} \frac{1}{\sigma}\tilde{\mathcal{S}}_0 & \tilde{\mathcal{A}} \\ \tilde{\mathcal{A}}^* & -\mathcal{I} \end{bmatrix} \begin{bmatrix} \bar{\xi}^k \\ \eta' \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma}\tilde{\mathcal{S}}_0 \xi^k - \frac{1}{\sigma}\left(\tilde{b} + \tilde{\mathcal{A}} w_s^k \right) - \tilde{\mathcal{A}}(v_\Lambda^k - \tilde{r}) \\ 0 \end{bmatrix}. \tag{31}$$

This lifting technique effectively circumvents the need to compute $\tilde{\mathcal{A}}\tilde{\mathcal{A}}^*$ and the matrix-vector product $\tilde{\mathcal{A}}^* \bar{\xi}^k$ in the update of $\bar{\Lambda}^k$.

Case 2. For large-scale problems, substituting the proximal operator $\mathcal{S}_2 = \text{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$ into the subproblem (27) reveals that $\bar{\xi}^k = [\bar{z}^k; \bar{y}^k]$ in Algorithm 4 is analogous to $\xi^{k+1} = [z^{k+1}; y^{k+1}]$ in Algorithm 2. Hence, $\bar{\xi}^k$ can be obtained by solving (14)–(16) in Algorithm 2 using the sGS decomposition approach. In particular, from the Eq. (14), $y^{k+1/2}$ is computed by solving

$$y^{k+1/2} = \underset{y \in \mathcal{Y}}{\text{argmin}} \left\{ \frac{\sigma}{2} \| B_w^\top y + H_1^b \|^2 + \frac{1}{2} \| y - y^k \|^2_{\sigma\mu_3 \mathcal{I}_\mathcal{Y}} \right\}.$$

The optimality condition for this problem implies that $y^{k+1/2}$ is a solution to the following linear equation

$$(\mu_3 \mathcal{I}_{\mathcal{Y}} + B_w B_w^\top) y = \mu_3 y^k - B_w H_1^b,$$

where $H_1^b = -r + \frac{1}{\sigma} w^{k+1} + \Lambda^{k+1} + A_w^\top z^k$. Moreover, the update for z^{k+1} in (15) is explicitly given by

$$z^{k+1} = \underset{z \in \mathcal{Z}}{\operatorname{argmin}} \left\{ \langle z, b_w \rangle + \frac{\sigma}{2} \| z + \frac{1}{\sigma} s^{k+1} + v^{k+1} \|^2 + \frac{\sigma}{2} \| A_w^\top z + H_2^b \|^2 + \frac{1}{2} \| z - z^k \|^2_{\sigma \mu_2 \mathcal{I}_{\mathcal{Z}}} \right\}.$$

From its optimality condition, we know that

$$\left[(\mu_2 + 1) \mathcal{I}_{\mathcal{Z}} + A_w A_w^\top \right] z^{k+1} = \mu_2 z^k - \left[v^{k+1} + \frac{1}{\sigma} (b_w + s^{k+1}) + A_w H_2^b \right].$$

To update z^{k+1}, we can employ a similar lifting technique as used in (30) to solve the following linear system:

$$\begin{bmatrix} (\mu_2 + 1) \mathcal{I}_{\mathcal{Z}} & A_w \\ A_w^\top & -\mathcal{I} \end{bmatrix} \begin{bmatrix} z^{k+1} \\ z' \end{bmatrix} = \begin{bmatrix} \mu_2 z^k - (v^k + \frac{1}{\sigma}(b_w + s^k) + A_w H_2^b) \\ 0 \end{bmatrix}, \quad (32)$$

where $H_2^b = -r + \frac{1}{\sigma} w^{k+1} + \Lambda^{k+1} + B_w^\top y^{k+1/2}$. Finally, we obtain y^{k+1} similarly to $y^{k+1/2}$ as follows:

$$(\mu_3 \mathcal{I}_{\mathcal{Y}} + B_w B_w^\top) y^{k+1} = \mu_3 y^k - B_w H_1^f,$$

where $H_1^f = -r + \frac{1}{\sigma} w^{k+1} + \Lambda^{k+1} + A_w^\top z^{k+1}$.

5 Numerical Experiments

In this section, we present the numerical performance of APADMM in solving ODC problems for linear time-invariant systems. The APADMM implementation was conducted in C language (using the MSVC compiler) and executed on a workstation equipped with an Intel(R) Core(TM) i9-10900 CPU@2.80GHz and 64GB RAM.

5.1 Implementation Details

For the ODC problem data (1), matrices A and B_2 were generated following a normal distribution with zero mean and unit variance. The matrices C and D were obtained by generating a random orthogonal matrix to satisfy $C^\top D = 0$. Additionally, the

matrix B_1 was generated to satisfy $\mathcal{F}_i(W) \preceq 0$ for $i = 1, \ldots, M$ with a $p \times p$ random positive semidefinite matrix W.

After generating the data, we conducted a performance comparison between APADMM and several other methods, including GPADMM in (20), sGS_PADMM [23], and widely-used solvers such as SCS (C source code version 3.2.3), COSMO (Python interface version 0.8.8 for Julia language), and MOSEK (Python interface). The sGS_PADMM framework is detailed as follows:

Algorithm 5 sGS_PADMM

Require: Let $\sigma > 0$, $\mu_1 > 0$, $\mu_2 > 0$, $\mu_3 > 0$, and $\tau \in (0, (1+\sqrt{5})/2)$ be given parameters. Set $u^0 = (v_\Lambda^0, \xi^0, w_s^0) \in \mathcal{U}$ to be the initial point.
for $k = 0, 1, \ldots,$ **do**

$$\xi^{k+1} = \underset{\xi \in \mathcal{Z}\times\mathcal{Y}}{\operatorname{argmin}} \mathcal{L}_\sigma(v_\Lambda^k, \xi, w_s^k) + \frac{1}{2}\|\xi - \xi^k\|^2_{\mathrm{sGS}(\mathcal{Q})+\tilde{\mathcal{S}}_0'}$$

$$v_\Lambda^{k+1} = \underset{v_\Lambda \in \mathcal{V}}{\operatorname{argmin}} \mathcal{L}_\sigma(v_\Lambda, \xi^k, w_s^k) + \frac{1}{2}\|v_\Lambda - v_\Lambda^k\|^2_{\mu_1\mathcal{I}_\mathcal{V}}$$

$$w_s^{k+1} = w_s^k + \sigma\tau(\tilde{\mathcal{A}}^*\xi^{k+1} + v_\Lambda^{k+1} - \tilde{r})$$

end for

In the numerical experiments, SCS, COSMO, and MOSEK were employed using their default settings (as outlined in the respective manuals: COSMO [14], MOSEK [31], and SCS [36]). For the APADMM, GPADMM, and sGS_PADMM, we utilized the $\mathtt{LDL}^\top$ factorization [8] or the `mkl-pardiso` function[2] to solve the linear systems from Eqs. (31) or (32). The updates of Λ in (28) and v in (29) were computed using the eigenvalue decomposition implemented in `MKL` for the projection calculations. Regarding parameter settings, we choose $\mu_0 = \mu_1 = \mu_2 = \mu_3 = 1e\text{-}4$ for all three methods, $\rho = 2$ for APADMM, $\rho = 1.8$ for GPADMM, and $\tau = 1.618$ for sGS_PADMM. Additionally, we adopted the penalty parameter σ adjustment strategy from [19]. APADMM, GPADMM, and sGS_PADMM utilized the following stopping criterion based on the KKT relative residuals, similar to the ones used in SCS.

$$\text{Err\_rel} := \max\left\{\text{p\_res}, \text{d\_res}, \eta_{\text{gap}}\right\},$$

where $\text{d\_res} := \max\{\eta_s^k, \eta_{eq}^k\}$ with

$$\eta_s^k = \frac{\|v^k + z^k\|_\infty}{1 + \max\left\{\|v^k\|_\infty, \|z^k\|_\infty\right\}},$$

$$\eta_{eq}^k = \frac{\|A_w^\top z^k + B_w^\top y^k + \Lambda^k - r\|_\infty}{1 + \max\left\{\|A_w^\top z^k\|_\infty, \|B_w^\top y^k\|_\infty, \|\Lambda^k\|_\infty, \|r\|_\infty\right\}},$$

[2] https://www.intel.com/content/www/us/en/docs/onemkl/get-started-guide/2023-0/overview.html.

and $\text{p\_res} := \max\left\{\eta_z^k, \eta_\Lambda^k, \eta_y^k, \max_{i=1,\ldots,M}\{\eta_{v_i}^k\}\right\}$ with

$$\eta_\Lambda^k = \frac{\|\Lambda^k - \Pi_{\Gamma^{\frac{p(p+1)}{2}}}(\Lambda^k - w^k)\|_\infty}{1 + \max\left\{\|\Lambda^k\|_\infty, \|w^k\|_\infty\right\}}, \qquad \eta_y^k = \frac{\|B_w w^k\|_\infty}{1 + \|B_w w^k\|_\infty},$$

$$\eta_z^k = \frac{\|b_w + A_w w^k + s^k\|_\infty}{1 + \max\left\{\|b_w\|_\infty, \|s^k\|_\infty, \|A_w w^k\|_\infty\right\}}, \quad \eta_{v_i}^k = \frac{\|v_i^k - \Pi_{\Gamma^{\frac{n(n+1)}{2}}}(v_i^k - s_i^k)\|_\infty}{1 + \max\left\{\|v_i^k\|_\infty, \|s_i^k\|_\infty\right\}}.$$

Finally, the relative gap η_{gap} is given by

$$\eta_{\text{gap}} := \frac{|\text{p\_obj} - \text{d\_obj}|}{1 + \max\{|\text{p\_obj}|, |\text{d\_obj}|\}},$$

where

$$\text{p\_obj} := \langle r, w^k\rangle, \quad \text{d\_obj} := -\langle z^k, b_w\rangle.$$

5.2 Verification of Decentralized Structure

In this subsection, we validate that the feedback gain matrix K constructed from the solution obtained by Algorithm 4, adheres to the decentralized structure specified in (3) using the chemical reactor system from [17] as a case study. Additionally, we assess the stability of the control system. For this experiment, the estimated matrices (A, B_2) and the given matrices (B_1, C, D) in the LTI system (1) are defined as follows:

$$A = \begin{bmatrix} -1.38 & -0.2077 & 6.715 & -5.676 \\ -0.5814 & -4.29 & 0 & 0.675 \\ 1.067 & 4.273 & -6.654 & 5.893 \\ -0.048 & -4.273 & 1.343 & -2.104 \end{bmatrix}, \; B_1 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0, & 0 \\ 5.679 & 0 \\ 1.136 & -3.146 \\ 1.136 & 0 \end{bmatrix}, \; C = \begin{bmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}, \; D = \begin{bmatrix} 0\,0 \\ 0\,0 \\ 1\,0 \\ 0\,1 \end{bmatrix}.$$

To introduce robustness, we incorporate noise into the system matrix A with a magnitude of $\pm 5\%$ of its nominal values and set $M = 4$ as the number of extreme systems. The solution W obtained by APADMM, with a relative error $\text{Err\_rel} \leq 1e{-}7$, is given by

$$W = \left[\begin{array}{cccc|cc} 1.94008 & -0.15198 & 0 & 0 & 0.05891 & 0 \\ -0.15198 & 0.09056 & 0 & 0 & 0.05654 & 0 \\ 0 & 0 & 0.30031 & 0.26107 & 0 & -0.17958 \\ 0 & 0 & 0.26107 & 0.47226 & 0 & -0.23248 \\ \hline 0.05891 & 0.05654 & 0 & 0 & 0.04935 & 0 \\ 0 & 0 & -0.17958 & -0.23248 & 0 & 0.13116 \end{array}\right].$$

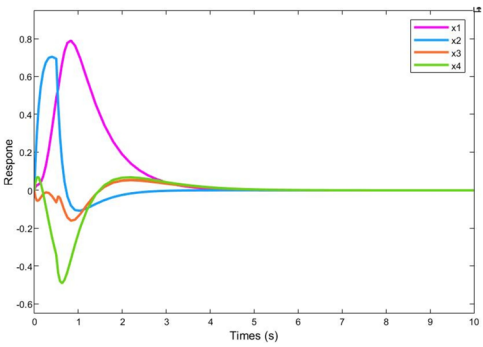


Fig. 1 System response under the feedback matrix K

The feedback gain matrix K, computed as $K = W_2^\top W_1^{-1}$, is

$$K = W_2^\top W_1^{-1} = \left[\begin{array}{cc|cc} 0.09128 & 0.77760 & 0 & 0 \\ 0 & 0 & -0.32737 & -0.31129 \end{array}\right], \tag{33}$$

which satisfies the decentralized structure requirements. In the simulation, the disturbance $w(t)$ is modeled as an impulse vector. The system responses of all state variables are depicted in Fig. 1.
The system response depicted in Fig. 1 confirms that the LTI system successfully maintains robust stability.

5.3 Numerical Results for the Lifting Technique and Acceleration

In this subsection, we demonstrate the effectiveness of the lifting technique for solving large-scale sparse linear systems, as opposed to directly solving the normal Eq. (30). Specifically, Table 1 presents the numerical results of the lifting technique, where PADMM_AA$^\top$ refers to the result obtained by solving the normal Eq. (30) using LDL$^\top$ decomposition directly, and PADMM_Lifting denotes the results obtained by using the lifting technique.

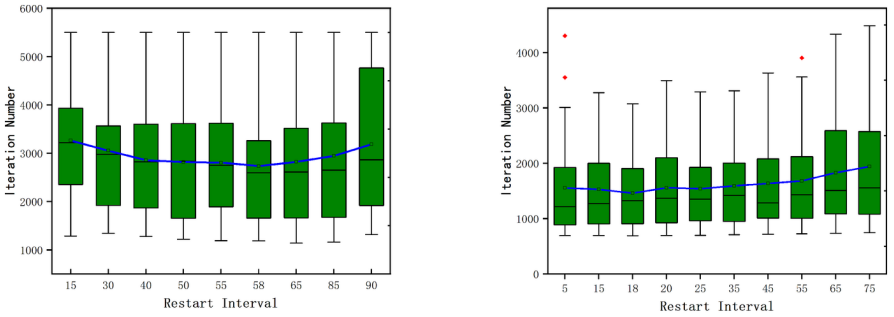
From Table 1, it is evident that the lifting technique is more than ten times faster than directly solving the normal Eq. (30). The acceleration becomes increasingly pronounced as the problem size scales up. This improvement is likely due to the fact that $\tilde{\mathcal{A}}\tilde{\mathcal{A}}^*$ results in a dense matrix, even if $\tilde{\mathcal{A}}$ is sparse. The complexity of the LDL$^\top$ decomposition is closely tied to the matrix's sparsity structure, as discussed in [9, Chapter 4]. Conversely, the lifting technique can avoid calculating $\tilde{\mathcal{A}}\tilde{\mathcal{A}}^*$ and fully exploit the sparsity of $\tilde{\mathcal{A}}$.

Numerical testing also indicates that the restart strategy proposed [33, Section 11.4] and [35, Section 5.1] significantly enhances APADMM's performance. To determine an appropriate fixed iteration number for restarting the APADMM algorithm, we conducted experiments on 30 randomly generated examples, testing various restart intervals to solve problem (12). As depicted in Fig. 2, setting the restart interval to 58 for APADMM with the operator sGS($\mathcal{Q}$) + $\tilde{\mathcal{S}}_0'$ considerably reduces both average

Table 1 Performance comparison between PADMM_AA$^\top$ and PADMM_Lifting with Err_rel=1e−5

Algorithm	n, m, M	Iter	Err_rel	Time (s)	Lin_sys time (s)[1]
test 1					
PADMM_AA$^\top$	(10, 5, 11)	993	4.994e−06	3.54e−01	1.59e−01
PADMM_Lifting	(10, 5, 11)	941	8.273e−06	1.87e−01	1.59e−02
test 2					
PADMM_AA$^\top$	(30, 15, 11)	1112	9.698e−06	5.22e+01	2.94e+01
PADMM_Lifting	(30, 15, 11)	1353	2.521e−06	4.55e+00	1.44e+00
test 3					
PADMM_AA$^\top$	(50, 25, 11)	1202	2.674e−06	6.54e+02	2.42e+02
PADMM_Lifting	(50, 25, 11)	1151	5.278e−06	2.13e+01	9.55e+00

[1]Lin_sys Time refers to the total time spent on solving the linear system



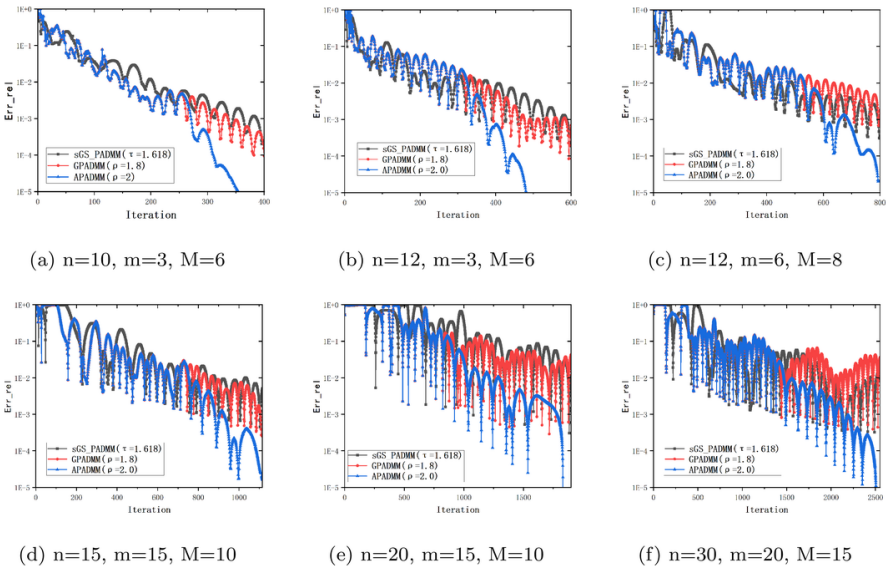
(a) APADMM with operator $\mathrm{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$ (b) APADMM with operator $\tilde{\mathcal{S}}_0$

Fig. 2 Performance of APADMM with various restart intervals

and median iteration numbers. In contrast, for APADMM using the operator $\tilde{\mathcal{S}}_0'$, the optimal restart interval is found to be 18.

After setting the restart interval, we compared APADMM against traditional PADMM-type methods, including GPADMM and sGS_PADMM. Both APADMM and GPADMM utilized the proximal operator $\mathrm{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$. As illustrated in Fig. 3, the APADMM outperforms both GPADMM and sGS_PADMM, achieving a better solution in terms of relative error Err_rel with fewer iterations across various problem scales. Additionally, GPADMM demonstrated performance comparable to sGS_PADMM.

To further analyze the impact of different proximal operators on the acceleration, we summarized the numerical results of APADMM and PADMM in Table 2, with specific experimental results for each example provided in Table 3. Here, “TB" denotes the proximal operator $\tilde{\mathcal{S}}_0$, and “sGS" refers to the proximal operator $\mathrm{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$. As shown in Table 2, sGS_APADMM exhibits an average iteration step improvement of **46.268**% and reduces the average time by **47.378**% compared to sGS_GPADMM. Likewise, TB_APADMM demonstrates an average iteration step improvement of



(a) n=10, m=3, M=6 (b) n=12, m=3, M=6 (c) n=12, m=6, M=8

(d) n=15, m=15, M=10 (e) n=20, m=15, M=10 (f) n=30, m=20, M=15

Fig. 3 Performance comparison of APADMM, sGS_PADMM, and GPADMM

25.668% and a time reduction of **26.809**% compared to TB_GPADMM. These findings highlight that the acceleration technique significantly boosts the efficiency of PADMM with a more pronounced impact when using the proximal operator $\mathrm{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$. One possible explanation is that the accelerated effect intensifies with larger iteration numbers, and PADMM with $\mathrm{sGS}(\mathcal{Q}) + \tilde{\mathcal{S}}_0'$ generally requires more iterations.

5.4 Numerical Results for APADMM in Comparison with COSMO, MOSEK, and SCS

In this subsection, we compare APADMM with COSMO, MOSEK (an interior point method), and SCS in solving the ODC problems of various scales. All ADMM-type solvers were terminated with a tolerance of 1e−4. For the following tests, the maximum number of iterations was set to 25000 for APADMM, SCS, and COSMO for small and medium-scale problems, and to 10000 for large-scale problems. The results are summarized in Tables 4, 5, and 6, where the status `max_iter` indicates the algorithm stopping due to reaching the maximum iteration steps, `solved` means that the solver has successfully solved the ODC problems within the given accuracy, and `memory error` denotes that the solver exceeded memory limits, resulting in an error. Additionally, "obj_bias" represents the absolute difference between the objective function values obtained by each algorithm and that of MOSEK. Furthermore, "p_abs" and "d_abs" denote the absolute residual errors, while "p_res" and "d_res" represent

Table 2 Comparison of the acceleration effects on proximal operators $\tilde{\mathcal{S}}_0$ and $\mathrm{sGS}(\mathcal{Q})+\tilde{\mathcal{S}}_0'$

Method	Iteration number				CPU times (s)			
	Avgs	Med.	Max.	Min.	Avgs	Med.	Max.	Min.
TB_GPADMM	1706.26	1503	4193	617	2.35e+02	5.91e+01	3.85e+03	2.44e−01
TB_APADMM	1268.30	1197	2604	572	1.72e+02	3.66e+01	2.38e+03	9.67e−02
sGS_GPADMM	5599.54	5152	13628	2201	5.34e+02	1.30e+02	7.86e+03	5.20e−01
sGS_APADMM	3008.76	3010	6210	1298	2.81e+02	8.92e+01	3.65e+03	2.40e−01

Table 3 Comparison of the acceleration performance of APADMM with different proximal operators for the ODC problems with Err_rel=1e−5

n	m	M	TB_GPADMM			TB_APADMM			sGS_GPADMM			sGS_APADMM		
			Iter	Err_rel	Time (s)	Iter	Err_rel	Time (s)	Iter	Err_rel	Time (s)	Iter	Err_rel	Time (s)
10	10	5	2674	9.807e−06	4.40e−01	692	9.827e−06	9.67e−02	2837	9.110e−06	5.20e−01	1320	8.900e−06	2.40e−01
10	5	10	1167	4.019e−06	2.44e−01	900	9.936e−06	1.86e−01	4245	9.368e−06	8.66e−01	1298	9.948e−06	2.45e−01
15	10	10	1639	4.267e−06	6.52e−01	704	7.701e−06	3.07e−01	4654	7.529e−06	2.27e+00	3080	9.879e−06	1.45e+00
15	15	10	948	5.776e−06	4.05e−01	1422	8.151e−06	6.41e−01	2231	9.488e−06	1.19e+00	1927	9.940e−06	9.82e−01
20	10	10	2489	9.500e−06	1.51e+00	887	7.659e−06	5.23e−01	4832	9.440e−06	3.15e+00	1416	8.511e−06	9.06e−01
20	15	10	980	9.033e−06	9.36e−01	600	5.024e−06	5.76e−01	4612	9.301e−06	4.36e+00	1561	9.957e−06	1.50e+00
25	20	15	2753	9.123e−06	4.61e+00	1466	9.570e−06	2.19e+00	5778	1.285e−06	8.80e+00	2597	9.586e−06	3.99e+00
30	20	10	676	5.876e−06	1.77e+00	755	4.145e−06	1.73e+00	6752	9.414e−06	1.39e+01	3741	6.331e−06	7.97e+00
30	20	15	617	6.577e−06	1.28e+00	572	9.101e−06	9.99e−01	5543	2.885e−06	9.19e+00	1964	9.097e−06	3.32e+00
35	20	10	1499	9.355e−06	4.03e+00	961	9.865e−06	2.46e+00	5321	5.865e−06	1.42e+01	3575	9.896e−06	8.54e+00
35	25	15	1530	4.502e−06	6.54e+00	1144	8.773e−06	4.41e+00	8117	7.063e−06	3.28e+01	6210	5.171e−06	2.43e+01
35	35	10	689	8.391e−06	2.95e+00	827	8.409e−06	3.13e+00	7262	4.548e−06	2.78e+01	2752	4.963e−06	1.12e+01
40	10	10	747	9.002e−06	2.04e+00	834	9.492e−06	2.11e+00	3200	1.250e−06	8.43e+00	1381	9.404e−06	4.50e+00
40	20	10	620	9.800e−06	2.06e+00	704	9.663e−06	2.10e+00	4173	2.117e−06	1.36e+01	1656	6.838e−06	6.18e+00
40	20	15	1503	8.902e−06	8.05e+00	1029	8.648e−06	5.10e+00	10477	1.743e−06	4.75e+01	3801	8.420e−06	2.11e+01
40	20	20	1154	9.547e−06	7.89e+00	1242	9.883e−06	7.47e+00	7689	3.400e−06	4.84e+01	2575	5.132e−06	1.62e+01
40	25	15	2176	9.616e−06	1.28e+01	2089	9.994e−06	1.13e+01	13628	5.546e−06	6.99e+01	5249	6.896e−06	3.13e+01
50	20	10	1551	9.444e−06	1.11e+01	1197	9.954e−06	2.26e+01	6611	5.208e−06	4.61e+01	2855	9.896e−06	2.02e+01
60	30	10	1235	7.927e−06	1.65e+01	1831	6.135e−06	2.05e+01	5199	7.484e−06	6.59e+01	2898	8.928e−06	3.75e+01
60	35	10	974	9.119e−06	1.57e+01	1120	9.896e−06	1.63e+01	7666	1.148e−06	1.21e+02	4524	8.759e−06	7.06e+01
70	10	30	1973	3.783e−06	5.91e+01	1623	4.046e−06	5.04e+01	11353	7.577e−06	3.47e+02	4069	4.447e−06	1.18e+02
80	10	10	879	7.636e−06	1.87e+01	910	7.944e−06	1.81e+01	3991	2.497e−06	7.89e+01	3433	4.377e−06	1.63e+02
100	6	8	843	4.347e−06	2.85e+01	1104	9.990e−06	3.37e+01	3394	8.479e−06	9.33e+01	2104	9.525e−06	6.40e+01

Table 3 continued

n	m	M	TB_GPADMM			TB_APADMM			sGS_GPADMM			sGS_APADMM		
			Iter	Err_rel	Time (s)	Iter	Err_rel	Time (s)	Iter	Err_rel	Time (s)	Iter	Err_rel	Time (s)
110	3	9	2037	7.834e−06	9.49e+01	1186	9.135e−06	5.22e+01	6366	4.250e−06	2.39e+02	3448	6.028e−06	1.45e+02
115	4	8	1534	5.638e−06	7.92e+01	1107	9.633e−06	5.51e+01	3968	9.229e−06	1.69e+02	4058	4.513e−06	1.84e+02
120	2	4	1655	9.518e−06	6.54e+01	975	8.984e−06	3.66e+01	3969	8.384e−06	1.30e+02	2438	6.211e−06	8.80e+01
120	2	5	2909	9.284e−06	1.23e+02	1599	8.804e−06	6.76e+01	3891	7.784e−06	1.44e+02	2343	8.654e−06	8.92e+01
120	6	10	1320	7.219e−06	8.06e+01	1499	9.713e−06	9.13e+01	4793	8.541e−06	2.52e+02	3260	3.264e−06	1.76e+02
125	3	9	1667	6.999e−06	1.06e+02	1649	8.362e−06	1.06e+02	5152	6.185e−06	2.94e+02	4373	4.220e−06	2.71e+02
125	4	8	1645	6.946e−06	1.02e+02	1376	6.567e−06	8.73e+01	4580	7.452e−06	2.54e+02	3295	8.380e−06	1.86e+02
125	8	10	1300	9.097e−06	9.79e+01	1477	9.526e−06	1.11e+02	6139	9.097e−06	4.05e+02	3135	6.535e−06	2.11e+02
130	2	5	4193	8.088e−06	2.25e+02	1895	5.067e−06	1.02e+02	8307	9.648e−06	4.01e+02	3694	4.138e−06	1.82e+02
130	3	9	2007	5.607e−06	1.44e+02	1512	7.701e−06	1.12e+02	5274	5.579e−06	3.38e+02	3217	9.377e−06	2.12e+02
130	6	8	1210	9.863e−06	9.09e+01	1297	8.555e−06	9.84e+01	4832	7.171e−06	3.11e+02	2588	7.735e−06	1.73e+02
135	8	10	1368	9.240e−06	1.32e+02	1481	9.772e−06	1.42e+02	5498	8.629e−06	4.56e+02	3639	4.945e−06	3.09e+02
140	2	5	2414	8.395e−06	1.71e+02	1602	8.484e−06	1.17e+02	4236	9.980e−06	2.67e+02	3010	7.989e−06	1.94e+02
140	7	10	1225	9.203e−06	1.23e+02	1361	9.372e−06	1.40e+02	5295	9.403e−06	4.60e+02	3670	7.849e−06	3.48e+02
145	2	4	2181	4.441e−06	1.60e+02	1264	7.132e−06	9.43e+01	3557	7.884e−06	2.32e+02	2715	6.599e−06	1.93e+02
145	4	8	1446	2.780e−06	1.46e+02	1246	7.344e−06	1.29e+02	4668	7.445e−06	4.13e+02	3958	7.208e−06	3.57e+02
165	3	9	2843	5.058e−06	4.37e+02	2088	7.901e−06	3.24e+02	5809	8.871e−06	8.03e+02	3913	9.992e−06	5.71e+02
185	2	4	2237	7.741e−06	3.80e+02	1143	9.136e−06	2.08e+02	5135	8.788e−06	7.71e+02	1684	9.096e−06	2.91e+02
195	2	4	2476	7.627e−06	5.01e+02	1417	6.029e−06	3.03e+02	5205	9.923e−06	9.54e+02	3651	8.626e−06	7.25e+02
215	2	4	1479	9.923e−06	5.47e+02	742	1.035e−06	3.11e+02	2972	6.812e−06	8.33e+02	1339	7.248e−06	4.39e+02
225	2	4	1115	9.901e−06	4.42e+02	827	9.994e−06	3.58e+02	2201	9.445e−06	7.98e+02	2385	9.926e−06	8.77e+02
275	3	9	2949	5.388e−06	2.51e+03	2604	5.643e−06	2.29e+03	8069	8.693e−06	6.73e+03	4329	9.324e−06	3.65e+03
300	2	4	3962	7.933e−06	3.85e+03	2382	7.942e−06	2.38e+03	8098	9.126e−06	7.86e+03	2275	4.454e−06	2.42e+03

the relative residual errors (refer to SCS[3], COSMO[4], and MOSEK[5] for details on stopping criteria).

The numerical results in Tables 4, 5, and 6 reveal several key observations: (1) APADMM is the only algorithm consistently capable of providing solutions within the specified accuracy and maximal iteration across all problem scales. (2) Compared to the second-order method MOSEK, while MOSEK efficiently solves small-scale problems shown in Table 4, its computational time increases significantly with the problem size. For medium-scale problems shown in Table 5, APADMM outperforms MOSEK in computation time. Moreover, as shown in Table 6, MOSEK encounters memory errors when addressing large-scale problems. (3) When compared to the first-order methods SCS and COSMO, APADMM demonstrates superior performance in terms of iteration number and computation time. For instance, COSMO fails to achieve the required accuracy within the maximum iteration for small-scale problems shown in Table 4, and SCS faces similar challenges with medium and large-scale problems.

6 Conclusions and Future Work

In this paper, we have developed an accelerated PADMM to efficiently solve the CP problem, which serves as a relaxation of the $\mathcal{H}_2$-guaranteed cost ODC problem. By establishing the equivalence between the (generalized) PADMM and the relaxed PPA, we were able to accelerate the (generalized) PADMM using the Halpern fixed-point iteration method, achieving a fast $\mathcal{O}(1/k)$ convergence rate. To further enhance the efficiency of Algorithm 4, we employed the lifting technique for solving the PADMM subproblems. Numerical experiments on medium and large-scale CP problems, derived from $\mathcal{H}_2$-guaranteed cost ODC problems, demonstrate that the proposed accelerated PADMM outperforms COSMO, MOSEK, and SCS in terms of computation time. It is also important to note that the projections in our algorithm are independent, and we used direct methods for solving the linear systems. In future work, we will explore the development of a parallel framework for solving the non-smooth terms and incorporate iterative methods for solving the linear systems into our algorithm.

Appendix A The Definition of V_{j1} and V_{j2}

According to the definition of the set $\mathcal{W}$ in (6), W_1 and W_2 are m block-diagonal matrices with $\sum_{i=1}^{m} D_i = n$. The coefficient matrix for the equality constraints (7) can be constructed by utilizing the symmetry of W_1 and the sparse structure of W_2. We define the matrices $\{V_i\}_{i=1}^{2m}$ to extract the off-diagonal block parts from W and restrict the corresponding positions are zeros. Specifically, for $1 \leq i \leq m-1$, we define the

[3] https://www.cvxgrp.org/scs/index.html.

[4] https://oxfordcontrol.github.io/COSMO.jl/stable/.

[5] https://docs.mosek.com/latest/faq/index.html.

Table 4 Performance of **MOSEK**, **COSMO**, **SCS**, and **APADMM** for solving small-scale ODC problems

Size of LTI system	n=7, m=4	n=8, m=6	n=9, m=6	n=10, m=3	n=15, m=2	n=24, m=6
Number of vertices	M=5	M=8	M=8	M=6	M=5	M=8
Size of variable ξ	178	354	438	382	669	2758
Number of constraints	206	393	480	421	753	2865
Nonzeos in $\tilde{\mathcal{A}}^*$	1213	3226	4302	2716	8244	57694
MOSEK						
p_res	2.5e−10	2.8e−12	8.1e−11	5.4e−10	2.9e−14	1.5e−13
d_res	6.0e−10	5.6e−09	8.0e−11	8.9e−09	1.5e−07	5.4e−09
η_{gap}	1.7e−13	1.7e−15	4.8e−14	4.4e−13	1.5e−18	1.5e−16
iter	11	15	14	12	17	16
p_obj	8.1533	4.6586	4.1229	17.2072	1.6019	2.5349
d_obj	8.1533	4.6586	4.1229	17.2072	1.6019	2.5349
total time (s)	1.500e−02	1.600e−02	3.100e−02	3.100−02	4.600e−02	1.570e−01
status	solved	solved	solved	solved	solved	solved
COSMO(direct)						
p_res	3.898e+00	6.307e+00	4.149e+00	2.023e+00	3.124e+00	1.303e+01
d_res	4.521e−01	1.378e−02	7.739e−02	1.256e−02	7.281e−03	7.496e−01
iter	25000	25000	25000	25000	25000	25000
total time (s)	8.154e+00	1.161e+01	2.030e+01	2.825e+01	3.480e+01	1.213e+02
status	max_iter	max_iter	max_iter	max_iter	max_iter	max_iter

Table 4 continued

Size of LTI system	n=7, m=4	n=8, m=6	n=9, m=6	n=10, m=3	n=15, m=2	n=24, m=6
Number of vertices	M=5	M=8	M=8	M=6	M=5	M=8
Size of variable ξ	178	354	438	382	669	2758
Number of constraints	206	393	480	421	753	2865
Nonzeos in $\tilde{\mathcal{A}}^*$	1213	3226	4302	2716	8244	57694
SCS(direct)						
p_abs	1.164e−03	1.856e−03	6.250e−04	8.367e−04	1.684e−04	1.956e−04
d_abs	8.722e−06	8.023e−05	1.435e−05	1.342e−04	2.168e−06	1.715e−06
η_{gap}	2.903e−04	1.232e−04	5.084e−04	4.368e−04	2.593e−04	3.522e−04
iter	1200	1100	3750	2800	3050	9625
objective	8.1531	4.6586	4.1225	17.2076	1.6013	2.5338
total time (s)	6.79e−02	1.11e−01	3.84e−01	2.79e−01	5.06e−01	4.96e+00
obj_bias	1.98e−04	2.5e−05	4.37e−04	3.87e−04	6.18e−04	1.11e−03
status	solved	solved	solved	solved	solved	solved
APADMM						
p_abs	1.131e−03	2.097e−04	1.580e−04	8.211e−05	9.063e−04	2.666e−04
d_abs	3.994e−05	7.380e−06	1.996e−05	5.227e−05	3.549e−05	3.155e−05
η_{gap}	1.344e−03	8.472e−04	9.207e−04	3.491e−02	2.143e−04	4.630e−04
iter	220	204	250	225	223	356
p_obj	8.1534	4.6585	4.1228	17.2073	1.6010	2.5341
d_obj	8.1548	4.6599	4.1219	17.2108	1.6008	2.5336
total time (s)	1.51e−02	2.02e−02	3.20e−02	3.37e−02	6.08e−02	2.50e−01
obj_bias	1.33e−04	3.40e−05	1.40e−04	6.00e−05	8.48e−04	8.44e−04
status	solved	solved	solved	solved	solved	solved

Table 5 Performance of **MOSEK**, **COSMO**, **SCS**, and **APADMM** for solving medium-scale ODC problems

Size of LTI system	n=40, m=6	n=60, m=5	n=80, m=6	n=100, m=8	n=120, m=8	n=150, m=8
Number of vertices	M=8	M=6	M=8	M=10	M=8	M=10
Size of variable ξ	7421	12653	28974	55533	79698	124094
Number of constraints	7641	13125	29661	56386	80856	125811
Nonzeos in $\tilde{A}^*$	241981	571817	1786744	4367733	7445888	14325184
MOSEK						
p_res	6.2e−13	1.4e−14	1.5e−13	3.1e−14	7.8e−15	7.4e−14
d_res	6.9e−09	2.8e−08	8.08e−08	1.0e−08	3.1e−09	1.7e−06
η_{gap}	1.8e−15	8.5e−18	1.5e−15	1.5e−16	4.7e−17	1.7e−17
iter	15	16	19	17	20	22
p_obj	48.2255	6.1406	1.6705	48.4216	31.2284	3.87385
d_obj	48.2255	6.1406	1.6705	48.4216	31.2284	3.87385
total time (s)	8.430e−01	3.016e+00	1.550e+01	6.242e+01	2.568e+02	4.436e+02
status	solved	solved	solved	solved	solved	solved
COSMO(direct)						
p_res	2.654e+01	2.096e+02	3.625e+01	9.763e+01	1.395e+02	1.153e+02
d_res	6.557e−02	3.095e−02	5.057e−02	3.114e−01	2.108e+00	1.301e−01
iter	25000	25000	25000	25000	25000	25000
total time (s)	3.214e+02	7.453e+02	1.654e+03	3.845e+03	7.138e+03	9.694e+03
status	max_iter	max_iter	max_iter	max_iter	max_iter	max_iter

Table 5 continued

Size of LTI system	n=40, m=6	n=60, m=5	n=80, m=6	n=100, m=8	n=120, m=8	n=150, m=8
Number of vertices	M=8	M=6	M=8	M=10	M=8	M=10
Size of variable ξ	7421	12653	28974	55533	79698	124094
Number of constraints	7641	13125	29661	56386	80856	125811
Nonzeos in $\tilde{A}^*$	241981	571817	1786744	4367733	7445888	14325184
SCS(direct)						
p_abs	1.206e−03	6.402e−04	3.157e−03	1.012e−02	3.462e−03	5.152e−03
d_abs	7.394e−05	3.510e−06	8.599e−06	4.450e−05	3.265e−06	1.578e−05
η_{gap}	4.483e−03	2.727e−03	1.756e−02	2.171e−01	4.730e−02	1.388e−01
iter	10225	25000	25000	25000	25000	25000
objective	48.2183	6.1334	1.6381	47.8352	31.0848	3.6930
total time (s)	1.78e+01	1.13e+02	3.27e+02	7.10e+02	1.22e+03	2.50e+03
obj_bias	7.19e−03	7.15e−03	3.24e−02	5.86e−01	1.44e−01	1.81e−01
status	solved	max_iter	max_iter	max_iter	max_iter	max_iter
APADMM						
p_abs	8.657e−04	2.106e−03	3.688e−03	3.125e−03	3.396e−03	3.418e−03
d_abs	1.380e−05	3.738e−05	1.342e−05	3.961e−06	1.422e−06	2.815e−06
η_{gap}	9.705e−03	8.180e−04	3.963e−04	9.708e−03	6.060e−03	1.495e−04
iter	816	433	418	1138	1075	793
p_obj	48.2258	6.1354	1.6591	48.4200	31.2178	3.8623
d_obj	48.2161	6.1346	1.6586	48.4103	31.2118	3.8624
total time (s)	1.68e+00	2.77e+00	6.75e+00	4.31e+01	7.15e+01	1.09e+02
obj_bias	3.20e−04	5.19e−03	1.15e−02	1.57e−03	1.06e−02	1.16e−02
Status	Solved	Solved	Solved	Solved	Solved	Solved

Table 6 Performance of **MOSEK**, **SCS**, and **APADMM** for solving large-scale ODC problems

Size of LTI system	n=300, m=4	n=350, m=4	n=400, m=5	n=480, m=3
Number of vertices	M=6	M=6	M=6	M=4
Size of variable ξ	305515	415336	546696	539463
Number of constraints	317260	431385	563415	578646
Nonzeos in $\tilde{\mathcal{A}}^*$	65652859	104442568	155504316	178508123
MOSEK				
p_res	8.6e−14	1.1e−13	1.2e−14	***
d_res	5.1e−07	4.4e−06	2.0e−07	***
η_{gap}	9.2e−15	4.1e−16	4.0e−15	***
iter	21	24	21	***
p_obj	38.1333	6.8503	103.9257	***
d_obj	38.1333	6.8503	103.9257	***
total time (s)	1.385e+04	3.574e+04	1.619e+05	***
status	solved	solved	solved	memory error
SCS(direct)				
p_abs	6.748e−03	2.640e−03	1.053e−02	5.524e−03
d_abs	1.410e−05	5.786e−06	2.646e−05	9.562e−06
η_{gap}	3.904e−01	1.345e−01	5.251e−01	2.594e−01
iter	10000	10000	10000	10000
objective	37.5071	6.5015	102.6840	57.3878
total time (s)	9.27e+03	1.67e+04	2.84e+04	5.94e+04
obj_bias	6.26e−01	3.49e−01	1.24e−01	–
status	max_iter	max_iter	max_iter	max_iter
APADMM				
p_abs	1.090e−02	6.424e−03	1.042e−02	3.216e−02
d_abs	4.141e−06	6.578e−06	4.701e−06	7.001e−05
η_{gap}	7.543e−03	6.021e−04	1.953e−02	3.092e−03
iter	1362	1413	1683	1150
p_obj	38.0940	6.7651	103.9076	57.8089
d_obj	38.0865	6.7651	103.8881	57.8120
total time (s)	1.54e+03	2.91e+03	5.87e+03	9.18e+03
obj_bias	3.93e−02	8.52e−02	1.57e−02	–
Status	Solved	Solved	Solved	Solved

Entries marked *** indicate failure due to memory limitations

matrices V_i as follows:

$$V_1 = \begin{bmatrix} e_1^\top \\ \vdots \\ e_{D_1}^\top \end{bmatrix},\ V_2 = \begin{bmatrix} e_{D_1+1}^\top \\ \vdots \\ e_{D_1+D_2}^\top \end{bmatrix}, \ldots, V_{m-1} = \begin{bmatrix} e_{D_1+\ldots D_{m-2}+1}^\top \\ \vdots \\ e_{D_1+\ldots D_{m-2}+D_{m-1}}^\top \end{bmatrix} \in \mathbb{R}^{D_{m-1}\times p}.$$

For $i = m$, the matrix V_m is given by

$$V_m = e_{D_1+D_2+\cdots+D_m}^\top = e_n^\top \in \mathbb{R}^{1\times p}.$$

For $k = 1, \ldots, m$, V_{m+k} is a vector whose $(n+k)$-th element is 1 and all other components are 0, i.e.,

$$V_{m+k} = [\underbrace{O_{1\times D_1}, \ldots, O_{1\times D_m}}_{\sum_{i=1}^m D_i = n}, \underbrace{0, \ldots, 1}_{n+k}, \ldots, 0] \in \mathbb{R}^{1\times p}.$$

The block diagonal structure of W_1 and W_2 in (6), along with the symmetry of W_1, implies that $N = (3m(m-1))/2$ off-diagonal block submatrices of W need to be zero. Using the defined matrices $\{V_i\}_{i=1}^{2m}$, we extract these off-diagonal blocks and enforce zero through the following linear constraints:

$$\begin{aligned} &V_i W V_{i+1}^\top = 0, \ldots, V_i W V_{i+m-1}^\top = 0, \\ &V_i W V_{i+m+1}^\top = 0, \ldots, V_i W V_{2m}^\top = 0,\ i = 1, \ldots, m-1, \end{aligned}$$

and

$$V_m W V_{m+1}^\top = 0, \ldots, V_m W V_{2m-1}^\top = 0,\ i = m.$$

Next, we divide the index array $[1, 2, \ldots, \frac{3m(m-1)}{2}]$ into m blocks:

$$\left[\underbrace{1, 2, \ldots}_{1} | \ldots | \underbrace{\ldots}_{i} | \ldots | \underbrace{\ldots, \frac{3m(m-1)}{2}}_{m}\right],$$

where for i-th block, the starting index is

$$(i-1)(2m-2) - \frac{(i-1)(i-2)}{2} + 1,\ i = 1, \ldots, m.$$

According to the above decomposition, we reorder $\{V_i\}_{i=1}^{2m}$ and construct V_{j1} and V_{j2} for $j = 1, \ldots, N$. Specifically, we define the matrix V_{j1} as follows

$$V_{j1} = \begin{cases} V_1, & \text{for } j = 1, 2, \ldots, 2m-2, \\ V_2, & \text{for } j = 2m-1, \ldots, 4m-5, \\ \vdots & \\ V_{m-1}, & \text{for } j = T+1, \ \ldots, \ T+m, \\ V_m, & \text{for } j = L+1, \ \ldots, \ L+m-1, \end{cases}$$

where

$$T = (m-2)(2m-2) - \frac{(m-2)(m-3)}{2},$$

$$L = (m-1)(2m-2) - \frac{(m-1)(m-2)}{2}.$$

For each $s(i) = [i+1,\ \ldots,\ i+(m-1),\ i+(m+1),\ \ldots,\ 2m]$, the matrix V_{j2} is defined by

$$V_{j2} = \begin{cases} V_{s(1)_j}, & \text{for } j = 1,\ 2, \ldots,\ 2m-2, \\ V_{s(2)_{j-(2m-2)}}, & \text{for } j = 2m-1,\ \ldots,\ 4m-5, \\ \vdots & \\ V_{s(m-1)_{j-T}}, & \text{for } j = T+1,\ \ldots,\ T+m, \\ V_{s(m)_{j-L}}, & \text{for } j = L+1,\ \ldots,\ L+m-1. \end{cases}$$

Appendix B The Equivalence of the Semi-proximal ADMM and the (Partial) PPA

In this section, we establish the equivalence between the proximal point method and the semi-proximal ADMM, which encompasses the version of PADMM used in this paper. Consider the following linearly constrained convex optimization problem with separable objective functions:

$$\begin{aligned} &\min_{x,y}\ f(x) + g(y) \\ &\text{s.t.}\ \ \mathcal{F}x + \mathcal{G}y = c, \end{aligned} \tag{34}$$

where $f : \mathcal{X} \to (-\infty, +\infty]$ and $g : \mathcal{Y} \to (-\infty, +\infty]$ are proper closed convex functions, $\mathcal{F} : \mathcal{X} \to \mathcal{Z}$ and $\mathcal{G} : \mathcal{X} \to \mathcal{Z}$ are given linear operators, $c \in \mathcal{Z}$ is given data, and $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ are real finite-dimensional Euclidean spaces, each equipped with an inner product $\langle \cdot,\ \cdot \rangle$ and its induced norm $\|\cdot\|$. The dual problem of (34) can be written as

$$\max_{z} \{-\langle c,\ z\rangle - f^*(-\mathcal{F}^*z) - g^*(-\mathcal{G}^*z)\}. \tag{35}$$

The KKT system associated with (34) and (35) is given by

$$\begin{aligned} 0 &\in \partial f(x) + \mathcal{F}^*z, \\ 0 &\in \partial g(y) + \mathcal{G}^*z, \\ 0 &= c - \mathcal{F}x - \mathcal{G}y. \end{aligned} \tag{36}$$

Let Ω be the solution set to the above KKT system. Suppose that $\Omega \neq \emptyset$. Define

$$\mathcal{T}_l(x, y, z) := \begin{pmatrix} \partial f(x) + \mathcal{F}^*z \\ \partial g(y) + \mathcal{G}^*z \\ c - \mathcal{F}x - \mathcal{G}y \end{pmatrix},$$

then $\mathcal{T}_l$ is a maximal monotone operator and $\mathcal{T}_l^{-1}(0) \neq \emptyset$. Given $\sigma > 0$, the augmented Lagrangian function associated with (34) is given by:

$$\mathcal{L}_\sigma(x, y; z) = f(x) + g(y) + \langle z,\ \mathcal{F}x + \mathcal{G}y - c\rangle + \frac{\sigma}{2}\|\mathcal{F}x + \mathcal{G}y - c\|^2.$$

Given two self-adjoint linear operators $\mathcal{S}_1 : \mathcal{X} \to \mathcal{X}$ and $\mathcal{S}_2 : \mathcal{Y} \to \mathcal{Y}$, define a self-adjoint linear operator $\mathcal{S} : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ as follows:

$$\mathcal{S} := \begin{pmatrix} \mathcal{S}_1 & & \\ & \sigma\mathcal{G}^*\mathcal{G} + \mathcal{S}_2 & \mathcal{G}^* \\ & \mathcal{G} & \sigma^{-1}\mathcal{I} \end{pmatrix},$$

where $\mathcal{I}$ is the identity operator on $\mathcal{Z}$. The following proposition establishes the equivalence between the proximal point method and the semi-proximal ADMM.

Proposition B.1 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two self-adjoint positive semi-definite linear operators such that $\partial f(\cdot) + \sigma\mathcal{F}^*\mathcal{F} + \mathcal{S}_1$ and $\partial g(\cdot) + \sigma\mathcal{G}^*\mathcal{G} + \mathcal{S}_2$ are maximal and strongly monotone, respectively. Consider $(x^0, y^0, z^0) \in \mathrm{dom}(f) \times \mathrm{dom}(g) \times \mathcal{Z}$ and $\rho \in \mathbb{R}$. Then the point (x^+, y^+, z^+) generated by the following generalized semi-proximal ADMM scheme (37)*

$$\begin{aligned} \bar{x}^+ &= \operatorname*{argmin}_x \mathcal{L}_\sigma(x, y^0; z^0) + \frac{1}{2}\left\|x - x^0\right\|^2_{\mathcal{S}_1}, \\ \bar{z}^+ &= z^0 + \sigma(\mathcal{F}\bar{x}^+ + \mathcal{G}y^0 - c), \\ \bar{y}^+ &= \operatorname*{argmin}_y \mathcal{L}_\sigma(\bar{x}^+, y; \bar{z}^+) + \frac{1}{2}\left\|y - y^0\right\|^2_{\mathcal{S}_2}, \\ (x^+, y^+, z^+) &= (1-\rho)(x^0, y^0, z^0) + \rho(\bar{x}^+, \bar{y}^+, \bar{z}^+) \end{aligned} \tag{37}$$

is equivalent to the one generated by the following proximal point scheme (38)

$$\begin{aligned} &0 \in \mathcal{T}_l(\bar{x}^+, \bar{y}^+, \bar{z}^+) + \mathcal{S}(\bar{x}^+ - x^0, \bar{y}^+ - y^0, \bar{z}^+ - z^0), \\ &(x^+, y^+, z^+) = (1-\rho)(x^0, y^0, z^0) + \rho(\bar{x}^+, \bar{y}^+, \bar{z}^+). \end{aligned} \tag{38}$$

Proof Since $\partial f(\cdot) + \sigma\mathcal{F}^*\mathcal{F} + \mathcal{S}_1$ and $\partial g(\cdot) + \sigma\mathcal{G}^*\mathcal{G} + \mathcal{S}_2$ are assumed to be maximal and strongly monotone, the objective function of each subproblem in the semi-proximal ADMM scheme is a proper closed strongly convex function [38, Exercise 8.8 and Exercise 12.59]. Consequently, one can deduce from [38, Theorem 8.15 and Proposition 12.54] that $\bar{x}^+ = \operatorname*{argmin}_x \mathcal{L}_\sigma(x, y^0; z^0) + \frac{1}{2}\left\|x - x^0\right\|^2_{\mathcal{S}_1}$ if and only if

$$0 \in \partial f(\bar{x}^+) + \mathcal{F}^*(z^0 + \sigma(\mathcal{F}\bar{x}^+ + \mathcal{G}y^0 - c)) + \mathcal{S}_1(\bar{x}^+ - x^0),$$

and $\bar{y}^+ = \operatorname*{argmin}_y \mathcal{L}_\sigma(\bar{x}^+, y; \bar{z}^+) + \frac{1}{2}\left\|y - y^0\right\|^2_{\mathcal{S}_2}$ if and only if

$$0 \in \partial g(\bar{y}^+) + \mathcal{G}^*(\bar{z}^+ + \sigma(\mathcal{F}\bar{x}^+ + \mathcal{G}\bar{y}^+ - c)) + \mathcal{S}_2(\bar{y}^+ - y^0).$$

Hence, (37) is equivalent to

$$\begin{aligned}
&0 \in \partial f(\bar{x}^+) + \mathcal{F}^*(z^0 + \sigma(\mathcal{F}\bar{x}^+ + \mathcal{G}y^0 - c)) + \mathcal{S}_1(\bar{x}^+ - x^0),\\
&\bar{z}^+ = z^0 + \sigma(\mathcal{F}\bar{x}^+ + \mathcal{G}y^0 - c),\\
&0 \in \partial g(\bar{y}^+) + \mathcal{G}^*(\bar{z}^+ + \sigma(\mathcal{F}\bar{x}^+ + \mathcal{G}\bar{y}^+ - c)) + \mathcal{S}_2(\bar{y}^+ - y^0),\\
&(x^+, y^+, z^+) = (1-\rho)(x^0, y^0, z^0) + \rho(\bar{x}^+, \bar{y}^+, \bar{z}^+).
\end{aligned} \tag{39}$$

Note that for any (x^+, y^+, z^+) satisfying (39), one has

$$-\sigma^{-1}(\bar{z}^+ - z^0) = -(\mathcal{F}\bar{x}^+ + \mathcal{G}y^0 - c) = c - \mathcal{F}\bar{x}^+ - \mathcal{G}\bar{y}^+ + \mathcal{G}(\bar{y}^+ - y^0).$$

Therefore, (39) can be recast as

$$\begin{aligned}
&0 \in \partial f(\bar{x}^+) + \mathcal{F}^*\bar{z}^+ + \mathcal{S}_1(\bar{x}^+ - x^0),\\
&0 \in \partial g(\bar{y}^+) + \mathcal{G}^*\bar{z}^+ + (\sigma\mathcal{G}^*\mathcal{G} + \mathcal{S}_2)(\bar{y}^+ - y^0) + \mathcal{G}^*(\bar{z}^+ - z^0),\\
&0 = c - \mathcal{F}\bar{x}^+ - \mathcal{G}\bar{y}^+ + \mathcal{G}(\bar{y}^+ - y^0) + \sigma^{-1}(\bar{z}^+ - z^0),\\
&(x^+, y^+, z^+) = (1-\rho)(x^0, y^0, z^0) + \rho(\bar{x}^+, \bar{y}^+, \bar{z}^+),
\end{aligned}$$

which is exactly the scheme (38) by the definition of $\mathcal{T}_l$ and $\mathcal{S}$. Hence, the point (x^+, y^+, z^+) generated by the generalized semi-proximal ADMM scheme (37) is equivalent to the one generated by the proximal point scheme (38). □

Acknowledgements The research of Xinyuan Zhao was supported in part by the National Natural Science Foundation of China under Project No. 12271015, Xudong Li's research was supported in part by the National Natural Science Foundation of China under Project No. 12271107 and Defeng Sun's research was supported in part by the Hong Kong Research Grant Council Grant under Project No. N PolyU504/19. The authors also wish to thank Mr. Guojun Zhang from the Department of Applied Mathematics at The Hong Kong Polytechnic University for helpful discussions regarding the equivalence between semi-proximal ADMM and (partial) PPA, as discussed in Appendix B.

Data Availability The data supporting the findings of this study are available from the corresponding author upon reasonable request.

References

1. Borrelli, F., Keviczky, T.: Distributed LQR design for identical dynamically decoupled systems. IEEE Trans. Autom. Control **53**(8), 1901–1912 (2008)
2. Callier, F.M., Desoer, C.A.: Linear System Theory. Springer, New York (1991)
3. Combettes, P., Bauschke, H.H.: Convex Analysis and Monotone Operators Theory in Hilbert Spaces, 2nd edn. Springer, New York (2011)
4. Contreras, J.P., Cominetti, R.: Optimal error bounds for non-expansive fixed-point iterations in normed spaces. Math. Program. **199**, 343–374 (2022)
5. Cui, Y., Li, X.D., Sun, D.F., Toh, K.C.: On the convergence properties of a majorized alternating direction method of multipliers for linearly constrained convex optimization problems with coupled objective functions. J. Optim. Theory App. **169**(3), 1013–1041 (2016)
6. Date, R.A., Chow, J.H.: A parametrization approach to optimal $\mathcal{H}_2$ and $\mathcal{H}_\infty$ decentralized control problems. Automatica **29**(2), 457–463 (1993)

7. Doyle, J.C., Glover, K., Khargonekar, P.P., Francis, B.A.: A State space solution to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problem. IEEE Trans. Aut. Control. **34**, 831–846 (1989)
8. Davis, T.A.: Algorithm 849: a concise sparse Cholesky factorization package. ACM Trans. Math. Softw. **31**(4), 587–591 (2005)
9. Davis, T.A.: Direct Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia (2006)
10. Eckstein, J., Bertsekas, D.P.: On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**(1–3), 293–318 (1992)
11. Eckstein, J.: Some saddle-function splitting methods for convex programming. Optim. Methods Softw. **4**(1), 75–83 (1994)
12. Furieri, L., Zheng, Y., Papachristodoulou, A., Kamgarpour, M.: Sparsity invariance for convex design of distributed controllers. IEEE Trans. Control. Netw. Syst. **7**(4), 1836–1847 (2020)
13. Furieri, L., Zheng, Y., Papachristodoulou, A., Kamgarpour, M.: An input-output parametrization of stabilizing controllers: amidst Youla and system level synthesis. IEEE Control Syst. Lett. **3**(4), 1014–1019 (2019)
14. Garstka, M., Cannon, M., Goulart, P.J.: COSMO: a conic operator splitting method for convex conic problems. J. Optim. Theory Appl. **190**, 779–810 (2021)
15. Geromel, J.C., Bernussou, J., Peres, P.L.D.: Decentralized control through parameter space optimization. Automatica **30**(10), 1565–1578 (1994)
16. Halpern, B.: Fixed points of nonexpanding maps. Bull. Am. Math. Soc. **73**(6), 957–961 (1967)
17. Hung, Y.S., MacFarlane, A.G.J.: Multivariable Feedback: A Quasi-Classical Approach. Springer, Berlin (1982)
18. Kim, D.W.: Accelerated proximal point method for maximally monotone operators. Math. Program. **190**(1–2), 57–87 (2021)
19. Lam, X.Y., Marron, J.S., Sun, D.F., Toh, K.C.: Fast algorithms for large-scale generalized distance weighted discrimination. J. Comput. Graph. Stat. **27**(2), 368–379 (2018)
20. Lavaei, J.: Decentralized implementation of centralized controllers for interconnected systems. IEEE Trans. Autom. Control **57**(7), 1860–1865 (2012)
21. Lessard, L., Lall, S.: Quadratic invariance is necessary and sufficient for convexity. In: Proceedings of the 2011 American Control Conference, pp. 5360–5362 (2011)
22. Lessard, L., Lall, S.: Convexity of decentralized controller synthesis. IEEE Trans. Autom. Control **61**(10), 3122–3127 (2016)
23. Li, X.D., Sun, D.F., Toh, K.C.: A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions. Math. Program. **155**, 333–373 (2016)
24. Li, X.D., Sun, D.F., Toh, K.C.: A block symmetric Gauss-Seidel decomposition theorem for convex composite quadratic programming and its applications. Math. Program. **175**, 395–418 (2019)
25. Li, X.D., Sun, D.F., Toh, K.C.: An asymptotically superlinearly convergent semismooth Newton augmented Lagrangian method for linear programming. SIAM J. Optim. **30**(3), 2410–2440 (2020)
26. Liang, L., Sun, D.F., Toh, K.C.: An inexact augmented Lagrangian method for second-order cone programming with applications. SIAM J. Optim. **31**(3), 1748–1773 (2021)
27. Lieder, F.: On the convergence rate of the Halpern-iteration. Optim. Lett. **15**(1–2), 405–418 (2021)
28. Lin, W.X., Bitar, E.: A convex information relaxation for constrained decentralized control design problems. IEEE Trans. Autom. Control **64**(11), 4788–4795 (2019)
29. Lunze, J.: Feedback Control of Large-Scale Systems. Prentice Hall, Englewood Cliffs, NJ (1992)
30. Ma, J., Cheng, Z.L., Zhang, X.X., Tomizuka, M., Lee, T.H.: Optimal decentralized control for uncertain systems by symmetric Gauss–Seidel semi-proximal ALM. IEEE Trans. Autom. Control **66**(11), 5554–5560 (2021)
31. MOSEK, ApS. The MOSEK optimization toolbox for MATLAB manual. Version 10.0. (2022)
32. Motee, N., Jadbabaie, A.: Optimal control of spatially distributed systems. IEEE Trans. Autom. Control **53**(7), 1616–1629 (2008)
33. Nemirovski, A.: Efficient methods in convex programming (1994). https://www2.isye.gatech.edu/~nemirovs/Lect_EMCO.pdf
34. Nesterov, Y.: A method for unconstrained convex minimization problem with convergence rate $\mathcal{O}(1/k^2)$. Dokl. Akad. Nauk SSSR **269**, 543–547 (1983)
35. Nesterov, Y.: Gradient methods for minimizing composite functions. Math. Program. **140**(1), 125–61 (2013)

36. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**(3), 1042–1068 (2016)
37. Palhars, R.M., Taicahashi, R.H.C., Peres, P.L.D.: $\mathcal{H}_\infty$ and $\mathcal{H}_2$ guaranteed costs computation for uncertain linear systems. Int. J. Syst. Sci. **28**(2), 183–188 (1997)
38. Rockafellar, R.T., Wets, R.J.-B.: Variational Analysis. Springer, Berlin (1998)
39. Rotkowitz, M.C., Martins, N.C.: On the nearest quadratically invariant information constraint. IEEE Trans. Autom. Control **57**(5), 1314–1319 (2012)
40. Scorletti, G., Duc, G.: An LMI approach to dencentralized $\mathcal{H}_\infty$ control. Int. J. Control **74**(3), 211–224 (2001)
41. Siljak, D.D.: Decentralized control and computations: status and prospects. Annu. Rev. Control. **20**, 131–141 (1996)
42. Siljak, D.D.: Decentralized Control of Complex Systems. Academic Press, Boston (2012)
43. Todd, M.J., Toh, K.C., Tütüncü, R.H.: On the Nesterov–Todd direction in semidefinite programming. SIAM J. Optim. **8**(3), 769–796 (1998)
44. Tsitsiklis, J., Athans, M.: On the complexity of decentralized decision making and detection problems. IEEE Trans. Autom. Control **30**(5), 440–446 (1985)
45. Witsenhausen, H.S.: Quasimonotonicity, regularity and duality for nonlinear systems of partial differential equations. SIAM J. Control. **6**(1), 131–147 (1968)
46. Wittmann, R.: Approximation of fixed points of nonexpansive mappings. Arch. Math. **58**(5), 486–491 (1992)
47. Zhai, G., Ikeda, M., Fujisaki, Y.: Decentralized $\mathcal{H}_\infty$ controller design: a matrix inequality approach using a homotopy method. Automatica **37**(4), 565–572 (2001)
48. Zhang, G.J., Yuan, Y.C., Sun, D.F.: An efficient HPR algorithm for the Wasserstein barycenter problem with $\mathcal{O}(Dim(P)/\varepsilon)$ computational complexity (2022). arXiv Preprint arXiv: 2211.14881v1
49. Zhang, X.X., Ma, J., Cheng, Z.L., Wang, W.X., Chen, X.L., Lee, T.H.: sGS-sPALM for optimal decentralized control: a distributed optimization approach. In: IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, pp. 1–6 (2021)