IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Novel Matrix Decomposition for Fast and Scalable Model Predictive Control

Muhammad Adil, Raman Goyal and Saman Mostafavi

Abstract— **This paper presents an inverse-free algorithm that exploits the inherent structural sparsity of a model predictive control problem. The scalability associated with large problem sizes and time horizons is one of the major obstacles in model predictive control applications. We address scalability issues by proposing a novel matrix decomposition technique, coupled with a first-order method, to efficiently solve predictive control problems. This approach solves the system of linear equations in an inverse-free manner. The iterative steps of the proposed approach are computationally efficient, require less memory, and can be easily warm-started based on the solution of the previous horizon. We evaluate the performance of the proposed approach on benchmark model predictive control problems, demonstrating its computational advantages over other state-of-the-art algorithms.**

Index Terms— **Model Predictive Control, Optimization, First-order Method, Matrix Decomposition, Inverse-free Algorithm.**

I. Introduction

Model predictive control (MPC) has become a widely used control framework in various industrial processes [1]. It offers a powerful tool for online multivariate nonlinear optimization problems that involve state and input constraints. With recent advances in processing, storage, and fast data transfer, MPC can now tackle problems with rapidly changing system dynamics, such as robotics, building HVAC control, and trajectory optimization. Nevertheless, scalability remains a significant challenge for MPC, particularly in problems that involve large prediction horizons and system dimensions.

To address the scalability challenge, many MPC problems are formulated as quadratic programming (QP) problems [2], [3], and both dense and sparse QP formulations are commonly used [4]. However, solving large sparse QPs remains a computational challenge due to the large prediction horizons or large system dimensions, or both [5]. To overcome this challenge, several numerical approaches and sparsity exploitation techniques have been proposed to efficiently solve the MPC problem [5].

Interior-point methods (IPMs) are well-suited for exploiting the sparsity structure of the sparse quadratic formulation. However, they require a warm start and can be computationally expensive due to the large system of linear equations that need to be solved [6]. On the other hand, first-order operator splitting methods have emerged as competitive alternatives that use gradient information to find the search direction without computing the second derivatives [7]–[12]. While they have lower accuracy, first-order approaches can be easily warm-started and implemented on parallel computing platforms.

However, one of the major challenges in large-scale real-time MPC is solving the sparse linear programming problem in each iteration of interior-point and first-order methods, which makes computational complexity dependent on the formulation structure [13], [14]. Recent approaches have attempted to overcome this challenge, such as the moving block or block-wise Cholesky factorization approach [15], [16] and the Riccati recursion method [17], [18]. Nonetheless, solving a considerable-sized MPC problem in real-time remains a challenging endeavor.

A. Contributions

We propose a novel approach to solve model predictive control (MPC) problems using a standard quadratic programming formulation. Our method is based on a factorization technique that exploits the inherent sparsity structure of the problem, enabling efficient computation of the linear system of equations. Specifically, we reformulate the iterative updates for primal and dual variables of the algorithm to involve only arithmetic operations on matrices and vectors.

In addition, we use a warm start technique that seeds the solution obtained in the previous step of the finite horizon problem to start the next steps. This approach significantly speeds up the computational time of solving MPC by an order of magnitude compared to the commonly used ADMM formulation. Furthermore, the iterative steps of our algorithm are easy to compute and do not involve complex and expensive matrix inversions, making it suitable for real-time implementation on embedded devices with limited memory and computational power. To demonstrate the effectiveness of our approach, we present experimental results on benchmark MPC problems and compare our method against state-of-the-art techniques.

The rest of the paper is structured as follows: Section II introduces the basics of MPC and quadratic programming optimization problems. In Section III, we provide a detailed explanation of our novel factorization technique, including the iterative updates for primal and dual variables and the warm start technique. Section IV presents the experimental results to validate our claims, and finally, in Section V, we conclude the paper and discuss future directions of research.

M. Adil, R. Goyal, and S. Mostafavi are with Palo Alto Research Center - A Xerox Company, Palo Alto, CA, USA. muhammad.adil@mavs.uta.edu, {rgoyal, smostafa}@parc.com,

B. Notations

In this work, we utilize the following notation conventions: matrices are indicated by bold uppercase letters, while bold lowercase letters denote vectors. The ℓ_n norm of a matrix or vector is represented by $\|\cdot\|n$, with the value of n dependent on the context. The transpose operator is denoted by $(\cdot)^\top$. The $n \times n$ identity matrix is denoted by $\boldsymbol{I}_n$. The optimal solution of an optimization problem is represented by the superscript $(\cdot)^*$. The augmented Lagrangian function is denoted by $\mathcal{L}$. Furthermore, positive semidefinite and positive definite matrices are denoted by $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$, respectively.

II. Problem Formulation

We consider a discrete-time system given as:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k,\\ \mathbf{y}_k &= \mathbf{C}_k\mathbf{x}_k,\end{aligned}$$

where $\mathbf{A}_k, \mathbf{B}_k$ and $\mathbf{C}_k$ are known time-varying system matrices and $\mathbf{x}_k \in \mathbb{R}^{n_x}$ and $\mathbf{u}_k \in \mathbb{R}^{u_n}$ are state and control input vectors, respectively. The aim is to find the optimal control sequence $\mathbf{u}^*_k$ at each time instance by using the explicit model of the process to predict future behavior along a horizon of length N and bring the system states $\mathbf{x}$ to the desired reference state $\mathbf{x}_r$. We consider MPC formulation of a linear time-varying dynamical system with quadratic cost, i.e., a Quadratic Programming (QP) problem, with the objective function of the form:

$$\begin{aligned}\mathbf{J}_k := \sum_{k=0}^{N-1} &\left(\mathbf{x}_k - \mathbf{x}_r\right)^\top \mathbf{Q}_k \left(\mathbf{x}_k - \mathbf{x}_r\right) + \mathbf{u}_k^\top \mathbf{R}_k \mathbf{u}_k\\ &+ \left(\mathbf{x}_N - \mathbf{x}_r\right)^\top \mathbf{Q}_N \left(\mathbf{x}_N - \mathbf{x}_r\right).\end{aligned}$$

Let us write the final MPC problem as the following optimization problem:

$$\begin{aligned}\underset{\mathbf{u}_k}{\text{minimize}} \quad & \mathbf{J}_k(\mathbf{x}_k, \mathbf{u}_k), && \text{(1a)}\\ \text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k, && \text{(1b)}\\ & \mathbf{x}_{\min} \le \mathbf{x}_k \le \mathbf{x}_{\max}, && \text{(1c)}\\ & \mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}, && \text{(1d)}\\ & \mathbf{x}_0 = \mathbf{x}(0). && \text{(1e)}\end{aligned}$$

The states and control input of the system are subject to polyhedral constraints and are bound by their respective minimum and maximum values. The matrices $\mathbf{Q}_k \in \mathbb{S}^{n_x}_+$ and $\mathbf{R}_k \in \mathbb{S}^{n_x}_+$ are the weights corresponding to the state and control costs at each time step, respectively, and $\mathbf{Q}_N \in \mathbb{S}^{n_x}_+$ is weight corresponding to the terminal stage cost. The objective is to find the optimal decision variables $\mathbf{u}_k$ at each time step to drive the system states to the desirable state $\mathbf{x}_r$. Similar to the receding horizon strategy, only the first element of optimal control sequence $\mathbf{u}^*$ is used to find the next control sequence at stage $k+1$.

It is desirable to formulate an MPC problem as a box-constrained quadratic programming problem as this allows for using fast first-order solvers for (embedded) online MPC applications. Specifically, various solvers for online MPC are developed on the basis of the Alternating Direction Method of Multiplier (ADMM). Here, we formulate the MPC problem as a standard quadratic problem where we minimize a quadratic objective function subject to a set of linear equality and inequality constraints. Such a formulation can be obtained by introducing the slack variable for corresponding lower and upper bound constraints (1c) and (1d) for states and inputs of the system as:

$$\begin{aligned}\underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{z}^\top\mathbf{P}\mathbf{z} + \mathbf{q}^\top\mathbf{z}, && \text{(2a)}\\ \text{subject to} \quad & \mathbf{H}\mathbf{z} + \mathbf{s} = \mathbf{b}, && \text{(2b)}\\ & \mathbf{s} \ge 0, && \text{(2c)}\end{aligned}$$

where the variable $\mathbf{z} \in \mathbb{R}^n$ is the new unknown variable, $\mathbf{P} \in \mathbb{S}^n_+$ and $\mathbf{q} \in \mathbb{R}^n$ are corresponding objective function costs. The compact constraint matrix $\mathbf{H} \in \mathbb{R}^{m\times n}$ and slack variable $\mathbf{s} \in \mathbb{R}^m$ are also defined, where the first n elements of vector $\mathbf{s}$ are equal to zero to enforce the equality constraint (1b) and ensure standard quadratic programming. In the Appendix, a detailed explanation is given for the transformation of variables from (1) to (2), where n is defined as $(N+1)n_x$ and m is defined as $n + Nn_u$.

In the next section, we briefly discuss the common numerical methods to solve the problem formulated in Eqn. (2) and then we will introduce the novel decomposition to remedy the shortcomings of existing approaches.

III. Novel UV Factorization for Large Scale MPC Problem

Interior point and active set methods have traditionally been the go-to solutions for solving MPC problems. However, the recent popularity of first-order methods is attributable to their scalability and reduced computational cost. The main computational challenge of first-order methods is solving the linear system of equations or KKT matrix. To address this challenge, various direct and indirect approaches have been adopted. Direct approaches compute the factors in the first iteration and reuse them in subsequent iterations. The well-known $\mathbf{LDL}^\top$ decomposition is a commonly used direct approach in such applications. However, for large-scale problems, direct methods become prohibitive and indirect methods, such as Preconditioned Conjugate Gradient (PCG) methods, are more applicable and scalable in practice. While indirect methods provide an approximate solution, they compromise the quality of the solution. In this study, we develop an efficient first-order method in combination with a novel $\mathbf{UV}$ factorization method to increase the scalability and reduce the computational complexity of solving large-scale MPC problems.

A. Decomposition of constraint matrix

Now, we develop a computationally efficient method to solve the real-time MPC problem for large-scale finite horizon systems. We first decompose the constraint matrix $\mathbf{H} = \mathbf{UV} \in \mathbb{R}^{m\times n}$ into two factors ($\mathbf{U} \in \mathbb{R}^{m\times o}, \mathbf{V} \in \mathbb{R}^{o\times n}$) in such a way that $\mathbf{UU}^\top$ and $\mathbf{V}^\top\mathbf{V}$ are both diagonal. The symbol o represents the number of non-zero elements of $\mathbf{H}$.

The factor $\mathbf{U}$ contains a single non-zero element in each column, whereas $\mathbf{V}$ contains a single non-zero entry in each row. These factors can be computed as follows:

$$\mathbf{U} \triangleq \sum_{k=1}^{o} H_{i_k,j_k} \mathbf{e}_{i_k} \mathbf{f}_k^\top, \tag{3}$$

$$\mathbf{V} \triangleq \sum_{k=1}^{o} \mathbf{f}_k \mathbf{g}_{j_k}^\top, \tag{4}$$

where $\mathbf{e}_k{}_{k=1}^{m} \in \mathbb{R}^m$, $\mathbf{f}_k{}_{k=1}^{o} \in \mathbb{R}^o$ and $\mathbf{g}_k{}_{k=1}^{n} \in \mathbb{R}^n$ are standard basis and H_{i_k,j_k} are the non-zero elements of $\mathbf{H}$. More details and illustrative examples of this decomposition can be found in [19], [20].

Based on the $\mathbf{UV}$ decomposition, the MPC problem (2) can be reformulated as:

$$\begin{aligned}
\underset{\mathbf{z},\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{z}^\top \mathbf{P}\mathbf{z} + \mathbf{q}^\top \mathbf{z}, && \text{(5a)}\\
\text{subject to} \quad & \mathbf{U}\mathbf{y} + \mathbf{s} = \mathbf{b}, && \text{(5b)}\\
& \mathbf{y} = \mathbf{V}\mathbf{z}, && \text{(5c)}\\
& \mathbf{z} = \mathbf{w}, && \text{(5d)}\\
& \mathbf{s} \geq 0, && \text{(5e)}
\end{aligned}$$

where $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^n$ are auxiliary variables in(5c) and (5d), respectively.

In order to solve (5), we apply Augmented Lagrangian Method and the corresponding Lagrangian for (5) is given as:

$$\begin{aligned}
\mathcal{L}(\mathbf{s},\mathbf{w},\mathbf{y},\mathbf{z}) = \frac{1}{2}\mathbf{z}^\top \mathbf{P}\mathbf{z} + \mathbf{q}^\top \mathbf{z} + {\lambda_\mathbf{1}}^\top(\mathbf{U}\mathbf{y} + \mathbf{s} - \mathbf{b})\\
+{\lambda_\mathbf{2}}^\top(\mathbf{y} - \mathbf{V}\mathbf{z}) + {\lambda_\mathbf{3}}^\top(\mathbf{z} - \mathbf{w})\\
+\frac{\mu}{2}\left(\|\mathbf{U}\mathbf{y} + \mathbf{s} - \mathbf{b}\|_2^2 + \|\mathbf{y} - \mathbf{V}\mathbf{z}\|_2^2 + \|\mathbf{z} - \mathbf{w}\|_2^2\right),
\end{aligned} \tag{6}$$

where $\lambda_\mathbf{1} \in \mathbb{R}^m$, $\lambda_\mathbf{2} \in \mathbb{R}^n$ and $\lambda_\mathbf{3} \in \mathbb{R}^n$ are the corresponding Lagrange multipliers associated with (5b), (5c) and (5d), respectively, and μ is a constant penalty weight. The closed form solution of iterative steps of the algorithm can be computed by minimizing the Lagrangian function with respect to the corresponding variable and freezing the other variables at their previous values. The iterative steps of the algorithm can be written as:

$$\begin{aligned}
\mathbf{z}^{k+1} &= \underset{\mathbf{z}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{w}^k, \mathbf{s}^k, \mathbf{y}^k, \mathbf{z}, {\lambda_\mathbf{1}}^k, {\lambda_\mathbf{2}}^k, {\lambda_\mathbf{3}}^k),\\
\mathbf{y}^{k+1} &= \underset{\mathbf{y}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{w}^k, \mathbf{s}^k, \mathbf{y}, \mathbf{z}^{k+1}, {\lambda_\mathbf{1}}^k, {\lambda_\mathbf{2}}^k, {\lambda_\mathbf{3}}^k),\\
\mathbf{w}^{k+1} &= \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{w}, \mathbf{s}^k, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}, {\lambda_\mathbf{1}}^k, {\lambda_\mathbf{2}}^k, {\lambda_\mathbf{3}}^k),\\
\mathbf{s}^{k+1} &= \underset{\mathbf{s}\geq 0}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{w}^{k+1}, \mathbf{s}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}).
\end{aligned}$$

The update for each of the variables can be computed by finding the following closed-form solution:

$$\begin{aligned}
\mathbf{z}^{k+1} &= (\mathbf{P} + \mu(\mathbf{V}^\top\mathbf{V} + \mathbf{I}))^{-1}\\
&\quad \left[-\mathbf{q} + \mathbf{V}^\top(\lambda_\mathbf{2} + \mu\mathbf{y}) - \lambda_\mathbf{3} + \mu\mathbf{w}\right], && \text{(7)}\\
\mathbf{y}^{k+1} &= (\mathbf{I} + \mathbf{U}^\top\mathbf{U})^{-1}\\
&\quad \left[\mathbf{U}^\top\left(-\mu^{-1}\lambda_\mathbf{1} - \mathbf{s} + \mathbf{b}\right) - \mu^{-1}\lambda_\mathbf{2} + \mathbf{V}\mathbf{z}\right], && \text{(8)}\\
\mathbf{w}^{k+1} &= \mathbf{z} - \mu^{-1}\lambda_\mathbf{3}, && \text{(9)}\\
\mathbf{s}^{k+1} &= \max(0, -\mu^{-1}\lambda_\mathbf{1} + \mathbf{b} - \mathbf{U}\mathbf{y}). && \text{(10)}
\end{aligned}$$

The update step for Lagrangian multipliers can also be computed similarly.

B. Solving Linear System

The variable updates in (7) and (8) involve solving the linear system of equations in each iteration. However, this system remains constant throughout the iterative steps, hence it is computationally beneficial to find the factors in the first iteration and then reuse them in subsequent iterative steps. Since usually we are dealing with a large-scale and highly sparse system, the traditional factorization techniques become computationally prohibitive. The inherent structure of the MPC problem and reformulation introduced in (5) provides a computationally efficient approach to deal with this problem. Based on the $\mathbf{UV}$ decomposition, the matrix $(\mathbf{P} + \mu\mathbf{V}^\top\mathbf{V} + \mu\mathbf{I})$ in equation (7) becomes diagonal and allows for the easy computation of the matrix inverse. Notice that the matrix $(\mathbf{I} + \mathbf{U}^\top\mathbf{U})$ in (8) is not diagonal, but we will use matrix inversion lemma to efficiently compute its inverse. According to the matrix inversion lemma, we can write $(\mathbf{I} + \mathbf{U}^\top\mathbf{U})^{-1}$ as follows:

$$(\mathbf{I} + \mathbf{U}^\top\mathbf{U})^{-1} = \mathbf{I} - \mathbf{U}^\top(\mathbf{I} + \mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U}. \tag{11}$$

Notice that the matrix $(\mathbf{I} + \mathbf{U}\mathbf{U}^\top)$ is diagonal (based on the $\mathbf{UV}$ decomposition) and thus easy to compute.

C. Inverse-free algorithm for MPC

In the previous subsections, we provide the closed form solution of primal variables and Lagrange multipliers updates. We now summarize a brief description of each step of the proposed algorithm 1 for dealing with large-scale real-time MPC problems. The algorithm takes $\mathbf{P}, \mathbf{q}, \mathbf{H}$, and $\mathbf{b}$ as input and returns the optimal values of variables as output when satisfactory termination criteria are met. In this algorithm, we use the following stopping criteria:

$$\|\mathbf{H}\mathbf{x} + \mathbf{s} - \mathbf{b}\|_\infty < \varepsilon_{\text{abs}} + \varepsilon_{\text{rel}} \max\{\|\mathbf{H}\mathbf{x}\|_\infty, \|\mathbf{b}\|_\infty\} \tag{12a}$$

$$\begin{aligned}
\|\mathbf{P}\mathbf{x} + \mathbf{H}^\top\boldsymbol{\lambda} + \mathbf{q}\|_\infty < \varepsilon_{\text{abs}}\\
+ \varepsilon_{\text{rel}} \max\{\|\mathbf{P}\mathbf{x}\|_\infty, \|\mathbf{H}^\top\boldsymbol{\lambda}\|_\infty, \|\mathbf{q}\|_\infty\}
\end{aligned} \tag{12b}$$

where $\varepsilon_{\text{abs}} = 10^{-4}$ and $\varepsilon_{\text{rel}} = 10^{-3}$ are default absolute and relative tolerance, respectively.

The proposed algorithm is designed to be efficient, requiring fewer computational resources and lower memory requirements than other methods. It is particularly suitable for applications in model predictive control, which often

Algorithm 1 Inverse free algorithm for solving large-scale MPC problems

Input: $(\mathbf{P}, \mathbf{q}, \mathbf{H}, \mathbf{b})$, fixed $\mu > 0$, and initial points $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$ $\mathbf{s} \in \mathbb{R}^m$

Construct a $\mathbf{UV}$ decomposition based on non-zero elements of $\mathbf{H}$
$\mathbf{U_i} := \mathbf{I} - \mathbf{U}^\top(\mathbf{I} + \mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U}$
$\mathbf{V_i} := (\mathbf{P} + \mu\mathbf{V}^\top\mathbf{V} + \mu\mathbf{I})^{-1}$
repeat
  $\mathbf{z} \leftarrow \mathbf{V_i}\left[(-\mathbf{q} + \mathbf{V}^\top(\lambda_2 + \mu\mathbf{y}) - \lambda_3 + \mu\mathbf{w}\right]$
  $\mathbf{y} \leftarrow \mathbf{U_i}\left[\mathbf{U}^\top\left(\mathbf{b} - \mu^{-1}\lambda_1 - \mathbf{s}\right) - \mu^{-1}\lambda_2 + \mathbf{V}\mathbf{z}\right]$
  $\mathbf{w} \leftarrow \mathbf{z} - \mu^{-1}\lambda_3$
  $\mathbf{s} \leftarrow \max(0, \mathbf{b} - \mu^{-1}\lambda_1 - \mathbf{U}\mathbf{y})$
  $\lambda_1 \leftarrow \lambda_1 + \mu(\mathbf{U}\mathbf{y} + \mathbf{s} - \mathbf{b})$
  $\lambda_2 \leftarrow \lambda_2 + \mu(\mathbf{y} - \mathbf{V}\mathbf{z})$
  $\lambda_3 \leftarrow \lambda_3 + \mu(\mathbf{z} - \mathbf{x})$
until stopping criteria is met.

Output: $\mathbf{z}^* \leftarrow \mathbf{z}$, $\mathbf{s}^* \leftarrow \mathbf{s}$, $\mathbf{y}^* \leftarrow \mathbf{y}$

involve solving large-scale optimization problems over multiple time steps. The algorithm achieves its efficiency by taking advantage of the sparsity structure of the problem. It only stores and performs arithmetic operations on the non-zero elements, reducing the memory requirements and computational cost. By minimizing the number of non-zero elements, the algorithm avoids unnecessary computations and allows for faster execution. Another key feature of the proposed algorithm is its warm-start capability. In model predictive control problems, the optimization factors only need to be computed at the first time step and can be reused in subsequent iterations for the entire time horizon. This means that the algorithm can be initialized with the factors from the previous iteration, reducing the computational time and improving the overall performance. The warm-start feature is particularly useful when solving large-scale optimization problems over multiple time steps, as it allows the algorithm to reuse previously computed factors, which saves time and computational resources.

Overall, the proposed algorithm is well-suited for model predictive control applications due to its computational efficiency, lower memory requirements, and warm-start capability. These features enable it to solve large-scale optimization problems over multiple time steps, making it a valuable tool in the field of control engineering.

IV. Experimental Results

To demonstrate the computational performance of our algorithm, we tested it on the benchmark problems developed by ABB Research Corporation [21]. For comparative analysis, we chose the popular first-order method, the Alternating Direction Method of Multipliers (ADMM), and a second-order solver called "qpOASES". "qpOASES" is an off-the-shelf open-source solver for solving convex quadratic programming problems and is based on active set methods [22], [23]. qpOASES uses a warm-start that relies on data from previous QP solutions. With a desirable warm-start, the computational performance of qpOASES is likely to increase dramatically, but this comes at the cost of application-specific tuning. ADMM is a popular approach for MPC applications. This method is based on a first-order operator splitting technique and is suitable for real-time applications in embedded systems. In contrast to ADMM, our first-order method solves the KKT matrix using $\mathbf{UV}$ factorization in one iteration and reuses these factors throughout the rest of the iterations. This alone makes the performance an order of magnitude faster than competing interior-point methods. With further memory caching or a warm start, the computational gains could be even more.

In the following section, we will conduct a comprehensive computational comparison of the three algorithms across all benchmark case studies. Additionally, we will select one linear time-invariant case study to conduct a more in-depth analysis of the MPC solution. All experiments for each algorithm will be carried out on an Intel Core i7 processor running MATLAB 2021b, which has eight 2.4 GHz cores and 16GB of RAM.

A. Benchmark MPC problems

The benchmark problems outlined in [21] span a broad spectrum of application areas, encompassing academic case studies as well as industrial applications. In this study, we conducted a performance evaluation of our algorithm 1 against the state-of-the-art first and second-order methods mentioned earlier. The results of the evaluation are summarized in Table I, where the numbers indicate the superiority of algorithm 1 over both first and second-order methods. To demonstrate the scalability of our approach, we compared the three algorithms across a time horizon ranging from 10-40 time steps as well as a longer horizon of 100 time steps. For each algorithm, we report the computation time in seconds and show that our proposed algorithm outperforms existing methods.

The comparison between the algorithms is based on various performance metrics, such as the system, the number of states and inputs, the horizon length, and the computation time in seconds. The first column of Table I presents the name of each system, followed by the number of states and inputs in the second and third columns, respectively. The fourth column denotes the horizon length for each problem. The subsequent three columns indicate the computation time in seconds for qpOASES, ADMM, and Algorithm 1, respectively, for the corresponding horizon length of each problem. The last two columns display the computation time in seconds for ADMM and Algorithm 1, respectively, for a fixed horizon length of 100.

The results in the table demonstrate that Algorithm 1 performs better than both qpOASES and ADMM in terms of computation time for most benchmark problems, especially for longer horizon lengths. For example, for the Ball On Plate system with a horizon length of 25, Algorithm 1 takes

TABLE I: Comparison of numerical methods for solving MPC benchmark problem.

System	States	Inputs	Horizon	qpOASES	ADMM	Algorithm 1	Horizon	ADMM	Algorithm 1
Ball On Plate	2	1	25	1.011	1.43	0.43	100	32.66	1.34
Quadcopter	12	4	20	0.11	0.95	0.06	100	93.30	0.62
Spring Mass	6	2	20	3.07	4.96	1.21	100	526.38	5.59
Double Inverted Pendulum	4	2	10	2.32	32.49	6.14	100	3177.57	32.19
Spacecraft	7	4	10	2.28	44.55	3.71	100	5047.48	37.77
BDL	11	3	15	0.14	0.67	0.09	100	0.63	0.07
Fiordos	2	1	20	0.065	0.02	0.044	100	0.32	0.02
Forces	2	1	20	0.095	0.17	0.043	100	1.46	0.06
Helicopter	6	2	10	0.53	3.67	0.49	100	335.11	2.47
Polytopic Terminal	2	1	10	0.093	0.086	0.038	100	1.48	0.06
Shell	9	3	40	0.19	1.37	0.12	100	48.85	0.25

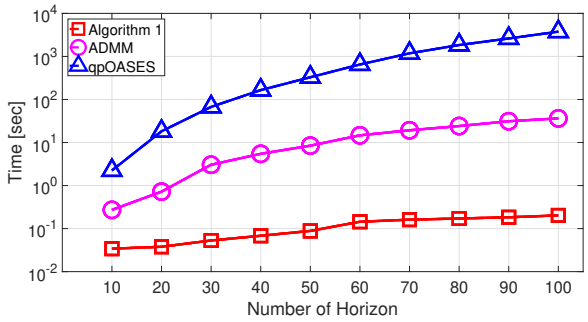

Fig. 1: Computational time comparison for increasing horizon length of Quadcopter problem.

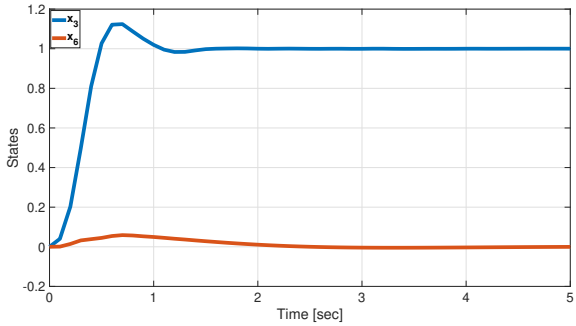

Fig. 2: Linear position and velocity in z direction of Quadcopter.

only 0.43 seconds compared to 1.011 seconds for qpOASES and 1.43 seconds for ADMM. Similarly, for the Quadcopter system with a horizon length of 100, Algorithm 1 takes only 0.62 seconds compared to 93.30 seconds for ADMM.

Our proposed technique's efficiency is particularly evident when solving larger MPC problems, as shown in Figure 1. The figure displays a log-scale comparison of computation times for a Quadcopter system using our algorithm, qpOASES, and ADMM, for various horizon lengths. The x-axis shows the horizon length, increased by 10 in each instance, while the y-axis displays the computation time in seconds. As the figure shows, our algorithm outperforms ADMM by a factor of 10 and qpOASES by a factor of 100, with this performance gain becoming more significant as the problem size increases.

B. Linear Model Predictive Control

The quadcopter has 12 states $[\mathbf{x}, \mathbf{y}, \mathbf{z}, \phi, \theta, \psi, \dot{\mathbf{x}}, \dot{\mathbf{y}}, \dot{\mathbf{z}}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$ where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ denotes the position in inertial reference and ϕ, θ, ψ, represent the roll, pitch and yaw angles, respectively. Additionally, the remaining states are velocities of linear and angular positions. The control inputs $[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4]$ are the squared angular velocities of the four rotors of the system. The system is discretized with a sampling time of $t = 0.5sec$ with input and state constraints defined as $\|\mathbf{u}_k\|_\infty \leq 2.0$ and $\|\mathbf{x}_k\|_\infty \leq 3.5$, respectively.

The goal is to design a predictive control to bring the system to a reference state of $\mathbf{x}_r = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ with a prediction horizon of $N = 10$. As shown in Figure 2,

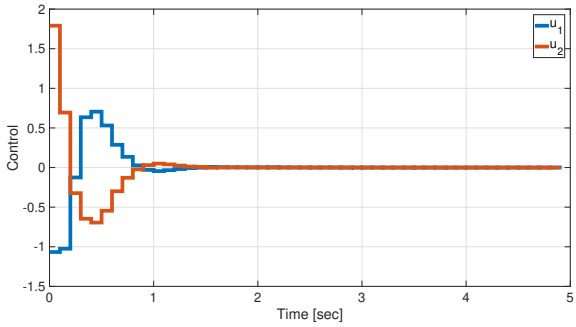

Fig. 3: Control actions to control Quadcopter motion.

which displays the states of the Quadcopter system over time, the controller quickly stabilizes the system using the corresponding control signal in Figure 3, which shows the control inputs applied to the system.

V. Conclusion

This paper introduces a new inverse-free algorithm for real-time model predictive control of large-scale systems. The algorithm combines the $\mathbf{UV}$ factorization method with a first-order approach to solve the quadratic programming problem for a linear time-varying system. The proposed factorization method eliminates the need for large matrix inversions in each optimization iteration, providing a computational advantage over both first-order (ADMM) and second-order (qpOASES) methods.

In addition to demonstrating the benefits of this approach

on standard linear benchmark problems, we also aim to apply it to a nonlinear MPC example using successive linearization. Our future work will focus on extending this algorithm to embedded real-world applications for solving large-scale nonlinear MPC problems, utilizing differentiable models and gradient information obtained through automatic differentiation. We will also focus on data-driven applications with unknown dynamics, with a specific focus on the HVAC control problem for commercial buildings where modeling the exact heat transfer dynamics is challenging.

REFERENCES

[1] A. Zanarini, M. Jafargholi, and H. Peyrl, "Exploiting parallelization in explicit model predictive control," in *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, 2013, pp. 1–7.

[2] J. Buijs, J. Ludlage, W. V. Brempt, and B. D. Moor, "Quadratic programming in model predictive control for large scale systems," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 301–306, 2002, 15th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667015387218

[3] P. Otta, O. Šantin, and V. Havlena, "On the quadratic programming solution for model predictive control with move blocking," in *2021 23rd International Conference on Process Control (PC)*, 2021, pp. 49–54.

[4] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A condensed and sparse qp formulation for predictive control," in *50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 5217–5222.

[5] D. Arnström, A. Bemporad, and D. Axehill, "A dual active-set solver for embedded quadratic programming using recursive LDL^T updates," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4362–4369, 2022.

[6] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[7] F. Rey, D. Frick, A. Domahidi, J. Jerez, M. Morari, and J. Lygeros, "Admm prescaling for model predictive control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3662–3667.

[8] L. Lu, "Separable model predictive control via alternating direction method of multipliers for large-scale systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10 499–10 504, 2014, 19th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016432817

[9] R. Rostami, G. Costantini, and D. Görges, "ADMM-based distributed model predictive control: Primal and dual approaches," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 6598–6603.

[10] S. P. Ahmadi and A. Hansson, "A distributed second-order augmented lagrangian method for distributed model predictive control," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 192–199, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896321013197

[11] M. Adil, S. Tavakkol, and R. Madani, "Rapid convergence of first-order numerical algorithms via adaptive conditioning," 2021. [Online]. Available: https://arxiv.org/abs/2103.00736

[12] S. Lee, N. Chatzipanagiotis, and M. M. Zavlanos, "A distributed augmented lagrangian method for model predictive control," in *56th IEEE Annual Conference on Decision and Control (CDC)*, 2017, pp. 2888–2893.

[13] C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann, and T. Bertram, "Exploiting sparse structures in nonlinear model predictive control with hypergraphs," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018, pp. 1332–1337.

[14] I. Nielsen, "Structure-exploiting numerical algorithms for optimal control," 2017.

[15] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 6974–6979, 2008, 17th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016400662

[16] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 668–674.

[17] G. Frison, D. K. M. Kufoalor, L. Imsland, and J. B. Jørgensen, "Efficient implementation of solvers for linear model predictive control on embedded devices," in *2014 IEEE Conference on Control Applications (CCA)*, 2014, pp. 1954–1959.

[18] I. Nielsen and D. Axehill, "A parallel structure exploiting factorization algorithm with applications to model predictive control," *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 3932–3938, 2015.

[19] M. Adil, R. Madani, S. Tavakkol, and A. Davoudi, "A first-order numerical algorithm without matrix operations," 2022. [Online]. Available: https://arxiv.org/abs/2203.05027

[20] M. Adil, "Fast and parallelizable numerical algorithms for large scale conic optimization problems," Ph.D. dissertation, The University of Texas at Arlington, 12 2021. [Online]. Available: http://hdl.handle.net/10106/30218

[21] D. Kouzoupis, A. Zanelli, H. Peyrl, and H. J. Ferreau, "Towards proper assessment of qp algorithms for embedded model predictive control," in *2015 European Control Conference (ECC)*, 2015, pp. 2609–2616.

[22] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[23] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

APPENDIX

We define the variable $\mathbf{z}$ as:

$$\mathbf{z} := \begin{bmatrix}\mathbf{x}_0, \cdots, \mathbf{x}_N, \mathbf{u}_0, \cdots, \mathbf{u}_{N-1}\end{bmatrix}^\top,$$

and the quadratic objective function of problem (2) can be defined by $\mathbf{P}$ and $\mathbf{q}$, where

$$\mathbf{P} := \operatorname{diag}\{\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_\mathrm{N}, \mathbf{R}_1, \cdots, \mathbf{R}_\mathrm{N}\},$$

and

$$\mathbf{q} := \begin{bmatrix}\mathbf{Q}_1\mathbf{x}_r & \mathbf{Q}_2\mathbf{x}_r & \cdots & \mathbf{Q}_N\mathbf{x}_r & \mathbf{0}^{N\times n_u}\end{bmatrix}^\top.$$

The equality constraint associated with (1b) in MPC formulation can be casted as:

$$\mathbf{A}_{\mathrm{eq}} := \left[\begin{array}{ccccc|cccc} -\mathbf{I} & \mathbf{0} & & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}_1 & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & -\mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{B}_2 & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{A}_N & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B}_N \end{array}\right],$$

whereas the inequality constraint (1c) and (1d) is

$$\mathbf{A}_{\mathrm{ineq}} := \mathbf{I}^m.$$

The new equality constraint of reformulated quadratic program is defined as:

$$\mathbf{H} := \begin{bmatrix}\mathbf{A}_{\mathrm{eq}} & -\mathbf{A}_{\mathrm{ineq}} & \mathbf{A}_{\mathrm{ineq}}\end{bmatrix}^\top,$$

and transformed right hand side is given as:

$$\mathbf{b} := \begin{bmatrix} -\mathbf{x}(0) \\ \mathbf{0}^{N\times n_x} \\ -\mathbf{x}_{\min} \times \mathbf{1}^{N+1} \\ -\mathbf{u}_{\min} \times \mathbf{1}^{N} \\ \mathbf{x}_{\max} \times \mathbf{1}^{N+1} \\ \mathbf{u}_{\max} \times \mathbf{1}^{N} \end{bmatrix}.$$