Automatica 155 (2023) 111157

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Primal–dual interior-point algorithm for symmetric model predictive control☆

Shirin Panahi, Ali Kashani, Claus Danielson *

Department of Mechanical Engineering, University of New Mexico, Albuquerque, NM, 87106, United States

ARTICLE INFO

Article history:
Received 1 September 2022
Received in revised form 6 March 2023
Accepted 9 May 2023
Available online 4 July 2023

Keywords:
Symmetry
Model predictive control
Optimization
Interior-point

ABSTRACT

This paper presents a primal–dual interior-point (PDIP) optimization algorithm for solving extreme-scale model predictive control (MPC) problems with linear dynamics, polytopic constraints, and quadratic/linear costs which are all invariant under the symmetric-group. We show that exploiting symmetry can reduce the computational and memory burden of extreme-scale or fast-paced applications of MPC. Our algorithm transforms the original inputs, states, and constraints of the MPC problem into a symmetric domain. The premise of our algorithm is that the numerical linear algebra used to solve the optimization problem has lower computational and memory complexity in the transformed domain. We demonstrate our algorithm for a heating, ventilation, and air-conditioning (HVAC) numerical example. We show that, for our largest HVAC control problem, our symmetry exploiting the PDIP algorithm reduces the computation-time from minutes to seconds in comparison with the baseline PDIP algorithm. Furthermore, we show that the presented symmetry exploiting PDIP algorithm outperforms a state-of-the-art symmetry exploiting optimization algorithm.

1. Introduction

Model predictive control (MPC) is a popular model-based control technique (Camacho & Alba, 2013), widely used in industry for high-performance control of systems subject to constraints (Qin & Badgwell, 2003). MPC has many advantages: it is intrinsically formulated for multi-input/multi-output systems, it can explicitly enforce constraints, and it typically provides good closed-loop performance due to its optimization-based nature. However, MPC for extreme-scale or fast-paced systems is inherently computationally challenging since it requires solving large optimization problems in real-time on embedded platforms with limited computational and memory resources.

This paper presents a novel PDIP optimization algorithm for solving extreme-scale quadratic programs (QPs) and linear programs (LPs) that naturally arise from MPC for dynamical systems with linear dynamics, polytopic constraints, and quadratic or linear costs, respectively. Furthermore, this algorithm is useful for nonlinear MPC, since it is often solved using sequential quadratic programming (Berberich, Köhler, Müller, & Allgöwer, 2022; Messerer, Baumgärtner, & Diehl, 2021), wherein a nonlinear program (NLP) is iteratively solved by solving a sequence of QP approximations that converge to a KKT point. The presented algorithm can be used to efficiently solve the sequence of QPs. This paper adapts an established PDIP algorithm (Borrelli, Bemporad, & Morari, 2017) to exploit symmetric problem structure to enable real-time implementation of MPC. interior-point (IP) algorithms are a class of optimization algorithms for convex optimization problems (Mehrotra, 1992; Wright, 1997), wherein Newton iterations (Luus, 2019) iteratively find a KKT point. PDIP algorithms are a subclass of IP algorithms that iteratively linearize the complementary slackness KKT condition and solve the resulting linearized KKT conditions. This paper exploits symmetries of the MPC problem to reduce the computational and memory costs of the PDIP algorithm.

Intuitively, symmetries are patterns in the MPC problem. In practice, these patterns arise from the substantial repetition of mass-produced components that are organized in regular patterns. For instance, a battery pack in an electric vehicle (Danielson et al., 2012; Danielson, Borrelli, Oliver, Anderson, & Phillips, 2013; Preindl, Danielson, & Borrelli, 2013) or HVAC systems for a large building (Bortoff et al., 2018; Bortoff, Schwerdtner, Danielson, Cairano, & Burns, 2022; Burns, Danielson, Zhou, & Di Cairano, 2017). More formally, a symmetry of an MPC problem is a transformation of the inputs, outputs, and states that preserve the cost, dynamics, and constraints. See Danielson and Borrelli (2014)

☆ This material is based upon work supported by the National Science Foundation under NSF Grant Number CMMI-2105631. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Martin Monnigmann under the direction of Editor Ian R Petersen.

* Corresponding author.
E-mail addresses: spanahi@unm.edu (S. Panahi), kashani@unm.edu (A. Kashani), cdanielson@unm.edu (C. Danielson).

https://doi.org/10.1016/j.automatica.2023.111157
0005-1098/

for details on identifying the symmetries of a constrained control problem. It is intuitively obvious that these symmetries can be exploited to reduce the computational burden of an MPC for extreme-scale systems. However, it is non-trivial to design optimization algorithms that can intelligently exploit these symmetries, especially when the patterns are obscured by high-dimensional and highly-coupled dynamics, costs, and constraints.

Symmetry has been widely exploited in the field of optimization (Bödi, Grundhöfer, & Herr, 2011; Boyd, Diaconis, Parrilo, & Xiao, 2009; Cogill, Lall, & Parrilo, 2008; Danielson & Di Cairano, 2015; Margot, 2009; Vallentin, 2009). However, their definition of symmetry is restrictive for control applications since it requires that the states, references, and disturbances are symmetric, as well as the dynamics, cost, and constraints. In contrast, our broader definition (Danielson & Borrelli, 2012, 2015b) of symmetry only requires that the cost function, dynamics, and constraints are symmetric. The main challenge addressed in this paper is that this broader symmetry prevents the full decomposition of the numerical linear algebra used to solve the KKT conditions. Nonetheless, we can reduce the computational complexity by transforming between domains to exploit the advantageous structure of different blocks in the KKT matrix.

Recently, Danielson (2021) proposed a symmetry exploiting ADMM algorithm to solve extreme-scale MPC problems which are invariant under the symmetric group. The ADMM algorithm decomposed the optimization problem into three sub-problems which were alternatively solved to find the optimal solution. Symmetry was exploited to decompose the most expensive of these sub-problems, reducing its computational complexity from quadratic to linear. As a result, the computational complexity of each iteration of the ADMM algorithm was reduced from quadratic to linear. This paper provides an alternative optimization algorithm for solving this symmetric MPC problem. According to the "no free lunch theorem" (Wolpert & Macready, 1997), there is no perfect optimization problem; the relative performance of algorithms will vary between problems as well as computer architectures. For instance, Section 5 shows that the presented PDIP is faster than the previous ADMM algorithm. However, the ADMM algorithm has lower read–write memory requirements, making it better suited for certain architectures. Together, this paper and Danielson (2021) provide practitioners with two options for solving their particular symmetric MPC problem on their specific computational hardware.

The remainder of this paper is organized as follows. In Section 2, we describe the baseline PDIP algorithm on which our symmetric algorithm is based. In Section 3, we formally define symmetry and describe the symmetric decomposition. In Section 4, we present our symmetry exploiting the PDIP algorithm and study its computational and memory benefits. Finally, in Section 5, we apply our PDIP algorithm to a case study; HVAC control problem.

Notation and Definitions: $\mathbb{R}$ and $\mathbb{R}_+$ denote the reals and positive reals, respectively. $I_n \in \mathbb{R}^{n\times n}$, $\mathbf{1}$ and 0 denote the identity matrix, all-ones vector, and zero vector (or matrix), respectively. For vectors $a, b \in \mathbb{R}^n$ and matrices $A, B \in \mathbb{R}^{n\times n}$ we will denote the 2-norm as $\|a\|$, element-wise division as $a./b$, the diagonal-operator as $A = \text{diag}(a)$ where $A_{ii} = a_i$ and $A_{ij} = 0$. We denote the vertical concatenation as (A, B), the Kronecker product as $A \otimes B$, and the direct sum as $A \oplus B$. The matrix inequality $A \succeq B$ means that $A - B \succeq 0$ is positive semidefinite.

2. Problem statement: MPC and PDIP

MPC obtains the input $u(t) = u_0^\star$ by solving the following constrained finite-horizon optimal control problem

$$\min_{x_k,u_k} \ \frac{1}{2}x_N^\top \mathcal{P} x_N + \frac{1}{2}\sum_{k=0}^{N-1}\begin{bmatrix}x_k\\u_k\end{bmatrix}^\top\begin{bmatrix}\mathcal{Q} & \mathcal{S}\\ \mathcal{S}^\top & \mathcal{R}\end{bmatrix}\begin{bmatrix}x_k\\u_k\end{bmatrix} \tag{1a}$$

$$\text{s.t. } x_{k+1} = \mathcal{A}x_k + \mathcal{B}u_k, \quad x_0 = x(t), \tag{1b}$$

$$\mathcal{C}x_k + \mathcal{D}u_k \le \bar{y}, \tag{1c}$$

$$\text{for } k = 0, \dots, N-1,$$

where $x_k \in \mathbb{R}^{n^x}$ is the predicted state over the horizon N under the control inputs $u_k \in \mathbb{R}^{n^u}$ and initial condition $x_0 = x(t)$, where $x(t)$ is the current state of the plant. Without loss of generality, polytopic state, and input constraints can be described by (1c) using properly defined $\mathcal{C}$ and $\mathcal{D}$, and bounds $\bar{y} \in \mathbb{R}^{n^y}$. See Danielson and Borrelli (2015a) for details about adding a symmetric stabilizing terminal cost and a symmetric recursively feasible terminal constraint. We make the following assumptions about the MPC problem (1):

Assumption 1. (i) The cost matrices satisfy $\mathcal{R} \succ 0$ and $\mathcal{Q} - \mathcal{S}\mathcal{R}^{-1}\mathcal{S}^\top \succeq 0$. (ii) The pair $(\mathcal{A}, \mathcal{B})$ is controllable. (iii) The pair $(\mathcal{A} - \mathcal{B}\mathcal{R}^{-1}\mathcal{S}^\top, \mathcal{Q} - \mathcal{S}\mathcal{R}^{-1}\mathcal{S}^\top)$ is observable.

Assumption (i) ensures that the MPC problem (1) is strictly convex and therefore has a unique solution. Although the presented algorithm is valid for non-strictly convex QPs, our convergence analysis will require a unique solution to which the algorithms converge. Assumptions (ii) and (iii) ensure that the MPC is stabilizing.

To apply the PDIP algorithm, we rewrite the MPC (1) in the reduced batch QP from (Boyd & Vandenberghe, 2004)

$$\min_{U,s} \ \frac{1}{2}U^\top H U + f^\top U \tag{2a}$$

$$\text{s.t. } AU + s = b, \quad s \ge 0, \tag{2b}$$

where $0 \le s \in \mathbb{R}_+^{Nn^y}$ contains the slack variables, and $U = (u_0, \dots, u_{N-1})$. See Boyd and Vandenberghe (2004) for details about converting (1) into (2). Solving the QP (2) is equivalent to solving the KKT conditions $HU + f + A^\top\lambda = 0$, $AU + s = b$, $\Lambda s = 0$, and $\Lambda, S \succeq 0$, where $0 \le \lambda \in \mathbb{R}_+^{Nn^y}$ contains the inequality-dual variables, $\Lambda = \text{diag}(\lambda)$, and $S = \text{diag}(s)$. Note that A and S are different from $\mathcal{A}$ and $\mathcal{S}$ in (1).

Algorithm 1
Primal-Dual Interior-Point

input: U_0, $\lambda_0 \ge 0$, $s_0 \ge 0$, $\sigma \in (0, 1)$
repeat
1: Average complementarity violation $\mu_i = s_i^\top\lambda_i/n$
2: Solve linearized KKT system (4) for direction
3: Slack direction (3)
4: Compute step-sizes:
$\alpha_p = \min(-\kappa/\min(\delta s./s), 1)$
$\alpha_d = \min(-\kappa/\min(\delta\lambda./\lambda), 1)$
for some $\kappa \in (0, 1)$
5: Update primal and dual variables:
$U^+ = U + \alpha_p\delta U$
$s^+ = s + \alpha_d\delta s \ge 0$
$\lambda^+ = \lambda + \alpha_d\delta\lambda \ge 0$
until $\sum_{j=1}^m \|U_j^+ - U_j\|^2 < \epsilon$, $\sum_{j=1}^m \|\lambda_j^+ - \lambda_j\|^2 < \epsilon$, and $\sum_{j=1}^m \|s_j^+ - s_j\|^2 < \epsilon$
output: U

The baseline PDIP algorithm is summarized by Algorithm 1 (see Borrelli et al. (2017) for details). The algorithm solves (2) by iteratively linearizing the complementary slackness $\Lambda_i s_i \approx \Lambda_{i-1}s_{i-1} + \Lambda_{i-1}\delta s + S_{i-1}\delta\lambda$ in each iteration i, and updating $(.)_{i+1} = (.)_i + \delta(.)$ the estimated KKT-point $(U_{i+1}, \lambda_{i+1}, s_{i+1})$. In contrast to typical Newton iterations, we relax the complementary slackness

by setting $\Lambda_i s_i = \mu_i \sigma \mathbf{1}$, which is tightened using the average violation $\mu_i = s_{i-1}^\top \lambda_{i-1}/n$ of the complementary slackness during the previous iteration $i-1$, where $\sigma \in (0,1)$ is the centering parameter, and n is the length of the vectors s and λ. Then, we use the linearized complementary slackness to substitute δs by

$$\delta s = -\Lambda_{i-1}^{-1} S_{i-1} \delta\lambda - s_{i-1} + \sigma \mu_i \Lambda_i^{-1} \mathbf{1} \tag{3}$$

to get the following reduced linearized KKT system,

$$\begin{bmatrix} H & A^\top \\ A & -V_i \end{bmatrix} \begin{bmatrix} \delta U \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} HU_i + f + A^\top \lambda_i \\ AU_i - b + \sigma \mu_i \Lambda_i^{-1} \mathbf{1} \end{bmatrix}, \tag{4}$$

where $V_i = \Lambda_i^{-1} S_i$.

3. Symmetric decomposition

This section formally defines the symmetries of the KKT system (4) and its symmetric decomposition.

3.1. Definition of symmetry

For this paper, we define symmetries as similarity transformations that map the symmetric part of the linearized KKT system (4) to itself. See Danielson (2021) for details about how these symmetries are related to the symmetries of the MPC (1).

Definition 1. A symmetry of the linearized KKT system (4) is a pair of invertible transformations $\Theta^{U,Y} = I_N \otimes \Theta^{u,y} \in \mathbb{R}^{n^{U,Y} \times n^{U,Y}}$ that satisfy the commutator equations

$$\begin{bmatrix} \Theta^U & 0 \\ 0 & \Theta^Y \end{bmatrix} \begin{bmatrix} H & A^\top \\ A & 0 \end{bmatrix} = \begin{bmatrix} H & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Theta^U & 0 \\ 0 & \Theta^Y \end{bmatrix}. \tag{5}$$

The matrix $V_i = \Lambda_i^{-1} S_i$ in (4) generally does not commute $\Theta^Y V_i \neq V_i \Theta^Y$ since the slacks s and duals λ are not required to be symmetric under our control-oriented definition (Danielson & Borrelli, 2012, 2015b) of symmetry, in contrast to the typical optimization-oriented symmetry.

In this paper, we are interested in symmetries defined in Definition 1 with a particular form (possibly under an appropriate change-of-basis Baker, 2005)

$$\Theta^{u,y} = \begin{bmatrix} \Pi \otimes I_{n_1^{u,y}} & 0 \\ 0 & I_{n_{m+1}^{u,y}} \end{bmatrix} \tag{6}$$

for some m-dimensional permutation matrix $\Pi \in \mathfrak{G}_m$. Note that the form (6) is not restrictive according to Danielson and Borrelli (2014) which showed that the symmetries of an MPC problem (1) are isomorphic to permutations of its constraints (1c). We say the linearized KKT system (4) is invariant under the symmetric-group $\mathfrak{G}_m$ if the commutator (5) holds for all $\Pi \in \mathfrak{G}_m$. The symmetries (6) have a fixed subspace for all $\Pi \in \mathfrak{G}_m$ due to the identity matrix $I_{n_{m+1}^{u,y}}$. The dimension of this fixed-space $n_{m+1}^{u,y}$ relative to the size m of the symmetric-group $\mathfrak{G}_m$ provides a measure of the asymmetry of the linearized KKT system (4), where $n^{u,y} = mn_1^{u,y} + n_{m+1}^{u,y}$. For instance, if $n_{m+1}^{u,y} = n^{u,y}$ then the linearized KKT system (4) is completely asymmetric. This dimension m of the group $\mathfrak{G}_m$ will strongly influence the computational benefits of our symmetric PDIP algorithm.

For the symmetry-group $\mathfrak{G}_m$ with symmetries of the form (5), the symmetric transformations $\Phi^{U,Y} = I_N \otimes \Phi^{u,y}$ where $\Phi^{u,y}$ will have a particular block structure (Danielson, 2021) as

$$\Phi^{u,y} = \begin{bmatrix} \Phi \otimes I_{n_1^{u,y}} & 0 \\ 0 & I_{n_{m+1}^{u,y}} \end{bmatrix}, \tag{7}$$

where the dimensions $n_1^{u,y}$ and $n_{m+1}^{u,y}$ match the dimensions in (6). In general, the orthogonal matrix Φ has the form $\Phi = [N, \ \mathbf{1}]$, where $N \in \mathbb{R}^{m \times (m-1)}$ is the orthogonal complement of the all-ones vector $\mathbf{1} \in \mathbb{R}^m$. In particular, a special case of the generic matrix Φ is

$$\Phi = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ -1 & 1 & \cdots & 1 & 1 \\ 0 & -2 & \cdots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & (1-m) & 1 \end{bmatrix} \Psi \in \mathbb{R}^{m \times m}, \tag{8}$$

where the diagonal matrix $\Psi = [\psi_{ii}]_{m \times m}$ ensures that the matrix Φ has normalized column vectors. The entries of the main diagonal of the matrix Ψ are $\psi_{ii} = 1/\sqrt{i+i^2}$ for $i = 1, 2, \ldots, m-1$ and $\psi_{mm} = 1/\sqrt{m}$. The matrix (8) has advantageous computational properties (detailed in Danielson (2021)) which our algorithm will exploit.

For the KKT system (4), the symmetric decomposition consists of two orthogonal transformations of inputs and constraints $\Phi^{U,Y} \in \mathbb{R}^{n^{U,Y} \times n^{U,Y}}$ that decompose the symmetric part of the KKT coefficient matrix (4) as

$$\begin{bmatrix} \Phi^U & 0 \\ 0 & \Phi^Y \end{bmatrix}^\top \begin{bmatrix} H & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Phi^U & 0 \\ 0 & \Phi^Y \end{bmatrix} \to \bigoplus_{j=1}^{m} \begin{bmatrix} \hat{H}_j & \hat{A}_j^\top \\ \hat{A}_j & 0 \end{bmatrix}, \tag{9}$$

where the left-hand side of (9) is equal to the right-hand side after permuting the rows and columns. The transformations $\Phi^{U,Y} \in \mathbb{R}^{n^{U,Y}, n^{U,Y}}$ are made of tall rectangular matrices $\Phi_j^{U,Y} \in \mathbb{R}^{n^{U,Y} \times n_1^{U,Y}}$ each of which defines the smaller matrix $\hat{H}_j = (\Phi_j^U)^\top H \Phi_j^U \in \mathbb{R}^{n_1^U \times n_1^U}$, and likewise for $\hat{A}_j$.

Throughout this paper, we will use the hat-notation $\hat{\cdot}$ to denote variables represented in the symmetry-adapted basis $\Phi^{U,Y}$. A procedure for computing the decomposition (9) for a generic symmetry group as defined in Definition 1 was presented in Danielson and Bauer (2015), de Klerk, Dobre, and Pasechnik (2011) and Murota, Kanno, Kojima, and Kojima (2010).

For the symmetric-group in Definition 1, the generic decomposition (9) has additional structure, namely that the first m blocks are identical

$$\begin{bmatrix} \hat{H}_j & \hat{A}_j^\top \\ \hat{A}_j & 0 \end{bmatrix} = \begin{bmatrix} \hat{H}_1 & \hat{A}_1^\top \\ \hat{A}_1 & 0 \end{bmatrix}, \tag{10}$$

for $j = 1, \ldots, m$. The explicit repetition (9) of problem-data will be used to reduce the memory required for the symmetric PDIP algorithm. An illustrative example of symmetric decomposition can be found in Danielson (2021).

4. Symmetric PDIP algorithm

This section presents our symmetry exploiting variant of the PDIP Algorithm 1 and compares its computational and memory complexity with the baseline.

Algorithm 2 describes our symmetry exploiting variant of Algorithm 1. Algorithm 2 solves the QP problem (2) by transforming it into the symmetric domain using transformation (7), which can be formulated as

$$\min_{\hat{U}_j, \hat{s}_j} \ \sum_{j=1}^{m} \frac{1}{2} \hat{U}_j^\top \hat{H}_j \hat{U}_j + \hat{f}_j^\top \hat{U}_j, \tag{11a}$$

$$\text{s.t.} \ \ \hat{A}_j \hat{U}_j + \hat{s}_j = \hat{b}_j, \qquad \Phi_j^Y \hat{s}_j \geq 0 \tag{11b}$$

$$\text{for } j = 1, \ldots, m,$$

where $\hat{U}_j = (\Phi_j^U)^\top U$ and $\hat{s}_j = (\Phi_j^Y)^\top s$ are the transformed inputs and slacks trajectories, respectively. The decomposed QP (11) can equivalently be derived by decomposing the MPC problem (1) and

Algorithm 2
Symmetric Primal-Dual Interior-Point

input: U_0, $\lambda_0 \geq 0$, $s_0 \geq 0$, $\sigma \in (0, 1)$, μ_o
Transform to symmetric domain: $\hat{U}_0$,
repeat
1: Average complementarity violation $\mu_i = s_i^\top \lambda_i / n$
Transform to symmetric domain: $\hat{\lambda}$, $\hat{s}$
2: Solve linearized KKT system (12) for direction
Transform to the original domain: $\delta\lambda = \Phi\delta\hat{\lambda}$
3: Slack direction (3)
4: Compute step-sizes:
$\alpha_p = \min(-\kappa / \min(\delta s./s), 1)$
$\alpha_d = \min(-\kappa / \min(\delta\lambda./\lambda), 1)$
for some $\kappa \in (0,\ 1)$
5: Update primal and dual variables:
$\hat{U}^+ = \hat{U} + \alpha_p \delta\hat{U}$
$\lambda^+ = \lambda + \alpha_d \delta\lambda \geq 0$
$s^+ = s + \alpha_d \delta s \geq 0$
until $\sum_{j=1}^m \|\hat{U}_j^+ - \hat{U}_j\|^2 < \epsilon$, $\sum_{j=1}^m \|\lambda_j^+ - \lambda_j\|^2 < \epsilon$, and $\sum_{j=1}^m \|s_j^+ - s_j\|^2 < \epsilon$
output: Back to the original domain: U

forming the equivalent reduced QP. The decomposed QP (11) has the following KKT system

$$\begin{bmatrix} \hat{H} & \hat{A}^\top \\ \hat{A} & -\hat{V}_i \end{bmatrix} \begin{bmatrix} \delta\hat{U} \\ \delta\hat{\lambda} \end{bmatrix} = - \begin{bmatrix} \hat{H}\hat{U}_i + \hat{f} + \hat{A}^\top \hat{\lambda}_i \\ \hat{A}\hat{U}_i - \hat{b} + \sigma\mu_i \hat{\Lambda}_i^{-1}\mathbf{1} \end{bmatrix}, \tag{12}$$

where $\hat{V}_i = (\Phi^Y)^\top \Lambda_i^{-1} S_i \Phi^Y$. The premise of Algorithm 2 is that the symmetric decomposition (9) reduces computational complexity by decomposing the cost (11a) and constraints (11b) of the QP (11). In other words, iteratively solving (12) in Algorithm 2 is less computationally expensive than solving (4) in Algorithm 1. However, there are two challenges: First, since the slacks and duals are not symmetric, the symmetric decomposition transforms the sparse matrix $V_i = \Lambda_i^{-1} S_i$ into the dense matrix $\hat{V}_i = (\Phi^Y)^\top \Lambda_i^{-1} S_i \Phi^Y$. Naively, the increased density of $\hat{V}_i$ negates the computational benefits of the increased sparsity of $\hat{H}$ and $\hat{A}$. Second, the slacks $s_i = \Phi\hat{s}_i \geq 0$ and duals $\lambda_i = \Phi\hat{\lambda}_i \geq 0$ have non-negativity constraints that must be enforced in the original domain. Again, naively, the cost of transforming between domains negates the computational benefits of the symmetric decomposition. In Section 4.2 we will prove that Algorithm 2 has lower computational complexity than Algorithm 1 despite these challenges.

4.1. Equivalence of PDIP algorithms

In this section, we show the symmetric Algorithm 2 has equivalent convergence properties to baseline Algorithm 1 in the sense that they require the same number of iterations to reach the same optimal solution. This is crucial since it means that the convergence proofs and properties (Borrelli et al., 2017) for the baseline Algorithm 1 apply to the decomposed Algorithm 2.

The following lemma shows that both PDIP algorithms produce identical iterates when the MPC problem has a unique solution.

Lemma 1. *Let Assumption 1 hold. Then, the iterates (U_i, s_i, λ_i) and $(\hat{U}_i, \hat{s}_i, \hat{\lambda}_i)$ produced by Algorithms 1 and 2, respectively satisfy $U_i = \Phi^U \hat{U}_i$, $s_i = \Phi^Y \hat{s}_i$, and $\lambda_i = \Phi^Y \hat{\lambda}_i$.*

Proof. Assumption 1 ensures that the MPC problem (1) and thus the reduced QP (2) are strictly convex and therefore has a unique solution. Moreover, since the symmetric transformation (7) does not change the convexity, the transformed QP (11) is also strictly convex and has a unique solution. It follows that the linearized KKT systems (4) and (12) have unique solutions since the respective KKT matrices are strictly positive definite. In addition, transforming the solution of the linearized KKT system (12) back to the original domain produces the same as the solution of the linearized KKT system (4). Thus, in each iteration, both Algorithms 1 and 2 have the same descent directions and step-sizes. ■

Lemma 1 means that Algorithms 1 and 2 produce equivalent iterates. In other words, the change-of-variables Φ does not change the sequence of iterates produced by the PDIP algorithms. This result does not necessarily hold for non-strictly convex MPC problems (1) since the linearized KKT systems (4) and (12) would have multiple solutions. Since pseudo-inverses can depend on the basis, this would produce different descent directions when pseudo-inverting (4) and (12).

The following theorem shows that Algorithms 1 and 2 have the same convergence rates.

Theorem 1. *Let Assumption 1 hold. Then, Algorithms 1 and 2 terminate after the same number of iterations and produce the same optimal solution $U^\star = \Phi^U \hat{U}^\star$.*

Proof. Since Φ is orthogonal, the terminal conditions of Algorithms 1 and 2 are equivalent $\|U^+ - U\| = \|\Phi^U(\hat{U}^+ - \hat{U})\|$ and likewise for s_i and λ_i. Thus, both algorithms terminate after the same number of iterations. According to Lemma 1, solutions are equivalent during this final iteration i.e. $U = \Phi^U \hat{U}$, $s = \Phi^Y \hat{s}$, and $\lambda = \Phi^Y \hat{\lambda}$. ■

Theorem 1 means that Algorithms 1 and 2 require the same number of iterations to converge to the same optimal. However, we will show that Algorithm 2 has lower computational costs since its iterations are cheaper.

4.2. Computational complexity

In this section, we compare the computational complexity of Algorithms 1 and 2. Our analysis assumes that the matrices in (4) are dense.

The premise of Algorithm 2 is that the symmetric decomposition (9) produces a linearized KKT system (12) that has advantageous structure over (4). However, this is non-trivial. While the symmetric decomposition sparsifies the matrices $\hat{H}$ and $\hat{A}$ in (12), it densifies the matrix V_i. The resulting KKT matrix from (12) has the structure

$$\begin{bmatrix} \hat{H}_1 & \hat{A}_1^\top & \cdots & 0 & 0 & 0 & 0 \\ \hat{A}_1 & \hat{V}_{11} & \cdots & 0 & \hat{V}_{1m} & 0 & \hat{V}_{1+} \\ \vdots & \vdots & \ddots & \vdots & & \vdots & \\ 0 & 0 & & \hat{H}_1 & \hat{A}_1^\top & 0 & 0 \\ 0 & \hat{V}_{m1} & \cdots & \hat{A}_1 & \hat{V}_{mm} & 0 & \hat{V}_{m+} \\ 0 & 0 & & 0 & 0 & \hat{H}_+ & \hat{A}_+^\top \\ 0 & \hat{V}_{+1} & \cdots & 0 & \hat{V}_{+m} & \hat{A}_+ & \hat{V}_{++} \end{bmatrix}, \tag{13}$$

where the subscript $+$ is the $m+1$ term. For highly constrained problems $n_y \gg n_u$, the matrix (13) has the same order-density $\mathcal{O}(N^2 n^2)$ as the original KKT system (4) since $V_{ij} \in \mathcal{O}(n^2)$. Thus, naively the decomposed KKT system (12) has the same complexity as the original (4). Fortunately, the following lemma shows that Gauss–Seidel iterations can exploit the advantageous structure of $\hat{H}$ and $\hat{A}$ in the symmetric domain and $V_i = \Lambda_i^{-1} S_i$ in the original domain to reduce computational complexity.

Lemma 2. *Gauss–Seidel iterations have complexity* $\mathcal{O}(N^2(m^2n_1^2 + n_{m+1}^2))$ *for* KKT *system (4) and* $\mathcal{O}(N^2(mn_1^2 + n_{m+1}^2))$ *for* KKT *system (12) where* $n_1 = n_1^u + n_1^y$ *is the dimension of the repeated blocks and* $n_{m+1} = n_{m+1}^u + n_{m+1}^y$ *is the dimension of the non-repeated blocks.*

Proof. Gauss–Seidel solves (12) by iterating

$$\underbrace{\begin{bmatrix}\hat{H}_U & \hat{A}\\ 0 & \hat{V}_i\end{bmatrix}}_{\mathbf{U}}\begin{bmatrix}\hat{U}^{(j+1)}\\ \hat{s}^{(j+1)}\end{bmatrix} = \begin{bmatrix}b_U\\ b_s\end{bmatrix} - \underbrace{\begin{bmatrix}\hat{H}_L & 0\\ \hat{A}^\top & 0\end{bmatrix}}_{\mathbf{L}}\begin{bmatrix}\hat{U}^{(j)}\\ \hat{s}^{(j)}\end{bmatrix}, \tag{14}$$

where $\hat{V}_i$, $b_U = \hat{H}\hat{U}_i + \hat{f} + \hat{A}^\top\hat{\lambda}_i$ and $b_s = \hat{A}\hat{U}_i - \hat{b} + \sigma\mu_i\hat{\Lambda}^{-1}\mathbf{1}$ are fixed for the Gauss–Seidel iterations j. The matrix $\hat{H} = \hat{H}_U + \hat{H}_L$ is decomposed into upper $\hat{H}_U$ and lower $\hat{H}_L$ triangular matrices. During each iteration, Gauss–Seidel inverts the upper-triangular matrix $\mathbf{U}$ via back-substitution.

First, consider the computation of $\hat{s}^{(j+1)} = \hat{V}_i^{-1}(b_s - \hat{A}^\top\hat{U}^{(j)})$. The matrix–vector multiplication $\hat{A}^\top\hat{U}^{(j)}$ has complexity $\mathcal{O}(N^2(mn_1^2 + n_{m+1}^2))$ since $\hat{A}$ is block diagonal with m blocks of dimension $\mathcal{O}(Nn_1 \times Nn_1)$ and 1 block of dimension $\mathcal{O}(Nn_{m+1} \times Nn_{m+1})$. The subtraction $z = b_s - \hat{A}^\top\hat{U}^{(j)}$ has linear complexity $\mathcal{O}(Nn)$. Finally, the matrix–vector multiplication $\hat{s}^{(j+1)} = \hat{V}_i^{-1}z$ with the inverse $\hat{V}_i^{-1} = \Phi^{Y\top}S_i^{-1}\Lambda_i\Phi^Y$ has linear complexity $\mathcal{O}(Nn)$ since S_i and Λ_i are diagonal matrices and the transformation matrix Φ has $\mathcal{O}(Nn)$ complexity according to Lemma 3 in Danielson (2021). Thus, the computational complexity of computing $\hat{s}^{(j+1)}$ is $\mathcal{O}(N^2(mn_1^2 + n_{m+1}^2))$.

Computing $\hat{U}^{(j+1)} = \hat{H}_U^{-1}(b_U - \hat{A}\hat{s}^{(j+1)})$ is dominated by inverting $\hat{H}_U$ via back-substitution which has complexity $\mathcal{O}(N^2(mn_1^2 + n_{m+1}^2))$ since $\hat{H}$ is block-diagonal with the same order as $\hat{A}$. Since computing $\hat{U}^{(j+1)}$ has the same computational complexity as computing $\hat{s}^{(j+1)}$, the overall computational complexity of the Gauss–Seidel iteration (14) is $\mathcal{O}(N^2(mn_1^2 + n_{m+1}^2))$ for (12).

For (4), Gauss–Seidel has computational complexity $\mathcal{O}(N^2n^2) = \mathcal{O}(N^2(m^2n_1^2 + n_{m+1}^2))$ since the matrices are dense. ■

If the dimension n_{m+1} of the non-repeated block is small $n_{m+1} \ll n_1$, then $n^2 = (mn_1 + n_{m+1})^2 \approx m^2n_1^2$ and $n_1 + n_{m+1} \approx n_1$. Thus, Lemma 2 means that the Gauss–Seidel iterations for solving (12) are approximately m-times faster than solving (4) where m is the order of the symmetry group.

Lemma 2 will not hold for a generic matrix decomposition; sparsifying H and A will densify V_i, which will conserve the computational complexity. Algorithm 2 exploits the advantageous structure (diagonal) of V_i in the original domain and structure (block-diagonal) of H and A in the symmetric domain by switching between these domains. The key is that we can perform this transformation cheaply due to the computationally beneficial properties of the matrix (8).

The final complication is that Algorithm 2 requires the additional steps of transforming the slacks $\hat{s}$ and duals $\hat{\lambda}$ back into the original domain to enforce primal $s = \Phi^Y\hat{s} \geq 0$ and dual $\lambda = \Phi^Y\hat{\lambda} \geq 0$ feasibility, respectively. The following theorem shows that Algorithms 2 can exploit advantageous properties of Φ to reduce computational complexity.

Theorem 2. *Let* M *be the number of Gauss–Seidel iterations for solving* KKT *systems (4) or (12). Then, each iteration of Algorithms 1 and 2 have the worst-case computational complexities* $\mathcal{O}(MN^2(m^2n_1^2 + n_{m+1}^2))$ *and* $\mathcal{O}(MN^2(mn_1^2 + n_{m+1}^2))$*, respectively, where* $n_1 = n_1^u + n_1^y$ *is the dimension of the repeated blocks and* $n_{m+1} = n_{m+1}^u + n_{m+1}^y$ *is the dimension of the non-repeated blocks.*

Proof. Algorithm 1 is asymptotically dominated by solving step 2 since all other steps only involve vector manipulations. Thus, according to Lemma 2 the computational complexity of Algorithm 1 is $\mathcal{O}(M_1N^2(m^2n_1^2 + n_{m+1}^2))$ where M_1 is the number of Gauss–Seidel iterations needed to solve (4). Algorithm 2 has additional matrix manipulations to transform the slack $\hat{s}$ and dual $\hat{\lambda}$ variables back into the original domain to enforce primal $s = \Phi^Y\hat{s} \geq 0$ and dual $\lambda = \Phi^Y\hat{\lambda} \geq 0$ feasibility. According to Lemma 3 in Danielson (2021), the computational cost of this transformation is linear $\mathcal{O}(Nn)$. Thus, Algorithm 2 is dominated by step 2 which has computational complexity $\mathcal{O}(M_2N^2(mn_1^2 + n_{m+1}^2))$ according to Lemma 2.

Finally, by similar uniqueness arguments as Theorem 1, we can show $M_1 = M_2$. ■

Theorem 2 shows that the computational cost of each iteration of the symmetric Algorithm 2 grows linearly with the repetition m rather than the quadratic growth of the baseline Algorithm 1. Since Theorem 1 shows that these algorithms converge after the same number of iterations, this means that Algorithm 2 is asymptotically m times faster than Algorithm 1.

4.3. Memory complexity

In this section, we compare the worst-case memory complexities of Algorithms 1 and 2. Again, our worst-case analysis assumes that the cost and state-space matrices in the MPC problem (1) are dense. The following lemma shows the memory complexity of storing the KKT systems (4) and (12).

Lemma 3. *Storing the* KKT *system (4) and (12) has worst-case memory complexity* $\mathcal{O}(N^2(mn_1 + n_{m+1})^2)$ *and* $\mathcal{O}(N^2(n_1 + n_{m+1})^2)$*, respectively, where* $n_1 = n_1^u + n_1^y$ *is the dimension of the repeated blocks and* $n_{m+1} = n_{m+1}^u + n_{m+1}^y$ *is the dimension of the non-repeated blocks.*

Proof. Storing the KKT system (4) requires storing the matrices $H, A \in \mathcal{O}(N^2n^2)$ and $V_i \in \mathcal{O}(Nn)$ where $\mathcal{O}(N^2n^2)$ dominates the $\mathcal{O}(Nn)$ complexity required to store the diagonal matrix V_i. Thus, storing (4) requires $\mathcal{O}(N^2(mn_1 + n_{m+1})^2)$ memory where $n = mn_1 + n_{m+1}$.

Likewise, storing the KKT system (12), requires storing the matrices $\hat{H}, \hat{A} \in \mathcal{O}(N^2n^2)$ and $\hat{V}_i \in \mathcal{O}(N^2n^2)$. First, note that we can store $\hat{V}_i$ as $\hat{V}_i = \Phi_Y^\top\Lambda_i^{-1}S_i\Phi_Y$ where $\Lambda_i^{-1}S_i \in \mathcal{O}(Nn)$ since Λ_i and S_i are diagonal. Thus, the memory complexity is dominated by storing $\hat{H}, \hat{A} \in \mathcal{O}(N^2n^2)$. Exploiting the m repetition (10), we only need to store the first $\hat{H}_1, \hat{A}_1 \in \mathcal{O}(N^2n_1^2)$ and last $\hat{H}_{m+1}, \hat{A}_{m+1} \in \mathcal{O}(N^2n_{m+1}^2)$ blocks of the block-diagonal matrices $\hat{H}$ and $\hat{A}$. Hence the memory complexity of storing these matrices is $\mathcal{O}(N^2(n_1^2 + n_{m+1}^2)) = \mathcal{O}(N^2(n_1 + n_{m+1})^2)$. ■

Lemma 3 says that the amount of read–write memory required to store KKT-system (4) grows quadratically with the number of repetitions m. In contrast, storing (12) requires constant memory regardless of m since we do not store multiple copies of the repeated matrix (10). This reflects our intuition that adding identical components to a problem does not require additional data to describe. The worst-case memory complexities of Algorithms 1 and 2 are summarized by the theorem below.

Theorem 3. *Algorithms 1 and 2 have asymptotic* $m \to \infty$ *worst-case memory complexities* $\mathcal{O}(N^2n^2)$ *and* $\mathcal{O}(Nn)$*, respectively, where* $n = n_u + n_y$.

Proof. For Algorithms 1, the memory complexity is dominated by step 4 since all other steps involve vector manipulations with complexity $\mathcal{O}(Nn)$. Thus, the memory complexity is $\mathcal{O}(N^2(mn_1 + n_{m+1})^2) = \mathcal{O}(N^2n^2)$ according to Lemma 3.

According to Lemma 3, step 5 of Algorithms 2 has constant memory complexity $\mathcal{O}(N^2(n_1+n_{m+1}))$ with respect to the number m of repetitions. Thus, the vector manipulations with complexity $\mathcal{O}(N(mn_1 + n_{m+1})) = \mathcal{O}(Nn)$ dominate the computational complexity. ■

Although Theorem 3 says that storing the vectors $\hat{U} \in \mathbb{R}^{N(mn_1^u+n_{m+1}^u)}$, and $s, \lambda \in \mathbb{R}^{N(mn_1^y+n_{m+1}^y)}$ asymptotically $m \to \infty$ dominates the memory requirements for Algorithm 2, in practice the constant memory requirement $\mathcal{O}(N^2(n_1 + n_{m+1}))$ for storing the KKT system (12) dominates. This will be verified in our case study.

5. Case study: Heating ventilation and air conditioning

In this section, we demonstrate our symmetry exploiting PDIP algorithm for a HVAC case study.

To examine the performance of Algorithm 2 and compare its results with Algorithm 1, we apply both algorithms to an HVAC control problem. The HVAC is an ideal choice of illustrative example since it has heavy computational cost as well as a high-degree of symmetry (Danielson, 2017). Among different types of controllers, MPC is a favorable option for HVAC problems since it provides both optimal energy efficiency and safe operation of the equipment by enforcing proper constraints (Wen & Mishra, 2018).

This HVAC system is a multi-evaporator vapor compression system (ME-VCS) which is operated in heating mode with one outdoor unit connected to m identical indoor units (Bortoff et al., 2018, 2022; Burns et al., 2017). The outdoor unit and each indoor units have 2 control inputs. Thus, we have $n^u = 2m + 2$. The valve position and indoor-fan speed are the 2 control inputs of the indoor units, and the compressor speed and outdoor-fan speed are the 2 control inputs of the outdoor unit. Each indoor unit contributes 4 coupled, non-physical states to the model, while the outdoor unit contributes 8 non-physical states, which implies that $n^x = 4m + 8$. The 8 non-physical states correspond to the states of the model proposed in He, Liu, and Asada (1997), where three of the states represent the pressure, enthalpy, and phase boundary of the refrigerant, and the fourth state represents the thermal-zone temperature.

Each of the control inputs has lower and upper bounds (1c) on its operation. In addition, there is a lower-bound on the compressor inlet-temperature to ensure only super-heated gas refrigerant enters. There is an upper-bound of the compressor outlet-temperature to prevent overheating. And there is a lower-bound on the evaporator inlet-temperature to prevent excessive frost formation, which implies that the outdoor unit has 3 constraints. The operational constraints of the inputs along with constraints form the constraints (1c) with $n^y = 2m+5$. See Burns et al. (2017) for details about the constraints.

The cost function has the form (1a) which penalizes the temperature tracking error and energy consumption. $\mathcal{Q} = \mathcal{C}_r^\top \mathcal{Q}_r \mathcal{C}_r$ corresponds to the temperature tracking error of the m thermal zones, where the tracked-outputs $\mathcal{C}_r x \in \mathbb{R}^m$ are the temperatures of the m thermal-zones, and $\mathcal{Q}_r$ is a design parameter. The matrix $\mathcal{R} \in \mathbb{R}^{(2m+2)\times(2m+2)}$ is diagonal with a large penalty on the compressor speed, medium penalties on the (indoor and outdoor) fan speeds, and small penalties on the valve positions, which reflects the relative power usage of these components. There is no coupling between the state and input costs i.e. $\mathcal{S} = 0 \in \mathbb{R}^{(4m+8)\times(2m+2)}$. All matrices $\mathcal{Q}_r$, $\mathcal{R}$, and $\mathcal{S}$ were designed such that Definition 1 holds. Considering the fastest and slowest time-scale of the HVAC system (i.e. respectively the refrigerant thermo-fluidics and the thermal-zones temperature), the MPC problem (1) is solved every 1 min with a prediction horizon of $N = 30$ min.

For $i = m + 1$, the fixed-space dynamics have $\hat{n}_{m+1}^u = 2 + 2$ inputs, $\hat{n}_{m+1}^x = 8 + 4$ states, and $\hat{n}_{m+1}^y = 5 + 2$ constraints. Thus, the dimension of the non-repeated block is $n_{m+1} = 11$. The $m-1$ identical subsystems have $\hat{n}_1^u = 2$ inputs, $\hat{n}_1^x = 4$ states, and $\hat{n}_1^y = 2$ constraints. Thus, the dimension of the repeated blocks is $n_1 = 4$. Intuitively, these subsystems model the deviation of the ith indoor unit from the aggregate dynamics of the remaining $m-i-1$ indoor units. More details about the symmetric decomposition (9) of HVAC systems can be found in Danielson (2017). The robustness of the symmetric decomposition to asymmetry is discussed in Chuang, Danielson, and Borrelli (2015) and Danielson and Bauer (2015).

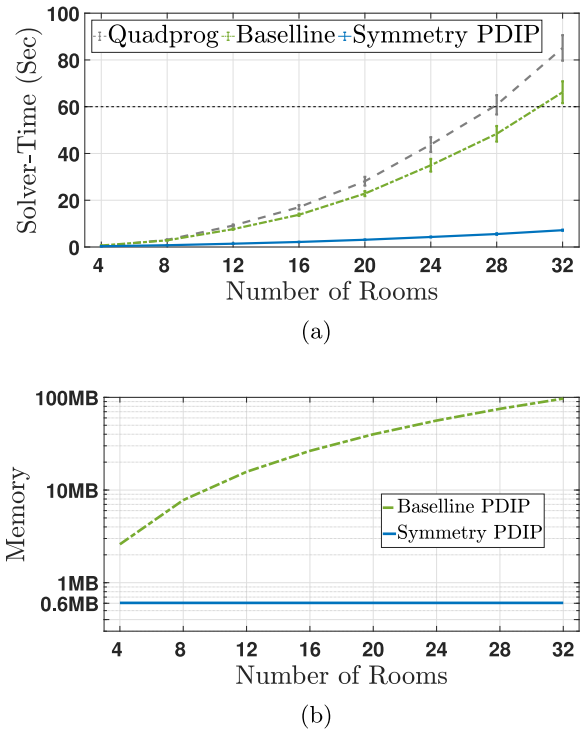


Fig. 1. Comparison of the two implementations of the PDIP algorithms (baseline and symmetric PDIP) for the HVAC control problem. The green (dash-dotted) and blue (solid) lines correspond to baseline and symmetric PDIP. (a) Comparison of the computation-time (second). For comparison, the computation-time of MATLAB quadprog is also included, which is shown by the gray (dashed) line. (b) Comparison of the required memory (megabytes).

In this example, the QPs (2) and (11) were solved for 50 random initial conditions by Algorithm 1 and Algorithm 2 in MATLAB using a single-core. The number of thermal-zones m varied from $m = 4$ zones to $m = 32$ zones. The performance of the Algorithm 1 and Algorithm 2 is shown in Fig. 1.

Fig. 1(a) shows computation-time for Algorithms 1 and 2, and MATLAB's quadprog using its interior-point method. The symmetric PDIP Algorithm 2 is significantly faster than the baseline Algorithm 1 and MATLAB's quadprog, as shown in Fig. 1(a). This empirically verifies Theorem 2, which predicted the sub-quadratic growth of the symmetric PDIP Algorithm 2 and quadratic growth of the baseline PDIP Algorithm 1 with respect to the number m of indoor units. The more essential point is that the baseline PDIP method is not applicable for real-time implementation of MPC problem (1). For instance, in the HVAC systems with $m \geq 32$ indoor units, both the baseline PDIP Algorithm 1 and MATLAB's quadprog required more than 1-min sample time to solve the MPC problem (1) for at least one random trial, as shown in Fig. 1.

The memory benefits of the symmetric PDIP Algorithm 2 over the baseline Algorithm 1 are equally impressive. Fig. 1(b) shows the read–write memory required to store the KKT matrix for each of the algorithms. The green dash-dotted line corresponds to the

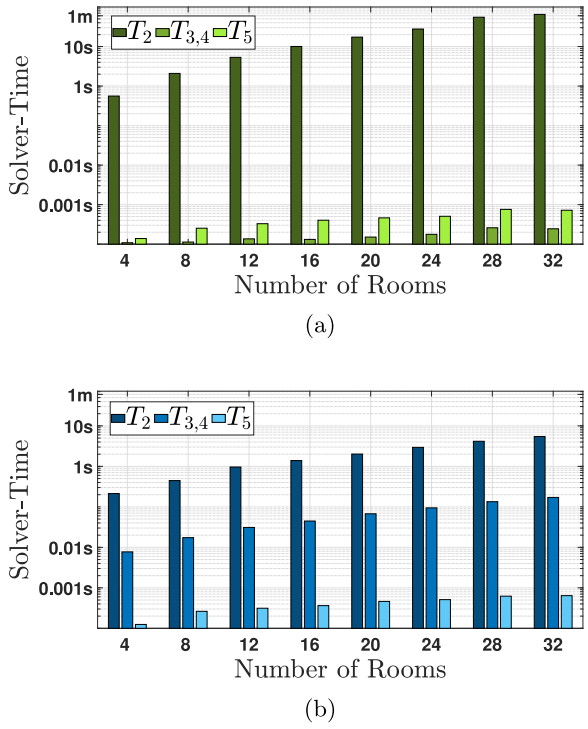


Fig. 2. Comparison between the computation-time of the main steps (2), (3,4), and (5), in the implementations of (a) the baseline PDIP algorithm and (b) the symmetric PDIP algorithm for the HVAC control problem. The computation-time includes T_2 for solving the linearized KKT system, $T_{3,4}$ for computing the slack direction and step-size, and T_5 for updating the decision variables.

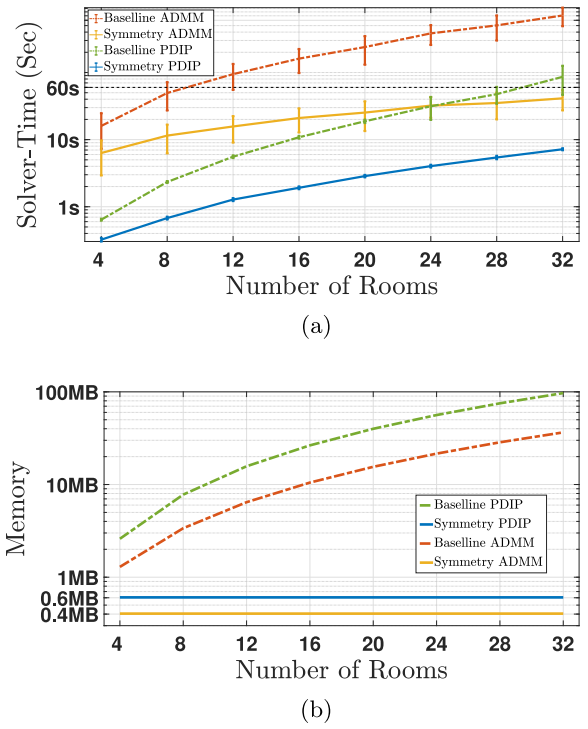


Fig. 3. (a) Solver time (second) (b) required memory (megabytes) of the baseline and symmetric ADMM and PDIP algorithms for the case study HVAC control problem. The red (dash-dotted) and yellow (solid) lines correspond to baseline and symmetric ADMM. The green (dash-dotted) and blue (solid) lines correspond to baseline and symmetric PDIP.

KKT matrix (4) of the baseline Algorithm 1 and the blue line corresponds to the KKT matrix (12) of the symmetric Algorithm 2. Considering the symmetric algorithm, the memory required to store the $i = 1$ and $i = m$ blocks is constant regardless of the number of indoor units m. Whereas the memory required to store the KKT matrix (4) for the baseline Algorithm 1 grows quadratically with the number of indoor units m. This empirically verifies Theorem 3. For the largest number of indoor units $m = 32$, the baseline Algorithm 1 required nearly 100 Mb of memory, whereas the symmetric PDIP Algorithm 2 required less than 600 kb. This is important since the computational benefits of IP algorithms often come at the cost of increased memory requirements over e.g. first-order methods. Note that the amount of read–write memory required to store the signals $x_k \in \mathbb{R}^{4m+8}$, $u_k \in \mathbb{R}^{2m+2}$, and $y_k \in \mathbb{R}^{2m+5}$ for $k = 1, \ldots, N$ is not shown since this is the same for both algorithms.

Fig. 2 shows the computation-time for the main steps 2-5 of Algorithms 1 and 2. T_2 is the computation-time for step 2 in Algorithms 1 and 2 in which the respective linearized KKT systems (4) and (12) are solved. The computation-time T_2 dominates the time spent on the other steps, which empirically supports the opening argument in the proof of Theorem 2. We note that the computation-time T_2 in Algorithm 2 (dark blue) is shorter than that in Algorithm 1 (dark green), which empirically verifies the results of Lemma 2. For Algorithm 1, $T_{3,4}$ is the time spent in steps 3-4 to calculate the slack direction and step-size. For Algorithm 2, $T_{3,4}$ also includes the time required to transform the dual variable λ from the symmetric to the original domains. Thus, Algorithm 2 (medium blue) has a longer computation-time $T_{3,4}$ than Algorithm 1 (medium green), although only by a constant factor that does not grow with problem-size m. This is consistent with Lemma 3 in Danielson (2021). The time T_5 spent on updating the primal and dual-variables in step 5 has the same order in Algorithms 1 (light green) and 2 (light blue).

Fig. 3 compares the performance of the PDIP algorithm with the ADMM presented in Danielson (2021). Fig. 3(a) compares the computational-time for these algorithms. Fig. 3(a) shows that the symmetric PDIP algorithm outperformed the symmetric ADMM algorithm in terms of computation-time for this HVAC case study. However, Fig. 3(b) shows that the symmetric ADMM algorithm has lower memory requirements. Thus, either of these algorithms could be preferable depending on the availability of computational-power versus memory for a particular application.

6. Conclusion

This paper presented a symmetry exploiting PDIP algorithm for extreme-scale MPC problems (1). The PDIP algorithm exploited the symmetric decomposition to introduce the advantageous structure of the KKT conditions of the QP. Gauss–Seidel iterations can exploit advantageous structure in both the symmetric and original domains to reduce computation complexity. Furthermore, the symmetric decomposition reduces the read–write memory requirements for the PDIP algorithm enabling its use on platforms with severe memory limitations. The symmetric PDIP Algorithm 2 was applied to an HVAC control problem. The numerical results of this example empirically confirm the theory and demonstrated that the computation time and memory usage of Algorithm 2 are significantly lower than Algorithm 1.

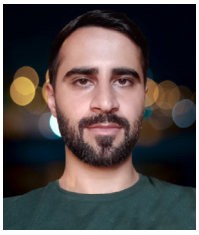
References

Baker, A. (2005). *Representations of finite groups*. University of Glasgow.

Berberich, J., Köhler, J., Müller, M. A., & Allgöwer, F. (2022). Linear tracking MPC for nonlinear systems—Part I: The model-based case. *IEEE Transactions on Automatic Control*, *67*(9), 4390–4405.

Bödi, R., Grundhöfer, T., & Herr, K. (2011). Symmetries of linear programs. *Note di Matematica*, *30*(1), 129–132.
Borrelli, F., Bemporad, A., & Morari, M. (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press.
Bortoff, S. A., Burns, D. J., Laughman, C. R., Qiao, H., Danielson, C., Goldsmith, A., et al. (2018). Power optimizing control of multi-zone heat pumps. In *2018 IEEE conference on control technology and applications (CCTA)* (pp. 826–833). IEEE.
Bortoff, S. A., Schwerdtner, P., Danielson, C., Cairano, S. D., & Burns, D. J. (2022). H-infinity loop-shaped model predictive control with HVAC application. *IEEE Transactions on Control Systems Technology*, *30*(5), 2188–2203. http://dx.doi.org/10.1109/TCST.2022.3141937.
Boyd, S., Diaconis, P., Parrilo, P., & Xiao, L. (2009). Fastest mixing Markov chain on graphs with symmetries. *SIAM Journal on Optimization*, *20*(2), 792–819.
Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.
Burns, D. J., Danielson, C., Zhou, J., & Di Cairano, S. (2017). Reconfigurable model predictive control for multievaporator vapor compression systems. *IEEE Transactions on Control Systems Technology*, *26*(3), 984–1000.
Camacho, E. F., & Alba, C. B. (2013). *Model predictive control*. Springer science & business media.
Chuang, F., Danielson, C., & Borrelli, F. (2015). Robust approximate symmetric model predictive control. In *Conference on decision and control*.
Cogill, R., Lall, S., & Parrilo, P. A. (2008). Structured semidefinite programs for the control of symmetric systems. *Automatica*, *44*(5), 1411–1417.
Danielson, C. (2017). Symmetric control design for multi-evaporator vapor compression systems. In *Dynamic systems and control conference, Vol. 58271*. American Society of Mechanical Engineers, Article V001T03A002.
Danielson, C. (2021). An alternating direction method of multipliers algorithm for symmetric model predictive control. *Optimal Control Applications & Methods*, *42*(1), 236–260.
Danielson, C., & Bauer, S. (2015). Numerical decomposition of symmetric linear systems. In *2015 54th IEEE conference on decision and control (CDC)* (pp. 2061–2066). IEEE.
Danielson, C., & Borrelli, F. (2012). Symmetric explicit model predictive control. In *IFAC nonlinear mpc conference*. IFAC Nonlinear MPC Conference.
Danielson, C., & Borrelli, F. (2014). Identification of the symmetries of linear systems with polytopic constraints. In *American control conference*.
Danielson, C., & Borrelli, F. (2015a). Symmetric constrained optimal control. In *IFAC nonlinear mpc conference*.
Danielson, C., & Borrelli, F. (2015b). Symmetric linear model predictive control. *IEEE Transactions on Automatic Control*, *60*(5), 1244–1259.
Danielson, C., Borrelli, F., Oliver, D., Anderson, D., Kuang, M., & Phillips, T. (2012). Balancing of battery networks via constrained optimal control. In *2012 American control conference (ACC)* (pp. 4293–4298). http://dx.doi.org/10.1109/ACC.2012.6315251.
Danielson, C., Borrelli, F., Oliver, D., Anderson, D., & Phillips, T. (2013). Constrained flow control in storage networks: Capacity maximization and balancing. *Automatica*.
Danielson, C., & Di Cairano, S. (2015). Reduced complexity control design for symmetric LPV systems. In *2015 54th IEEE conference on decision and control (CDC)* (pp. 72–77). IEEE.
He, X.-D., Liu, S., & Asada, H. H. (1997). Modeling of Vapor Compression Cycles for Multivariable Feedback Control of HVAC Systems. *Journal of Dynamic Systems, Measurement, and Control*, *119*(2), 183–191.
de Klerk, E., Dobre, C., & Ṗasechnik, D. V. (2011). Numerical block diagonalization of matrix*-algebras with application to semidefinite programming. *Mathematical Programming*, *129*(1), 91–111.
Luus, R. (2019). *Iterative dynamic programming*. Chapman and Hall/CRC.
Margot, F. (2009). Symmetry in integer linear programming. In *50 years of integer programming 1958-2008: from the early years to the state-of-the-art* (pp. 647–686). Springer.
Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, *2*(4), 575–601.
Messerer, F., Baumgärtner, K., & Diehl, M. (2021). Survey of sequential convex programming and generalized Gauss-Newton methods. *ESAIM: Proceedings and Surveys*, *71*, 64–88.
Murota, K., Kanno, Y., Kojima, M., & Kojima, S. (2010). A numerical algorithm for block-diagonal decomposition of matrix *-algebras with application to semidefinite programming. *Japan Journal of Industrial and Applied Mathematics*, *27*(1), 125–160.
Preindl, M., Danielson, C., & Borrelli, F. (2013). Performance evaluation of battery balancing hardware. In *2013 European control conference (ECC)* (pp. 4065–4070). IEEE.
Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, *11*(7), 733–764.
Vallentin, F. (2009). Symmetry in semidefinite programs. *Linear Algebra and its Applications*, *430*(1), 360–369.
Wen, J. T., & Mishra, S. (2018). Intelligent building control systems. In *Advances in industrial control, A survey of modern building control and sensing strategies*. Springer.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.
Wright, S. J. (1997). *Primal-dual interior-point methods*. SIAM.

Shirin Panahi received her BS degree in Electrical Engineering at Sadjad University of Technology in 2014, M.S. in 2016, and Ph.D. in 2020 both in bioelectrical engineering at Amirkabir University of Technology. She is correntlly a postdoctoral scholar in Engineering at the University of New Mexico. Her current research interests include complex dynamical networks, nonlinear dynamics, chaotic biological modeling, neuroscience, and neural network.



Ali Kashani received a bachelor's degree in electrical engineering from the University of Shiraz in 2014, a master's degree in electrical engineering from the University of Tehran in 2017, and is currently a Ph.D. student in Engineering at the University of New Mexico.

Dr. Claus Danielson joined the Department of Mechanical Engineering at the University of New Mexico as an Assistant Professor in August 2020. From 2014 to 2020, he was a Principal Research Scientist at Mitsubishi Electric Research Laboratories in Cambridge, MA. He received his Doctorate in 2014 from the Model Predictive Control Laboratory at the University of California, Berkeley. He received his masters and Bachelor of Science in mechanical engineering from Rensselaer Polytechnic Institute and the University of Washington, respectively. His research interests are in constrained planning and control. His specialty is developing methods for exploiting structure in extreme-scale or complex planning, control, and optimization problems. He has applied his research to a variety of fields that include autonomous vehicles, robotics, spacecraft guidance and control, heating ventilation and air conditioning, energy storage networks, adaptive optics, atomic force microscopy, and cancer treatment.