Cloud-based computational model predictive control using a parallel multi-block ADMM approach

Yaling Ma, Runze Gao, Li Dai*, Jinxian Wu and Yuanqing Xia Senior Member, IEEE

Abstract

Heavy computational load for solving nonconvex problems for large-scale systems or systems with real-time demands at each sample step has been recognized as one of the reasons for preventing a wider application of nonlinear model predictive control (NMPC). To improve the real-time feasibility of NMPC with input nonlinearity, we devise an innovative scheme called cloud-based computational model predictive control (MPC) by using an elaborately designed parallel multi-block alternating direction method of multipliers (ADMM) algorithm. This novel parallel multi-block ADMM algorithm is tailored to tackle the computational issue of solving a nonconvex problem with nonlinear constraints. It is ensured that the designed algorithm converges to a locally optimal solution of the optimization problem under reasonable assumptions by using the Kurdyka–Łojasiewicz property. With the help of this distributed optimization algorithm, a computational MPC scheme is developed, which can transform the NMPC optimization problem into a set of subproblems only associated with the decision variables at one prediction step. Through the parallel computing algorithm, the computational MPC can deal with large computational loads caused by high dimensional optimization problems, and improve computational efficiency. Furthermore, to allow for a more efficient implementation of the developed computational MPC and alleviate local calculation loads, a cloud-based computational MPC architecture is devised, which makes significantly better use of computational resources provided by a cloud server. An important advantage of this architecture with Docker container to implement parallelization is that it does not lead to large increases in the solution time regardless of how long the prediction horizon is set. Finally, the developed cloud-based computational MPC architecture is trialed on a group of plug-in hybrid electric vehicles (PHEVs).

Index Terms

Computational model predictive control, cloud control system, parallel multi-block ADMM, parallel computing, nonconvex problems, nonlinear system, Docker container.

I. INTRODUCTION

Nonlinear model predictive control (NMPC) is an advanced control strategy that uses nonlinear models or takes into account general nonlinear boundary conditions as well as nonsquare cost-functional conditions [1]. NMPC has become popular in chemical and process applications that typically have slow dynamics due to its ability to systematically handle diverse constraints and nonlinear dynamics [2]. Whereas the computational need for solving finite-horizon optimization problems with nonlinear constraints, especially nonconvex optimization problems, in real-time at each sample step prevents its wider applications [3]. To satisfy the computing requirements and improve the real-time feasibility of NMPC, interest in using the characteristics of optimization problems to improve the computational efficiency has been concerned with first-order approaches to solve optimization problems, such as fast gradient methods [4] or alternating direction method of multipliers (ADMM) [5].

ADMM is a process of decomposition and coordination that can decompose large global problems into smaller local problems and coordinate their solutions to find a solution for the original problem [6]. One strength of ADMM is that each subproblem allows predicting its local future depending only on the model and objective function of each part. In principle, ADMM is better suited to large-scale distributed optimization problems because it relieves computational and communication loads [7]. The standard ADMM is a general framework to solve MPC optimization problems with linear constraints due to their favorable decomposability over nonlinear constraints [8]. The ADMM approach for solving an NMPC optimization problem, especially when there are a nonconvex objective...
function and nonlinear constraints in the problem, up to now has only been considered sporadically. One of the reasons is that convergence has only been guaranteed under certain assumptions when ADMM is directly applied to nonconvex problems [9].

The work in [10] designs a new generic two-block ADMM framework called neADMM for handling nonlinear constraints, which can converge to a global optimal solution of the problem and yield a sublinear convergence rate $o(1/k)$, where $k$ is the number of iterations. Whereas, the practical utility of two-block ADMM for nonconvex optimization problems is limited by the excessive computational effort due to its slow (sublinear) convergence rate and non-parallelizability [7], which usually results in real-time control being impossible. Multi-block ADMM, which involves at least three variables, was first discussed by Chen et al [11]. But it could be divergent to extend the ADMM directly to the case of a multi-block separable optimization problem [12]. The authors in [13] design a general framework for proving convergence of multi-block ADMM and show that its global subsequential convergence rate is $o(1/\sqrt{k})$. To further enhance its computational performance, the authors in [7] propose an accelerated parallel ADMM, which integrates Nesterov acceleration technique with multi-block parallelization into an ADMM scheme. Moreover, to accelerate the solution process of finding the optimal solutions, based on the separable objective functions and an augmented Lagrangian function, the update of the optimization variable consisting of $N$ subvectors can be performed simultaneously, with each subvector updated by a separate minimization [6]. However, these algorithms may still suffer from heavy computation as the empirical environment has intrinsically high overhead and only a finite number of cores. It can be expected that the implementation of parallel multi-block ADMM algorithms would become more efficient when computing is outsourced to other platforms, such as those with many-core processors [3] or powerful remote servers, generically called clouds [14].

As an implementation of utility computing, cloud computing can provide more flexible allocation of scalable computing resources, such as CPUs, memories, and disks, in accordance with application requirements for varying computing scales [15]. Cloud computing applications are growing rapidly as a result of their ease of deployment, scalability, and cost reduction. They provide a perfect engine for storing and processing large amounts of data, controlling systems, and optimizing performance [16]. In several industrial control contexts, cloud-based control is a powerful solution when situations arise in which clients do not have enough computing resources to carry out control tasks or in which the control performance needs to be improved by sharing operational data with other clients [17]. An illustrative example of the benefits of outsourcing computing can be observed in the field of MPC [18]. A model predictive controller containing optimization problems may need the cloud with “infinite” computing capacity to operate when sufficient computational resources on the device are not locally available [19].

The body of existing work on cloud-based MPC can be roughly categorized into three main areas.

(i) Designing different cloud-based MPC architectures suited to a variety of applications.

(ii) Studying privacy concerns in cloud-based control.

(iii) Focusing on parallel computing implementation solution using the cloud for MPC.

Category (i) contains various practical applications, including vehicle control [20], building thermostatic controls [21], industrial robot control [22] and so on. For example, [22] shows the feasibility of cloud-based robot control while outsourcing even real-time critical low-level functions. In category (ii), to maintain the privacy of cloud outsourced MPC, encrypted methods based on homomorphic encryption [23], differential privacy [24], and algebraic transformations [25] are taken into account in the context of MPC. An example of cloud-based MPC in category (iii) is Spark cloud-based parallel computing for traffic network flow predictive control in [15]. By designing a two-level hierarchical parallel genetic algorithm, the computationally intensive MPC optimization problem is decomposed into multiple parallel subproblems, and a parallel computing architecture for the MPC on the cloud is presented to improve the computational efficiency. Although widespread research has been conducted on cloud-based MPC, there are few studies on how to best utilize the possibilities that the cloud provides, such as its large and flexible compute capacity. Benefitting from computation resources on the cloud, we devise a novel cloud-based computational MPC architecture in this paper.

To computationally efficiently tackle the optimization problem involved in NMPC with input nonlinearity, especially nonconvex optimization problems with heavy calculation loads, the ADMM algorithm, well suited to distributed optimization, provides the possibility of parallel computing. In this paper, to accelerate the solution process and improve the real-time feasibility of the NMPC, we design a parallel multi-block ADMM algorithm. Moreover, inspired by the idea in [26], the convergence of the designed optimization algorithm for NMPC optimization problem is established by using Kurdyka–Lojasiewicz property. Different from [26], which establishes the
convergence of multi-block ADMM under nonconvex frameworks with a linear equality constraint, we expand the proof of the convergence to parallel multi-block ADMM under nonconvex frameworks with nonlinear constraints. By utilizing this distributed optimization algorithm, we develop a computational MPC scheme that can decompose the NMPC optimization problem into multiple individual subproblems only associated with the decision variables at one prediction step. By making use of the parallel multi-block ADMM algorithm, the computational MPC can handle large computational loads caused by high dimensional optimization problems, and therefore improve the computational efficiency.

The practical implementation of computational MPC may be limited by the local computing resources available. When clients lack the computational resources to handle optimization problems locally, outsourcing computations to the cloud appears to be an attractive and cost-effective solution, since developed extensively cloud computing services can provide abundant computational resources. Most existing works in cloud-based MPC are performed in a single-node computing environments at a cloud server, which is non-ideal since it does not fully exploit the potential of cloud computing. The recently popular cloud computing technology by Docker containers, which can support multi-node computing environment by containers, is suitable for solving complex large-scale data iteration optimization problems. Accordingly, we design a cloud-based computational architecture based on Docker cloud computing to efficiently implement the developed computational MPC scheme and to reduce the time required for the solution process. This architecture, as depicted in Fig. 1, is composed of local controlled plants and a cloud-based computational model predictive controller at the cloud server. The model predictive controller, Docker containers, and remote dictionary server (Redis\(^2\)) modules make up the cloud-based controller. The innovative architecture integrates parallel computing with container cloud technology, which allows better utilization of the computational resources offered by mutually isolated containers. A parallel calculation is performed in corresponding containers for the decomposed multiple subproblems, which is more advanced than the prior single-node computing architecture. In addition, all intermediate results are cached in Redis for the cloud to further speed up the data acquisition at the following iteration. The devised cloud-based computational MPC architecture is verified by a group of plug-in hybrid electric vehicles (PHEVs), and experimental simulation results demonstrate that distributing a large optimization problem to multiple containers at the cloud server decreases the execution time by parallelizing smaller subproblems. It is worth mentioning that, with the increase of prediction horizon, the proposed cloud-based computational MPC architecture that uses Docker containers to achieve parallelization does not result in significant increases in solution time.

The contributions of this paper are three folds.

1. A computationally efficient parallel multi-block ADMM algorithm is designed to solve nonconvex optimization problems with nonlinear constraints. Based on this parallel computing algorithm, we develop a computational MPC scheme, which can transform the NMPC optimization problem into a sequence of subproblems only associated with the decision variables at one prediction step.

2. Inspired by [26] and using the Kurdyka–Łojasiewicz property, the convergence of the parallel multi-block ADMM algorithm for nonconvex optimization problems involved in NMPC with input nonlinearity is established under suitable assumptions.

3. Based on Docker cloud computing technique, a cloud-based computational MPC architecture is devised for a paralleled implementation of the developed computational MPC, which does not lead to large increases in the solution time regardless of how long the prediction horizon is chosen.

The rest of the paper is organized as follows. In Section II problem formulation and the preliminaries are presented. In Section III computational MPC using a parallel multi-block ADMM approach is proposed and convergence analysis is established. Section IV describes the framework of the cloud-based computational model predictive controller. In Section V, the plug-in hybrid electric vehicle as a controlled plant demonstrates the efficacy of the cloud-based computational MPC architecture. Conclusions and future work are outlined in Section VI.

Notations: Throughout this paper, \(\mathbb{R}\) denotes the set of real numbers, \(\mathbb{R}_{>0}\) and \(\mathbb{R}_{\geq0}\) denote the sets \(\{r \in \mathbb{R} : r > 0\}\) and \(\{r \in \mathbb{R} : r \geq 0\}\), and \(\mathbb{R}^n\) denotes the n-dimensional Euclidean space. Let \(I\) be the identity matrix and \(0\) be a vector or a matrix with zero entries. \(P^\top\) is the transpose of matrix \(P\), and \(P = \text{diag}(P_1, \ldots, P_N)\) is a diagonal matrix with diagonal blocks \(P_i, i = 1, 2, \ldots, N\). For any two vectors \(x, y \in \mathbb{R}^n\), \(\langle x, y \rangle = x^\top y\).

\(^2\)https://redis.io/
II. PROBLEM FORMULATION AND PRELIMINARIES

A. Problem formulation

Consider the following discrete-time nonlinear systems

\[ x_{k+1} = A_k x_k + b(u_k), \]

where \( x_k \in \mathbb{R}^{n_x} \) and \( u_k \in \mathbb{R}^{n_u} \) represent the state and the control input at the current sample step \( k \), respectively, \( A_k \in \mathbb{R}^{n_x \times n_x} \) is a system state matrix, which could be time varying, and \( b(u_k) : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x} \) is a nonlinear continuous function, which is allowed to be nonconvex. The state and the control input are assumed to be subject to mixed constraints

\[ (x_k, u_k) \in \mathcal{X} \times \mathcal{U}, \]

where \( \mathcal{X} \subseteq \mathbb{R}^{n_x} \) and \( \mathcal{U} \subseteq \mathbb{R}^{n_u} \) are nonempty, closed, and possible nonconvex sets. The objective of this paper is to design a computationally efficient model predictive controller for the nonlinear system (1) which can make the constraints (2) satisfied and the system state stabilized to the desired equilibrium point,

\[ \lim_{k \to \infty} x_k = x_{\text{exp}}. \]

We simulate forward in time over a prediction horizon of \( N \) steps using the current state \( x_k \), so as to obtain the predicted state sequence \( x_k = [x_{1|k}, \ldots, x_{i+1|k}, \ldots, x_{N|k}]^T \) and the predicted control sequence \( u_k = [u_{0|k}, \ldots, u_i|k, \ldots, u_{N-1|k}]^T \), where \( x_{i|k} \) and \( u_{i|k} \) represent the \( i \)-step-ahead predicted states and predicted inputs at sample step \( k \), respectively. The optimal control sequence \( u^*_k \) can be found by solving an MPC optimization problem (4), and the first element of \( u^*_k \), i.e., \( u^*_{0|k} \), is used as the actual plant input \( u_k = u^*_{0|k} \) at time \( k \).

\[ \begin{align*}
\text{minimize} & \quad f(x_k) + g(u_k) \\
\text{s.t.} & \quad x_{0|k} = x_k
\end{align*} \]
\[ x_{i+1|k} = A_{i|k}x_{i|k} + b(u_{i|k}), \quad i = 0, 1, \ldots, N - 1 \]  
(4c)

\[(x_{i+1|k}, u_{i|k}) \in \mathcal{X} \times \mathcal{U}, \quad i = 0, 1, \ldots, N - 1 \]  
(4d)

For the cost function (4a), the cost function in terms of states \( f(x_k) : \mathbb{R}^{Nn_x} \rightarrow \mathbb{R}_{\geq 0} \) is a continuously differentiable function, and the control cost function \( g(u_k) : \mathbb{R}^{Nn_u} \rightarrow \mathbb{R}_{\geq 0} \) is a proper lower semicontinuous function. Both of them are separable, that is,

\[
f(x_k) = \sum_{i=1}^{N} f_i(x_{i|k})
\]
(5)

\[
g(u_k) = \sum_{i=0}^{N-1} g_i(u_{i|k})
\]
(6)

where \( f_i(x_{i|k}) : \mathbb{R}^{n_x} \rightarrow \mathbb{R}_{\geq 0} \) and \( g_i(x_{i|k}) : \mathbb{R}^{n_u} \rightarrow \mathbb{R}_{\geq 0} \) are the stage state cost function and the stage control cost function, respectively. For (4c), letting \( b(u_{i|k}) = [b^T(u_{0|k}), \ldots, b^T(u_{N-1|k})]^T \in \mathbb{R}^{Nn_x} \) and considering the initial condition \( x_{0|k} = x_k \) in (4b), the prediction model can be rewritten as

\[
x_k = \Phi_k x_0 + \Psi_k b(u_k),
\]
(7)

or,

\[
x_{i|k} = \Phi_k x_{i-1} + \Psi_k b(u_k), \quad i = 1, \ldots, N
\]
(8)

where \( \Phi_{k,i} \) and \( \Psi_{k,i} \) are the \( i \)th block rows of \( \Phi_k \) and \( \Psi_k \)

\[
\Phi_k = \begin{bmatrix}
A_{0|k} \\
A_{0|k}A_{1|k} \\
\vdots \\
A_{0|k}A_{1|k}\cdots A_{N-1|k}
\end{bmatrix}, \quad \Psi_k = \begin{bmatrix}
I & 0 & \cdots & 0 \\
A_{1|k} & I & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
A_{N-1|k} \cdots A_{N-1|k}
\end{bmatrix}
\]
(9)

Here the \( p \)th block-row of \( \Phi_k \in \mathbb{R}^{Nn_x \times n_x} \) is \( \prod_{i=0}^{p-1} A_{i|k} \), and \( \Psi_k \in \mathbb{R}^{Nn_x \times Nn_u} \) is lower-triangular with the \( p \)th row and the \( q \)th column block given by \( \prod_{i=q}^{p-1} A_{i|k} \) for \( q < p \) and by the identity matrix for \( p = q \). For the handling of the constraint (4d), define the stage indicator functions \( h^x_i(x_{i|k}) \) and \( h^u_i(u_{i|k}) \) as

\[
h^x_i(x_{i|k}) = \begin{cases} 
0 & x_{i|k} \in \mathcal{X} \\
\infty & \text{otherwise}
\end{cases}
\]
(10)

\[
h^u_i(u_{i|k}) = \begin{cases} 
0 & u_{i|k} \in \mathcal{U} \\
\infty & \text{otherwise}
\end{cases}
\]

and the indicator functions \( h^x(x_k) \) and \( h^u(u_k) \) as

\[
h^x(x_k) = \sum_{i=1}^{N} h^x_i(x_{i|k})
\]

\[
h^u(u_k) = \sum_{i=0}^{N-1} h^u_i(u_{i|k})
\]

Through these definitions, we incorporate state and input constraints into the cost function (4a) and rewrite problem (4) as

\[
\begin{align*}
\text{minimize} & \quad f(x_k) + g(u_k) + h^x(x_k) + h^u(u_k) \\
\text{s.t.} & \quad \Phi_k x_0 + \Psi_k b(u_k) - x_k = 0
\end{align*}
\]
(11)

Now solving the MPC optimization problem (4) is transformed into solving the optimization problem (11).
B. Preliminaries

Throughout this subsection, we review some well-known related mathematical concepts and properties, which are helpful to prove the convergence of the designed parallel multi-block ADMM algorithm in Section III.

In mathematical analysis, semicontinuity is a property of extended real-valued functions that is weaker than continuity. An extended real-valued function $f$ is lower semicontinuous at a point $x_0$ if, roughly speaking, the function values for arguments near $x_0$ are not much lower than $f(x_0)$. The formal definition of the lower semicontinuous function is stated in Definition 1.

**Definition 1.** (Lower semicontinuous function) [26]. Let a function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$, $x \in \mathbb{R}^n$ be a limit point of $\mathbb{R}^n$, and $\text{dom} f$ represent for the domain of $f$. $f$ is proper if dom$f \neq \emptyset$ and lower semicontinuous (l.s.c.) at $x_0$ if and only if

$$\liminf_{x \to x_0} f(x) \geq f(x_0)$$

where $\liminf$ is the limit inferior of the function $f$ at point $x_0$, $f$ is called a proper lower semicontinuous function if it is proper and lower semicontinuous at each point of its domain of definition.

The concept of subdifferentiation will play an essential role in the development of the subsequent theories and algorithms.

**Definition 2.** (Subdifferential) [27]. Given a proper lower semicontinuous function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ and any $x \in \text{dom} f$, the Fréchet subdifferential and limiting subdifferential are respectively defined as:

- **Fréchet subdifferential:** a set of vectors $u \in \mathbb{R}^n$ satisfying
  $$\liminf_{y \neq x \to y} \frac{f(y) - f(x) - \langle u, y - x \rangle}{\|y - x\|} \geq 0$$

  written by $\hat{\partial} f(x)$, is the Fréchet subdifferential of $f$ at $x$.

- **Limiting subdifferential:** $\partial f(x)$, the limiting subdifferential of $f$ at $x$, is denoted by
  $$\partial f(x) = \left\{ u \in \mathbb{R}^n : \exists x^k \to x, \text{ such that } f(x^k) \to f(x), u^k \in \hat{\partial} f(x^k) \to u, \text{ as } k \to \infty \right\}$$

For a proper lower semicontinuous function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$, it have the following two properties.

(i) Fermat’s rule [28]: if $f$ attains a local minimum at $x^*$, then $x^*$ is a stationary point of $f$, that is, $0 \in \partial f(x^*)$.

(ii) If $f$ satisfies the condition $\|f(x) - f(y)\| \leq L\|x - y\|$, for any $x, y \in \text{dom} f$, where $L$ is a Lipschitz constant, then $f$ is $L$-Lipschitz continuous.

One important assumption for the study of the algorithms in this work is that functions satisfy the following Kurdyka–Łojasiewicz property (K-L property), which means that the function under consideration is sharp except for a reparameterization. K-L property has been successfully used to analyze various types of asymptotic behavior, such as gradient methods [29], proximal methods [30], projection methods or alternating methods [31].

**Definition 3.** (Kurdyka–Łojasiewicz property) [32]. A function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is said to have the Kurdyka-Łojasiewicz property at $\tilde{x} \in \text{dom} \partial f$, if there exists $\eta > 0$, $\delta > 0$, and a continuous concave function $\varphi(\cdot) : [0, \eta] \to \mathbb{R}_{>0}$ such that

(i) $\varphi(0) = 0$,
(ii) $\varphi'(x) > 0, \forall x \in [0, \eta]$,
(iii) $\varphi$ is continuous on $[0, \eta]$,
(iv) $\varphi$ is smooth concave on $[0, \eta]$,
(v) $\forall x \in O \cap \{x : f(\tilde{x}) < f(x) < f(\tilde{x}) + \eta\}$, the Kurdyka–Łojasiewicz inequality holds

$$\varphi'(f(x) - f(\tilde{x})) \text{ dist}(0, \partial f(x)) \geq 1,$$

where $O$ denotes the neighborhood of $\tilde{x}$ and $\text{dist}(0, \partial f(x)) = \inf\{\|\tilde{x} - y\| : y \in \partial f(x)\}$.

Bregman distance is a measure of difference between two points, defined in terms of a strictly convex function.

**Definition 4.** (Bregman distance) [33]. Let $\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}$ be a continuously-differentiable, strictly convex function defined on a closed convex set. The Bregman distance linked to $\phi(\cdot)$ for points $x, y \in \mathbb{R}^n$, which is denoted by
\( \Delta \phi(x, y) \), is the difference between the value of \( \phi(\cdot) \) at point \( x \) and the value of the first-order Taylor expansion of around point \( y \) evaluated at point \( x \):

\[
\Delta \phi(x, y) = \phi(x) - \phi(y) - \langle \nabla \phi(y), x - y \rangle,
\]

(12)

where the symbol \( \nabla \) denotes the gradients.

Moreover, if \( \phi(\cdot) \) is \( \alpha \)-strongly convex with the constant \( \alpha \), then

\[
\Delta \phi(x, y) \geq \frac{\alpha}{2} \| x - y \|^2
\]

(13)

### III. Computational model predictive control using a parallel multi-block ADMM approach

#### A. Algorithm design

The ADMM algorithm is an extensively applied distributed optimization scheme and has also proven to be very useful and rapid in solving various nonconvex optimization problems. However, through a mass of numerical experiments, it is shown that when the optimization variables \( u_k, x_k \) of problem (11) are high-dimensional, the solution time is much longer than that of low-dimensional optimization variables. Inspired by the results in [5], [26], we design a novel parallel multi-block ADMM algorithm to reduce the solution time of problem (11). By utilizing this parallel computing algorithm, a computational MPC is developed naturally to decouple the nonconvex optimization problem (11) involved in NMPC to multiple subproblems that can be solved at the cloud server in parallel. One favorable feature is that each subproblem is only associated with the decision variable at one prediction step.

To design the parallel multi-block ADMM algorithm, we first introduce an auxiliary variable \( v_k = [v_0^T, \ldots, v_{N-1}^T]^T \in \mathbb{R}^{Nn} \) and let \( u_k = b(u_k) \). The problem (11) can then be rewritten as

\[
\begin{align*}
\text{minimize} & \quad f(x_k) + g(u_k) + h^x(x_k) + h^u(u_k) \\
\text{s.t.} & \quad b(u_k) - v_k = 0 \\
& \quad \Phi_k x_0 | k + \Psi_k v_k - x_k = 0
\end{align*}
\]

(14)

After that, denote the dual variables for the constraint \( b(u_k) - v_k = 0 \) and \( \Phi_k x_0 | k + \Psi_k v_k - x_k = 0 \) by \( y_k \in \mathbb{R}^{Nn} \) and \( z_k \in \mathbb{R}^{Nn} \). The corresponding augmented Lagrangian function for (14) is as follows:

\[
\mathcal{L}_{\rho_1, \rho_2}(u_k, v_k, x_k, y_k, z_k) = f(x_k) + g(u_k) + h^x(x_k) + h^u(u_k) \\
+ y_k^T (b(u_k) - v_k) + \frac{\rho_1}{2} \| b(u_k) - v_k \|^2 \\
+ z_k^T (\Phi_k x_0 | k + \Psi_k v_k - x_k) + \frac{\rho_2}{2} \| \Phi_k x_0 | k + \Psi_k v_k - x_k \|^2,
\]

(15)

with \( \rho_1, \rho_2 > 0 \) being penalty parameter choices by designers.

The proposed parallel multi-block ADMM algorithm for solving the problem (14) consists of the following iterations. For \( i = 0, 1, \ldots, N - 1 \),

\[
\begin{align*}
u_{i+1}^j &= \text{argmin}_{u_{i+1}^j} \mathcal{L}_{\rho_1, \rho_2}(u_k, v_k^j, x_k^j, y_k^j, z_k^j) + \Delta \phi_1 \left( u_{i+1}^j, u_{i+1}^j \right) \\
&= \text{argmin}_{u_{i+1}^j} \left\{ g(u_{i+1}^j) + h^u(u_{i+1}^j) + \left[ y_{i+1}^j \right]^T (b(u_{i+1}^j) - v_k^j) + \frac{\rho_1}{2} \| b(u_{i+1}^j) - v_k^j \|^2 + \Delta \phi_1 (u_{i+1}^j, u_{i+1}^j) \right\}, \quad (16a)
\end{align*}
\]

\[
\begin{align*}v_{i+1}^j &= \text{argmin}_{v_k} \mathcal{L}_{\rho_1, \rho_2}(u_k^{i+1}, v_k, x_k^j, y_k^j, z_k^j) + \Delta \phi_2 \left( v_k^j, v_k^j \right) \\
&= \text{argmin}_{v_k} \left\{ \left[ y_k^j \right]^T (b(u_k^{i+1}) - v_k) + \frac{\rho_1}{2} \| b(u_k^{i+1}) - v_k \|^2 + \left[ z_k^j \right]^T (\Phi_k x_0 | k + \Psi_k v_k - x_k^j) \\
+ \frac{\rho_2}{2} \| \Phi_k x_0 | k + \Psi_k v_k - x_k^j \|^2 + \Delta \phi_2 (v_k^j, v_k^j) \right\}, \quad (16b)
\end{align*}
\]
\[
x_{i+1}^{j+1} = \text{argmin}_{x_{i+1}^{j+1}} \mathcal{L}_{\rho_1, \rho_2} \left( u_k^{j+1}, v_k^{j+1}, x_k, y_k^j, z_k^j \right) + \Delta \phi_3 \left( x_{i+1}^{j+1}, x_{i+1}^{j+1} \right)
\]

\[
= \text{argmin}_{x_{i+1}^{j}} \left\{ f_{i+1}(x_{i+1}) + h_{i+1}(x_{i+1}) + \left[ z_{i+1}^j \right]^\top \left( \Psi_{k,i+1} v_{k,i+1} + \Phi_{k,i+1} x_{i+1} - x_{i+1} \right) \right\},
\]

\[
y_{i+1}^{j+1} = y_{i+1}^j + \rho_1 \left( b \left( u_{i+1}^j \right) - v_{i+1}^j \right),
\]

\[
z_{i+1}^{j+1} = z_{i+1}^j + \rho_2 \left( \Phi_{k,x_{i+1}} v_{k,x_{i+1}} - x_{i+1} \right),
\]

where \( j \) is the iteration counter, \( \Delta \phi_3 \) is the Bregman distance given in Definition 4 with respect to function \( \phi_3 \), \( \lambda = 1, 2, 3 \), and \( \phi_3 \) is chosen to be strongly convex. To guarantee the convergence of the parallel multi-block ADMM algorithm, different Bregman distances are appended here to the corresponding subproblems, helping to limit the differences between the intermediate solutions between two iterations. Take \( \Delta \phi_1 \left( u_{i,jk}, u_{i,jk}^j \right) \) as an example.

It can be chosen as \( \frac{1}{2} \left\| u_{i,jk} - u_{i,jk}^j \right\|^2 \). It is worthy to note that the \( u_k \) and \( x_k \)-updates decouple across the subvectors of \( u_k \) and \( x_k \) and \( \mathcal{L}_{\rho_1, \rho_2} \) is fully separable in their components giving the credit to the introduction of the auxiliary optimization variable \( u_k \). Specifically, instead of updating \( u_k \) or \( x_k \) as a whole, the \( u_k \) and \( x_k \)-updates can be carried out independently in parallel over the cloud by (16d) and (16e), with the subvectors \( u_{i,jk} \) and \( x_{i,jk} \) being updated by \( N \) separate minimizations for each \( i \). Moreover, the iteration of \( v_k \) in (16b) can be expressed as an explicit expression and the dual updates (16d) and (16e) are only direct and simple algebraic calculations. By implementing this parallel computing algorithm, the computational MPC can handle large computational loads caused by high dimensional optimization problems with a long prediction horizon and improve the computational efficiency.

### B. Convergence of the proposed algorithm

This subsection is dedicated to the convergence analysis of the parallel multi-block ADMM for nonconvex problems with nonlinear constraints. We first fix some notations, then make some necessary assumptions, and lastly state and substantiate a series of theoretical results.

To simplify the presentation, we define the following symbols and operators

\[
G_i(u_{i,jk}) = g_k(u_{i,jk}) + h_i^u(u_{i,jk}), \quad i = 0, 1, \ldots, N - 1
\]

\[
G(u_k) = [G_0(u_{0,jk}), \ldots, G_{N-1}(u_{N-1,jk})] \in \mathbb{R}^N
\]

\[
\partial B(u_k) = \text{diag} \left( \partial b^\top(u_{0,jk}), \ldots, \partial b^\top(u_{N-1,jk}) \right) \in \mathbb{R}^{N_{\text{x}}, \times N_{\text{x}}}
\]

\[
F_{i+1}(x_{i+1}) = f_{i+1}(x_{i+1}) + h_{i+1}(x_{i+1}), \quad i = 0, 1, \ldots, N - 1
\]

\[
F(x_k) = [F_0(x_{1,jk}), \ldots, F_N(x_{N,jk})] \in \mathbb{R}^N
\]

\[
w_k = (u_k, v_k, x_k, y_k, z_k)
\]

\[
q_k = (u_k, v_k, x_k)
\]

\[
\|w_k\| = (\|u_k\|^2 + \|v_k\|^2 + \|x_k\|^2 + \|y_k\|^2 + \|z_k\|^2)^{1/2}
\]

\[
\|w_k\|_1 = \|u_k\| + \|v_k\| + \|x_k\| + \|y_k\| + \|z_k\|
\]

where \( \partial B \) is the Jacobian matrix of \( B \).

Before analyzing the convergence, we assume that

(A1) There exists a proper lower semicontinuous function \( \Omega \) which has the K-L property;

(A2) \( F \) in (21) is continuously differentiable so that \( \nabla F \) is \( L \)-Lipschitz continuous;

(A3) \( \phi_\lambda \) is \( \alpha_\lambda \)-strongly convex and \( \nabla \phi_\lambda \) is \( L_\lambda \)-Lipschitz continuous for \( \lambda = 1, 2, 3 \);

(A4) Define \( \alpha = \min \{ \alpha_1, \alpha_2, \alpha_3 \} \) and parameters \( \alpha, \rho_1, \rho_2, L, L_2, L_3 \) are chosen to satisfy

\[
\frac{\alpha}{2} > 2 \left( \frac{3}{\rho_1} \left( 3 \lambda^2 + L_3^2 + \rho_2^2 \right) + \frac{3}{\rho_2} \left( L_2^2 + L_3^2 \right) + \frac{3}{\rho_1} L_2^2 \right)
\]
The function $\Omega(\cdot)$ is defined by
\[
\Omega(\hat{\mathbf{w}}_k) = \mathcal{L}_{\rho_1, \rho_2}(\mathbf{u}_k, \mathbf{v}_k, \mathbf{x}_k, \mathbf{y}_k, \mathbf{z}_k) + \frac{\tau_1}{2} \|\mathbf{v}_k - \hat{\mathbf{v}}_k\|^2 + \frac{\tau_2}{2} \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2
\] (27)
where
\[
\hat{\mathbf{w}}_k = (\mathbf{u}_k, \mathbf{v}_k, \mathbf{x}_k, \mathbf{y}_k, \mathbf{z}_k, \hat{\mathbf{v}}_k, \hat{\mathbf{x}}_k)
\]
\[
\frac{\tau_1}{2} = \frac{3}{\rho_1} L_2^2
\]
\[
\frac{\tau_2}{2} = \frac{9\|\Psi_k\|^2}{\rho_1} (3L_2^2 + \rho_2^2) + \frac{3}{\rho_2} L_3^2
\]
where $\hat{\mathbf{v}}_k$ and $\hat{\mathbf{x}}_k$ will be specified in Lemma 2.

Below a set of lemmas is given to prove the convergence of the procedure (16). Lemma 1 is at the core of the convergence analysis.

**Lemma 1.** [32] If $\Omega$ has the K-L property and $\{\hat{\mathbf{w}}_k^j\}$ is a sequence that satisfies conditions (C1)-(C3),

(C1) (Sufficient decrease condition). For each $j \in \mathbb{N}$, there exists a fixed positive constant $a$, such that $\Omega(\hat{\mathbf{w}}_k^{j+1}) \leq \Omega(\hat{\mathbf{w}}_k^j) - a\|\hat{\mathbf{w}}_k^{j+1} - \hat{\mathbf{w}}_k^j\|^2$;

(C2) (Relative error condition). For each $j \in \mathbb{N}$, there exist a fixed positive constant $\ell$ and $c \in \partial \Omega(\hat{\mathbf{w}}_k^{j+1})$, such that $\|c\| \leq \ell\|\hat{\mathbf{w}}_k^{j+1} - \hat{\mathbf{w}}_k^j\|$;

(C3) (Continuity condition). There exists a subsequence $\{\hat{\mathbf{w}}_k^{j^*}\}$ converging to $\hat{\mathbf{w}}_k^*$ such that $\Omega(\hat{\mathbf{w}}_k^{j^*}) \to \Omega(\hat{\mathbf{w}}_k^*)$ as $\ell \to \infty$.

Then $\{\hat{\mathbf{w}}_k^j\}$ converges to $\hat{\mathbf{w}}_k^*$, i.e., a stationary point of $\Omega$. Furthermore, the sequence $\{\hat{\mathbf{w}}_k^j\}$ has a finite length, i.e., $\sum_{j=1}^{\infty} \|\hat{\mathbf{w}}_k^{j+1} - \hat{\mathbf{w}}_k^j\|_1 < \infty$.

For the detailed proof of Lemma 1 we refer readers to [32]. The following lemma shows that the constructed function $\Omega$ in (27) satisfies Condition (C1). The proof is given in Appendix A.

**Lemma 2.** There exists $a > 0$ such that $\Omega(\hat{\mathbf{w}}_k^{j+1}) \leq \Omega(\hat{\mathbf{w}}_k^j) - a\|\hat{\mathbf{w}}_k^{j+1} - \hat{\mathbf{w}}_k^j\|^2$ for each $j \in \mathbb{N}$.

Before moving on, we give a formal definition of the stationary point for problem (14).

**Definition 5.** (Stationary point) [6]. Define $\mathbf{w}_k^* = (\mathbf{u}_k^*, \mathbf{v}_k^*, \mathbf{x}_k^*, \mathbf{y}_k^*, \mathbf{z}_k^*)$ as the stationary point for problem (14) if $\mathbf{w}_k^*$ meets the primal feasibility
\[
0 = \Phi_k \mathbf{x}_0 + \Psi_k \mathbf{v}_k^* - \mathbf{x}_k^*, \quad (28a)
\]
\[
0 = \mathbf{b}(\mathbf{u}_k^*) - \mathbf{v}_k^*, \quad (28b)
\]
and the dual feasibility
\[
0 \in \partial G(\mathbf{u}_k^*) + [\partial B(\mathbf{u}_k^*)]^\top \mathbf{y}_k^*, \quad (29a)
\]
\[
0 \in -\mathbf{y}_k^* + \Psi_k \mathbf{z}_k^*, \quad (29b)
\]
\[
0 \in \partial F(\mathbf{x}_k^*) - \mathbf{z}_k^*, \quad (29c)
\]
which are the necessary and sufficient optimality conditions for the problem (14).

Below we show that each bounded sequence $\{\mathbf{w}_k^j\}$ generated by procedure (16) converges to a stationary point of $\mathcal{L}_{\rho_1, \rho_2}$.

**Lemma 3.** The sequence $\{\mathbf{w}_k^j\}$ is convergent, i.e., $\sum_{j=1}^{\infty} \|\mathbf{w}_k^{j+1} - \mathbf{w}_k^j\|^2 < \infty$. Additionally, $\{\mathbf{w}_k^j\}$ satisfies the asymptotic regularity, that is, $\lim_{j \to \infty} \|\mathbf{w}_k^{j+1} - \mathbf{w}_k^j\| = 0$. Furthermore, any accumulation point of $\{\mathbf{w}_k^j\}$ is a stationary point of the augmented Lagrangian function $\mathcal{L}_{\rho_1, \rho_2}$.
Proof: On account of (A.5) and (A3), \( \|z^j_k\| \) is bounded by
\[
\|z^j_k\| \overset{A.5}{\leq} \|\nabla F(x^j_k) + \nabla \phi_3(x^j_k) - \nabla \phi_3(x^{j-1}_k)\| \\
\leq \|\nabla F(x^j_k)\| + \|\nabla \phi_3(x^j_k) - \nabla \phi_3(x^{j-1}_k)\| \\
\overset{(A3)}{\leq} \|\nabla F(x^j_k)\| + L_3\|x^j_k - x^{j-1}_k\| 
\]
Due to the boundness of \( \{x^j_k\} \) in (4d) and the continuity of \( \nabla F(x^j_k) \) in (A2), we can obtain that \( \|z^j_k\| \) is bounded. Similarly, according to (A.2) and (A3), we derive the bound on \( \|y^j_k\| \)
\[
\|y^j_k\| \overset{(A.2)}{\leq} \|\Psi_k^T(z^j_k + \rho_2(x^j_k - x^{j-1}_k)) + \nabla \phi_2(v^j_k) - \nabla \phi_2(v^{j-1}_k)\| \\
\leq \|\Psi_k^T z^j_k\| + \rho_2\|\Psi_k(x^j_k - x^{j-1}_k)\| + \|\nabla \phi_2(v^j_k) - \nabla \phi_2(v^{j-1}_k)\| \\
\overset{(A3)}{\leq} \|\Psi_k^T z^j_k\| + \rho_2\|\Psi_k(x^j_k - x^{j-1}_k)\| + L_2\|v^j_k - v^{j-1}_k\| 
\]
Since \( \{u^j_k\} \) is bounded by (4d) and \( b(\cdot) \) is continuous, it yields that \( \{v^j_k\} \) is bounded. Hence \( \{y^j_k\} \) is bounded, and so are \( \{w^j_k\} \) and \( \{\hat{w}^j_k\} \). Thus there is a subsequence \( \{\hat{w}^{j\prime}_k\} \) of \( \{w^j_k\} \) that converges to \( \hat{w}_k \). According to our assumption, the function \( \Omega \) is lower semicontinuous, which results in \( \lim \inf_{\hat{w}^{j\prime}_k \to \hat{w}_k} \Omega(\hat{w}^{j\prime}_k) \geq \Omega(\hat{w}_k) \), and further \( \Omega(\hat{w}_k) \) is bounded from below. By Lemma 2, \( \Omega(\hat{w}_k) \) is nonincremental, and consequently convergent. Due to
\[
\|\hat{w}^{j+1}_k - \hat{w}^j_k\|^2 \leq \|q^{j+1}_k - q^j_k\|^2 + \|y^{j+1}_k - y^j_k\|^2 + \|z^{j+1}_k - z^j_k\|^2 + \|w^j_k - w^{j-1}_k\|^2 + \|x^j_k - x^{j-1}_k\|^2, 
\]

\[
\text{it can be inferred that} \\
\|\hat{w}^{j+1}_k - \hat{w}^j_k\|^2 \geq \|q^{j+1}_k - q^j_k\|^2, 
\]
Using Lemma 2 the upper bound on the sum of \( \|q^{j+1}_k - q^j_k\|^2 \) can be derived
\[
a \sum_{j=1}^{\infty} \|q^{j+1}_k - q^j_k\|^2 \leq a \sum_{j=1}^{\infty} \|\hat{w}^{j+1}_k - \hat{w}^j_k\|^2 \\
\leq \Omega(\hat{w}^1_k) - \Omega(\hat{w}_k^2) + \Omega(\hat{w}^2_k) - \Omega(\hat{w}_k^3) + \ldots + \Omega(\hat{w}^j_k) - \Omega(\hat{w}^{j+1}_k) + \ldots \\
= \Omega(\hat{w}^1_k) - \Omega(\hat{w}^\infty_k) \\
\leq \Omega(\hat{w}^1_k) - \Omega(\hat{w}^\infty_k) 
\]
where the last inequality follows from that \( \Omega(\hat{w}^j_k) \geq \Omega(\hat{w}^\infty_k) \) for each \( j \) by Lemma 2. This leads to
\[
\sum_{j=1}^{\infty} \|w^{j+1}_k - w^j_k\|^2 \leq \sum_{j=1}^{\infty} \|\hat{w}^{j+1}_k - \hat{w}^j_k\|^2 < \infty, 
\]
and therefore the sequence \( \{w^j_k\} \) satisfies the convergence. In particular, \( \{w^j_k\} \) is asymptotically regular, that is,
\[
\lim_{j \to \infty} \|\hat{w}^{j+1}_k - \hat{w}^j_k\| = 0. 
\]
Defining \( w^*_k = (u^*_k, v^*_k, x^*_k, y^*_k, z^*_k) \), a point of \( \{w^j_k\} \), and supposing that \( \{w^{j\prime}_k\} \) is a subsequence of \( \{w^j_k\} \) converging to \( w^*_k \), on the basis of (16) and (A.2)-(A.5), it yields that
\[
0 \in \partial G(u^{j+1}_k) + \left[ \partial B(u^{j+1}_k) \right]^T y^{j+1}_k + \rho_1 \left[ \partial B(u^{j+1}_k) \right]^T (v^{j+1}_k - v^j_k) + \nabla \phi_1(u^{j+1}_k) - \nabla \phi_1(u^j_k) \\
0 \in -y^{j+1}_k + \Psi_k^T (z^{j+1}_k + \rho_2(x^{j+1}_k - x^j_k)) + \nabla \phi_2(v^{j+1}_k) - \nabla \phi_2(v^j_k) \\
0 \in \nabla F(x^{j+1}_k) - z^{j+1}_k + \nabla \phi_3(x^{j+1}_k) - \nabla \phi_3(x^j_k) \\
y^{j+1}_k = y^j_k + \rho_1 (b(u^{j+1}_k) - v^{j+1}_k) \\
z^{j+1}_k = z^j_k + \rho_2 (\Phi_k x_0 + \Psi_k v^{j+1}_k - x^j_k) 
\]
In virtue of the continuity of $\nabla \phi_\lambda, \lambda = 1, 2, 3$ and the asymptotic regularity of $\{w_k^j\}$ by (30), it can be concluded that $w_k^\ast$ satisfies the primal feasibility (28) and the dual feasibility (29) when $d \to \infty$, and therefore $w_k^\ast$ is a stationary point for problem (14).

Now two lemmas are provided to show the satisfaction of conditions (C2) and (C3) in Lemma 1. For ease of reading, proofs of Lemma 4 and 5 are given in Appendix B and C.

**Lemma 4.** There exists $\ell > 0$ and $c \in \partial \Omega(w_k^{j+1})$, such that $\|c\| \leq \ell \|w_k^{j+1} - w_k^j\|$.

**Lemma 5.** There exists a subsequence $\{w_k^j\}$ converging to $w_k^\ast$ such that $\Omega(w_k^j) \to \Omega(w_k^\ast)$ as $d \to \infty$.

Considering Lemmas 4, 5, and 6 together, the sequence $\{w_k^j\}$ is verified to satisfy conditions (C1)-(C3). From Lemma 4 and Lemma 5, we can obtain the main theorem of this paper.

**Theorem 1.** (Convergence) If $\Omega$ has the K-L property and $\{w_k^j\}$ is a sequence that satisfies sufficient decrease condition, relative error condition and continuity condition, then $\{w_k^j\}$ converges to $w_k^\ast$, which is a stationary point of $\Omega$. Especially, $\{w_k^j\}$ generated by the process (16) converges to $w_k^\ast$, a stationary point of $\mathcal{L}_{\rho_1, \rho_2}$. In addition, $\{w_k^j\}$ has a finite length, i.e., $\sum_{j=1}^\infty \|w_k^{j+1} - w_k^j\|^2 < \infty$.

In the practical implementation, the stopping criterion for the parallel multi-block ADMM algorithm is required. According to (31)-(33), the residual variables can be defined by

$$\begin{align*}
r_k^{j+1} &= \begin{bmatrix}
\Phi_k x | x_0 | + \Psi_k u_k^{j+1} - x_k^{j+1} \\
| b u_k^{j+1} |
\end{bmatrix} \\
x_k^{j+1} &= \begin{bmatrix}
v_k^{j+1} - v_k^j \\
x_k^{j+1} - x_k^j
\end{bmatrix}
\end{align*}$$

where $r_k^{j+1}$ and $x_k^{j+1}$ are the primal residual and the dual residual at iteration $j$, respectively. The iteration (16) converges to a point satisfying the first order necessary conditions (28)-(29) if and only if the residual variables converge to zero. It can also be terminated when primal and dual residual variables have fallen below predefined thresholds, $\epsilon_{\text{pri}}$ and $\epsilon_{\text{dual}}$, which are chosen based on the scale of the typical variable values. In this paper, the stopping criterion is set as

$$\begin{align*}
\|r_k^{j+1}\| &\leq \epsilon_{\text{pri}} \\
\|s_k^{j+1}\| &\leq \epsilon_{\text{dual}}
\end{align*}$$

(36a) (36b)

where $\epsilon_{\text{pri}} > 0$ and $\epsilon_{\text{dual}} > 0$ are feasibility tolerances for the primal and dual feasibility conditions (28) and (29).

Now we summarize the computational MPC algorithm based on the parallel multi-block ADMM for nonconvex problems with nonlinear constraints in Algorithm 1. It is worth pointing out that calculating $\{u_i|k\}_{i=0}^{N-1}$ and $\{x_i|k\}_{i=1}^N$ in parallel can be operated in the cloud environment based on Docker containers technique, which will be elaborated in the following section.

**IV. CLOUD-BASED COMPUTATIONAL MODEL PREDICTIVE CONTROLLER**

Cloud computing has recently become a rapidly available and widely developed service offering, which is an attractive and cost-effective option when clients lack the computational resources and/or expertise to perform computation locally. Taking into consideration the restrictions on local computational resources, an innovative architecture called cloud-based computational model predictive controller suitable for cloud computing environment is devised to implement the proposed computational MPC. The architecture is depicted in Fig. 2 which concretizes Fig. 1.

The architecture is composed of local controlled plants and a cloud-based model predictive controller at the cloud server. The cloud-based controller consists of three main modules: (a) model predictive controller module, (b) Docker containers module, (c) Redis module.

As shown in Fig. 2 and Algorithm 1, at one sample step, the local controlled plants send their actual states to the model predictive controller at the cloud server. Incorporating a prediction model, a cost function, and constraints into an optimization problem, the controller solves it using the designed parallel multi-block ADMM algorithm.
Algorithm 1 Computational MPC using the parallel multi-block ADMM

1: Initialize controller parameters: $\rho_1, \rho_2, \epsilon_{\text{pri}}, \epsilon_{\text{dual}} > 0$; Determine Bregman distance $\Delta_{\phi_k}$ with $\lambda = 1, 2, 3$; Choose the prediction horizon $N$.
2: Set $k = 0$.
3: Measure the system state $x_k$, and let $x_{0|k} = x_k$, the index $j \leftarrow 1$.
4: while stopping criterion (36) is not satisfied do
5: for all $i = 0, 1, \ldots, N - 1$ (based on Docker containers in parallel) do
6: Obtain $u_{i|k}^{j+1}$ from (16a);
7: end for
8: Calculate $v_{k}^{j+1}$ using (16b);
9: for all $i = 1, 2, \ldots, N$ (based on Docker containers in parallel) do
10: Solve for $x_{i+1|k}^{j+1}$ with (16c);
11: end for
12: Update $y_{k}^{j+1} \leftarrow y_{k}^{j} + \rho_1(b(u_{k}^{j+1}) - v_{k}^{j+1})$;
13: Update $z_{k}^{j+1} \leftarrow z_{k}^{j} + \rho_2(\Phi_k x_0 + \Psi_k v_{k}^{j+1} - x_{k}^{j+1})$;
14: $j \leftarrow j + 1$.
15: end while
16: Set the solution $u_{0|k}^{*} = u_{0|k}^{j+1}$ and apply $u(k) = u_{0|k}^{*}$ to the controlled plant.
17: Wait for next sampling time, let $k \leftarrow k + 1$ and return to step 3.

Through this parallel computing algorithm, the decomposed multiple subproblems (16a) and (16c) are solved on corresponding containers in parallel. The iterative process (16) continues until the stopping criterion is satisfied. And data communication between the controller and Docker containers is conducted through Redis, which caches all iterative intermediate results. Finally, the optimal solution is sent back to the local controlled plants. Below, the working mechanism of these three components is explained in detail.

A. Model predictive controller module

The model predictive controller consists of a model part that contains the system prediction model of the controlled plants and the optimization part that contains a cost function, boundary constraints, and optimization algorithm. The controller needs to receive the actual states and solve MPC optimization problems.

More precisely, at one sample step, the model part of the controller receives the actual states of the controlled plants via the Transmission Control Protocol (TCP) and provides the system model of the controlled plants to the optimization part for prediction.

In the optimization part, we adopt the proposed parallel multi-block ADMM algorithm, namely, looping the iterations of (16) to solve the optimization problem until the stopping criterion is satisfied. The proposed algorithm consists of an $u_{ijk}$-minimization step (16a) for all $i$, an auxiliary optimization variable $v_k$ update (16b), an $x_{i+1|k}$-minimization step (16c) for all $i$, and two dual variables updates (16d) and (16e). The variable values updated in each iteration could be stored in Redis, since the containers have to use the variable values updated in the last iteration when solving $u_{ijk}$ and $x_{i+1|k}$. To carry out the first and third steps in parallel, we integrate the ThreadPool with Hyper Text Transfer Protocol (HTTP).

ThreadPool can be thought of as a container for threads, which is suitable for the execution of several short tasks in parallel without interfering with each other, such as solving $u_{ijk}$ and $x_{i+1|k}$. HTTP is a stateless protocol, which means that the server does not maintain any state of transactions with clients. This significantly reduces the memory load on the server and thus may achieve faster response times. A client opens a socket on a specific TCP port, and if the server continues to wait for a connection on that known port, the connection is made. The client then sends a request block containing the request method to the server over this connection.
Fig. 2: Architecture of the cloud-based computational model predictive controller

In the $u_i|k-$ and $x_i|k-$ updates for all $i$, we first create multiple threads with ThreadPool and write Uniform Resource Locators (URLs) that contain each container execution signal into a list. Then, use the Python built-in function `map` to map the list according to the `request.get`, which is a request method of HTTP. The optimization part next receives the response status code indicating the end of the container operation, and the next variable update is carried out. Since the ports of each container have been mapped to different ports on the server, we only need to send execution signals to these mapped ports. Each container, which has been started and is monitoring the state of its port, executes its instance to solve the sub-optimization problem.

Finally, when the stopping criterion is satisfied, the optimal solution to the MPC optimization problem obtained as control actions is sent back to the controlled plants.

B. Docker containers module

Docker containers are introduced to make full use of cloud computing resources to implement the $u_i|k-$ and $x_i|k-$ updates independently in parallel for each $i$. The Docker containers module consists of three parts:

1. a registry storing the basic image composed of all needed Python packages,
2. a filesystem containing Python files and script files for solving optimization problems,
3. multiple containers, which are isolated environments to solve the sub-optimization problems (16a) and (16c).

Using Docker containers to realize the parallel solutions is roughly a three-step process.

The preparatory work

Before sampling, use Compose[^1] to complete container deployment, including the edition of three files: (1) a Dockerfile for the basic image, (2) scripts and Python files, (3) a container configuration file, and start multiple containers in advance.

[^1]: https://docs.docker.com/compose/
Compose is a tool for defining and running multi-container Docker applications. The basic image containing all packages used to solve sub-optimization problems is built with a Dockerfile and pushed to the registry. The scripts and Python files in the filesystem, which are executed when the containers solve subproblems, are stored at the cloud server. The container configuration file called `docker-compose.yml` is a YAML file to configure the application’s services. The file `docker-compose.yml` defines the services, the basic image of the containers, ports mapping, volumes mounting paths of the scripts and Python files, and the bridge network mode. In Fig. 2, containers are represented by the stacked yellow boxes labeled Container, the services are indicated by the stacked light blue boxes labeled Service, which include the names of each container, and ports mapping are indicated by the green dots.

After the preparation of these configuration files, entire containers are started simultaneously by a Docker compose command, i.e., `docker-compose up`. At this point, the containers switch on monitoring port mode and wait for execution signals from the optimization part. These ports mapped to the cloud server must be different from each other, represented in Fig. 2 by dots with different colors on the edges of the Docker module box. Preparing and starting containers in advance can significantly reduce the time it takes to deploy containers.

**During solving subproblems**

At one sample step $k$, after the model predictive controller receives the actual states of the controlled plants, in order to execute the $u_{i|k}$- and $x_{i+1|k}$-updates for each $i$, the controller triggers the corresponding containers by sending execution signals simultaneously to the mapped ports on the server via HTTP. Once the containers detect the execution signals, they solve the $u_{i|k}$- or $x_{i+1|k}$-subproblems by (16a) or (16c) independently in parallel for each $i$. During the solution process, some needed data such as the controller parameters can be retrieved in Redis and the updated values of the iteration are also stored in Redis. After the parallel solution via containers is completed, the status code indicating the end of the container operation is returned to the controller by HTTP.

The optimization cycles the $u_{i|k}$-, $v_{k}$-, $x_{i+1|k}$-, $y_{k}$- and $z_{k}$-updates until the stopping criterion is satisfied, and the optimal solution corresponding to the current sampling time is then obtained.

**After solving subproblems**

The first element of the optimal control input sequence for the optimization problem is sent back and applied to the controlled plants. At the sample step $k + 1$, continue to sample the actual states of the plants.

**C. Redis module**

As an open-source in-memory data structure store, Redis serves as a database, cache, and message broker. In this paper, Redis is used as a shared data pool for two reasons. For one thing, Redis is convenient to invoke the system to transmit the intermediate variables. This feature is useful for accessing small-sized data with high frequency in the solution of MPC optimization, and therefore suitable for the situation in our paper where the original optimization problem is decomposed into multiple small-sized sub-problems. For another, Redis works as a shared data pool for all containers. That means the users could avoid designing and building complex communication topologies. All the read and write operations of data could be finished in this data pool.

Through the use of Redis, all containers can build information-sharing channels with the cloud server. In the process of solving the optimization problem, the model predictive controller writes controller parameters to Redis and accesses the optimization variables updated from Redis. In the Docker containers module, each container obtains controller parameters from Redis and stores the value of the optimization variable in each iteration in Redis.

**V. Simulation Results and Analysis**

This section presents a proof-of-concept implementation via five plug-in hybrid electric vehicles (PHEVs) charging. The cloud-based control architecture of energy management for PHEVs charging is shown in Fig. 3. In contrast to previous approaches implementing control algorithms locally, in our demonstration setup, we host the model predictive controller on the remote pay-as-you-go Alibaba Elastic Cloud Server (ECS) with Container Service via standard communication protocols (TCP and HTTP). The model of PHEVs is simulated by using Python 3.6.10 on a PC with an eight-core Intel (R) Core (TM) CPU running at 4.00 GHz and 8.00 GB RAM. The controlled PHEVs send their charge states over TCP to the cloud, which performs the MPC optimization problem computation and returns the optimal control inputs.

From [34] and [35], the five vehicles system can be modelled as
Fig. 3: Cloud-based control architecture of energy management for PHEVs charging

\[ x_{k+1} = A_k x_k + b(u_k), \]  

(37)

where \( x_k = [x_k^{[1]}, x_k^{[2]}, x_k^{[3]}, x_k^{[4]}, x_k^{[5]}]^\top \) and \( u_k = [u_k^{[1]}, u_k^{[2]}, u_k^{[3]}, u_k^{[4]}, u_k^{[5]}]^\top \) represent the State of Charge (SoC) of five vehicles and the available power allocated at the current sample step \( k \), respectively, the state matrix \( A_k \in \mathbb{R}^{5 \times 5} \) is an identity matrix, and \( b(u_k) \) is a vector of five functions of the allocated powers to five vehicles, i.e., \( b(u_k) = [b(u_k^{[1]}), b(u_k^{[2]}), b(u_k^{[3]}), b(u_k^{[4]}), b(u_k^{[5]})]^\top \). Each function of the allocated power to every vehicle is given by

\[ b(u_k^{[m]}) = \frac{C^{[m]} Q^{[m]} (-V^{[m]} + \sqrt{V^{[m]}^2 + 2\eta^{[m]} u_k^{[m]} \Delta t})}{C^{[m]}} \]  

(38)

where for \( m = 1, 2, 3, 4, 5 \), \( C^{[m]} \) is the equivalent capacitance of the \( m \)th battery, \( Q^{[m]} \) is the capacity of the \( m \)th battery in Coulombs, \( V^{[m]} \) is the voltage of the \( m \)th battery at the current sample step \( k \), \( \Delta t \) is a fixed cycle time for discretizing the system, and \( \eta^{[m]} \) represents the efficiency of the charging. The limit to the total system is that the total power allocated to five vehicles cannot exceed the upper bound of the utility power delivery, \( u_{\text{total}} \), and it is formulated as follows

\[ \sum_{m=1}^{5} u_k^{[m]} \leq u_{\text{total}}. \]  

(39)

And the individual constraint for each vehicle is

\[ U^{[m]} = \{ u_k^{[m]} \in \mathbb{R}^1 : 0 \leq u_k^{[m]} \leq \tilde{u}_k^{[m]} \}, \quad m = 1, 2, 3, 4, 5 \]  

(40)

\[ X^{[m]} = \{ x_k^{[m]} \in \mathbb{R}^1 : 0\% \leq x_k^{[m]} \leq 100\% \}, \quad m = 1, 2, 3, 4, 5 \]  

(41)

where \( U^{[m]} \) and \( X^{[m]} \) are constraint sets for the SoC and the available power allocated. \( \tilde{u}_k^{[m]} \) is the upper bound of the allocated power taking into account all local limitations. The SoC of each vehicle varies from 0% to 100%. The cost function is to maximize the weighted sum of the SoC of five vehicles at the next time step based on the
optimally allocating available power at the current sample step $k$: \[
\max_{u_k} \sum_{m=1}^{5} \varpi[m] x_{k+1}^{[m]}
\]
where $\varpi^{[m]}$ is the willingness to pay (WtP) for the $m^{th}$ vehicle and it is a nonnegative value. Let a vector $\varpi = [\varpi[1], \varpi[2], \varpi[3], \varpi[4], \varpi[5]]^\top$ and $g(u_k)$ be the cost of charging, that is, \[
g(u_k) = -\sum_{m=1}^{5} \varpi[m] C[m] \sqrt{V[m]} + \frac{2\eta[m] u_k[m] \Delta t}{C[m]}
\]
In order for the SoC of five vehicles to reach our expected value 90% quickly, we define $f(x_k) = (x_k - x_{\text{exp}})^\top P(x_k - x_{\text{exp}})$, where $P \in \mathbb{R}^{5 \times 5}$ is a diagonal matrix having $\varpi$ on the diagonal, and $x_{\text{exp}} = [90\%, 90\%, 90\%, 90\%, 90\%]^\top$.

According to (14), we formulate the MPC optimization problem for the SoC maximization problem over a prediction horizon of $N$ steps
\[
\min_{u_k, x_k} f(x_k) + g(u_k)
\]
s.t. 
\[
x_{0|k} = x_k \nonumber \\
x_{i+1|k} = A_{i|k} x_{i|k} + b(u_{i|k}), \ i = 0, 1, \ldots, N - 1 \nonumber \\
(x_{i+1|k}, u_{i|k}) \in \mathcal{X} \times \mathcal{U}, \ i = 0, 1, \ldots, N - 1 \nonumber \\
\sum_{m=1}^{5} u_{i|k}^{[m]} \leq u_{\text{total}}, \ i = 0, 1, \ldots, N - 1 
\]
where the initial values of system states are given by $x_{0|k} = [20\%, 20\%, 20\%, 20\%, 20\%]^\top$ and
\[
f(x_k) = \sum_{i=1}^{N} f(x_{i|k}) \\
g(u_k) = \sum_{i=0}^{N-1} g(u_{i|k}) \\
\mathcal{X} = \prod_{m=1}^{5} \mathcal{X}^{[m]} \\
\mathcal{U} = \prod_{m=1}^{5} \mathcal{U}^{[m]}
\]

The parameters regarding the battery side of the vehicles and their WtP are listed in TABLE I. In addition to these system parameters, the total available power $u_{\text{total}}$ is 5000W, the charging efficiency $\eta^{[m]} = 0.9$ for each vehicle, and the fixed cycle time for discretizing the system is $\Delta t = 0.1s$. In the proposed cloud-based computational MPC, we set the Bregman distance $\Delta_{\varpi}(x, y) = \frac{1}{2} \|x - y\|^2$, and other controller parameters are chosen as follows: $\varepsilon^{\text{phil}} = \varepsilon^{\text{phil}} = 0.01$, $\rho_1 = 10$, $\rho_2 = 8$, $y_0^{[m]} = 0$, $z_0^{[m]} = 0$. Simulation is performed for 20 minutes with a sampling interval of 60s. The prediction horizon is assigned by $N = 18$. It can be observed from Fig. 4 that system states are stabilized to the desired equilibrium point $x_{\text{exp}}$ (at about 5 minutes) by implementing the cloud-

| Vehicle | $V^{[m]}(V)$ | $u_k^{[m]}(W)$ | $C^{[m]}(\text{F})$ | $Q^{[m]}(A.H)$ | $\varpi^{[m]}$ |
|---------|----------------|----------------|-------------------|----------------|----------------|
| PHEV 1  | 238            | 800            | 7200              | 20             | 0.25           |
| PHEV 2  | 240            | 1000           | 7920              | 22             | 0.3            |
| PHEV 3  | 242            | 900            | 18000             | 25             | 0.4            |
| PHEV 4  | 236            | 1100           | 8640              | 24             | 0.5            |
| PHEV 5  | 243            | 1200           | 8800              | 22             | 0.35           |

TABLE I: Parameters of the Five Vehicles
Fig. 4: States and controls of the closed-loop system with the proposed cloud-based computational MPC based computational MPC. Meanwhile, the control inputs tend to be 0 and are bounded within their constraints. The evolution of the primal and dual residual norms by iteration are depicted in Fig. 5. The dashed lines represent the stopping criterions which are all satisfied after 59 iterations. The proposed parallel multi-block ADMM algorithm is verified to ensure that the primal residual and the dual residual variables converge to zero.

Fig. 5: Convergence of the parallel multi-block ADMM with the proposed cloud-based computational MPC at $k = 1$
Additionally, the performance of the computation time to find the solution using container parallel computing is evaluated by comparing it to containerless serial computing. We adopt different lengths of prediction horizon $N$ varying from 2 to 20, the same system parameters and controller parameters during the execution of the simulation. The comparison result is shown in Fig. 6. As expected, with increasing $N$, the advantage of parallelism becomes clearer, since parallel computing with multiple containers takes less time than containerless serial computing. When $N = 2$, through the container technique, the computation time is increased by 34.14\% of that under containerless. This is because the latency for both the read and write operations using Redis slows down the MPC computation for small $N$. When $N = 4$, the parallel computing time increases to 1.89\%, due to the trade-off of the time cost between communication and the containerless serial computing. The computation time decreases with horizon length, from 1.89\% for $N = 4$, to 22.15\% for $N = 20$, which illustrates the effectiveness of the proposed method. It should also be noticed that this advancement is not an order-of-magnitude one, mostly due to the intrinsic non-convexity of the nonlinear MPC problem and inter-container communication. However, the reduction in computation time for a single iteration is expected to be more significant for larger-scale systems. These simulations intuitively show that distributing a large optimization problem to multiple containers improves the execution time by performing smaller subproblems parallelly.

VI. CONCLUSIONS

In this paper, an innovative cloud-based computational MPC scheme has been proposed for nonlinear systems with input nonlinearity to improve computational efficiency. A parallel multi-block ADMM algorithm has been tailored to speed up the solution process of obtaining the optimal control sequence of the nonconvex MPC optimization problem. In addition to its relatively low computational effort, we have formally established the convergence analysis of the parallel multi-block ADMM algorithm. Based on this algorithm, the computational MPC can decompose the NMPC optimization problem into multiple individual subproblems. These subproblems are only in relation to the decision variables at one prediction step and therefore are allowed to be simultaneously calculated in a parallel manner. Furthermore, to implement the parallel computing algorithm efficiently and alleviate local calculation loads, we have devised a cloud-based computational MPC architecture, which can solve the decomposed multiple subproblems in corresponding Docker containers in parallel. By taking explicitly into account the particularities of the cloud computing platform and the characteristics of the subproblems, the devised architectures take sufficient advantage of the cloud computing technology and resources. Experimental simulations have verified the effectiveness of the cloud-based computational MPC scheme. Also, the experimental results demonstrate that the execution time can be greatly improved using parallelization, more efficient data structures, and cloud technologies. Future research could be focused on the challenge of a potentially long latency time between field devices and the cloud server. Another interesting topic would be to expand the scheme towards encrypted MPC.
A. Proof of Lemma 2

Based on Fermat’s rule, the limiting subdifferential of (15) at $v_{i|k}^{j+1}$ and $x_{i+1|k}^{j+1}$ are derived

\[
0 \in \partial G_i(v_{i|k}^{j+1}) + \left[ \partial B(u_{i|k}^{j+1}) \right] y_i^{j+1} + \rho_i \left[ \partial B(u_{i|k}^{j+1}) \right]^T (b(u_{i|k}^{j+1}) - v_{i|k}^{j+1}) + \nabla \phi_1(v_{i|k}^{j+1}) - \nabla \phi_1(u_{i|k}^{j+1}) \tag{A.1}
\]

\[
0 \in -y_{k}^{j+1} - \rho_1 \left( b(u_{k}^{j+1}) - v_{k}^{j+1} \right) + \Psi_k^T(z_k^{j+1} + \rho_2 \Psi_k^T(\Phi_k x_{0|k} + \Psi_k v_{k}^{j+1} - x_k^{j}) + \nabla \phi_2(v_{k}^{j+1}) - \nabla \phi_2(v_{k}^{j}) \tag{A.2}
\]

\[
0 \in \nabla F_{i+1}(x_{i+1|k}^{j+1}) - z_{i|k}^{j} - \rho_2(\Phi_k x_{0|k} + \Psi_k v_{i+1|k}^{j+1} - x_{i+1|k}^{j+1}) + \nabla \phi_3(x_{i+1|k}^{j+1}) - \nabla \phi_3(x_{i+1|k}^{j}) \tag{A.3}
\]

In virtue of (16d), (18), (19) and (21), the following expressions in terms of $u_k$ and $x_k$ can be further obtained

\[
0 \in \partial G(u_{k}^{j+1}) + \left[ \partial B(u_{k}^{j+1}) \right] y_{k}^{j+1} + \rho_i \left[ \partial B(u_{k}^{j+1}) \right]^T (b(u_{k}^{j+1}) - v_{k}^{j+1}) + \nabla \phi_1(u_{k}^{j+1}) - \nabla \phi_1(u_{k}^{j+1}) \tag{A.4}
\]

\[
0 \in \nabla F(x_{k}^{j+1}) - z_{k}^{j} - \rho_2(\Phi_k x_{0|k} + \Psi_k v_{k}^{j+1} - x_k^{j}) + \nabla \phi_3(x_{k}^{j+1}) - \nabla \phi_3(x_k^{j}) \tag{A.5}
\]

where, for convenience, we permit a slight abuse of notation that $\nabla \phi_1(u_{k}^{j}) = [\nabla \phi_1(u_{0|k}), \ldots, \nabla \phi_1(u_{N-1|k})]^T$ and $\nabla \phi_3(x_{k}^{j}) = [\nabla \phi_3(x_{0|k}), \ldots, \nabla \phi_3(x_{N|k})]^T$.

For further derivation, we give the following useful inequalities. From

\[
0 \leq 2\|a - \frac{1}{2}b\|^2
\]

it holds that

\[
2\|a\||\|b\|\leq 2\|a\|^2 + \frac{1}{2}\|b\|^2 \tag{A.6}
\]

\[
\leq 2\|a\|^2 + \|b\|^2 \tag{A.7}
\]

where $a$ and $b$ are arbitrary vectors. Through applying Cauchy-Schwarz inequality, the upper bound on the difference of $z_{k}^{j}$ between adjacent iteration steps is given by

\[
\|z_{k}^{j+1} - z_{k}^{j}\|^2 \leq 3\|\nabla F(x_{k}^{j+1}) - \nabla F(x_{k}^{j})\|^2 + 3\|\nabla \phi_3(x_{k}^{j+1}) - \nabla \phi_3(x_{k}^{j})\|^2 + \|\nabla \phi_3(x_{k}^{j+1}) - \nabla \phi_3(x_{k}^{j})\|^2 \tag{A.8}
\]

where the inequalities for the last two steps come from the fact that $\nabla F$ has $L$-Lipschitz continuous property by (A2) and $\nabla \phi_3$ is $L_3$-Lipschitz continuous by (A3). Analogously, the upper bound on the norm of $y_{k}^{j+1} - y_{k}^{j}$ is
derived by using Cauchy-Schwarz inequality

\[ \| y^{j+1}_k - y^j_k \|^2 \leq \| \Psi_k^T (z^{j+1}_k - z^j_k) + \rho_2 \Psi_k^T ((x^{j+1}_k - x^j_k) - (x^{j+1}_k - x^{j+1-1}_k)) + \nabla \phi_2 (v^{j+1}_k) - \nabla \phi_2 (v^j_k) - (\nabla \phi_2 (v^{j+1}_k) - \nabla \phi_2 (v^{j-1}_k)) \|^2 \]

\[ \leq \| \Psi_k^T (z^{j+1}_k - z^j_k) + \rho_2 \Psi_k^T ((x^{j+1}_k - x^j_k) - (x^{j+1}_k - x^{j+1-1}_k)) + 2\| \nabla \phi_2 (v^{j+1}_k) - \nabla \phi_2 (v^j_k) - (\nabla \phi_2 (v^{j+1}_k) - \nabla \phi_2 (v^{j-1}_k)) \|^2 \]

\[ \leq 3 \| \Psi_k^T (z^{j+1}_k - z^j_k) + \rho_2 \Psi_k^T ((x^{j+1}_k - x^j_k) - (x^{j+1}_k - x^{j+1-1}_k)) \|^2 + \frac{3}{2} \| \nabla \phi_2 (v^{j+1}_k) - \nabla \phi_2 (v^j_k) \|^2 + \| \nabla \phi_2 (v^{j+1}_k) - \nabla \phi_2 (v^{j-1}_k) \|^2 \]

On the basis of (13) and the optimality in (16a), we have

\[ L_{\rho_1, \rho_2} (u^{j+1}_k, v^j_k, x^j_k, y^j_k, z^j_k) + \Delta \phi_1 (u^{j+1}_k, u^j_k) \leq L_{\rho_1, \rho_2} (u^j_k, v^j_k, x^j_k, y^j_k, z^j_k) + \Delta \phi_1 (u^j_k, u^j_k) \]

\[ L_{\rho_1, \rho_2} (u^{j+1}_k, v^j_k, x^j_k, y^j_k, z^j_k) \leq L_{\rho_1, \rho_2} (u^j_k, v^j_k, x^j_k, y^j_k, z^j_k) \]

with \( \Delta \phi_1 (u^j_k, u^j_k) = 0 \) by (12). By following the similar way,

\[ L_{\rho_1, \rho_2} (u^{j+1}_k, v^{j+1}_k, x^{j+1}_k, y^{j+1}_k, z^{j+1}_k) \leq L_{\rho_1, \rho_2} (u^{j+1}_k, v^{j+1}_k, x^{j+1}_k, y^{j+1}_k, z^{j+1}_k) - \frac{\alpha}{2} \| u^{j+1}_k - u^j_k \|^2 \]

\[ L_{\rho_1, \rho_2} (u^{j+1}_k, v^{j+1}_k, x^{j+1}_k, y^{j+1}_k, z^{j+1}_k) \leq L_{\rho_1, \rho_2} (u^{j+1}_k, v^{j+1}_k, x^{j+1}_k, y^{j+1}_k, z^{j+1}_k) + \frac{1}{\rho_1} \| y^{j+1}_k - y^j_k \|^2 + \frac{1}{\rho_2} \| z^{j+1}_k - z^j_k \|^2 \]

Letting \( q^j_k = (u^j_k, v^j_k, x^j_k) \) and summing up the above inequalities (A.10)-(A.14), we have

\[ L_{\rho_1, \rho_2} (w^{j+1}_k) \leq L_{\rho_1, \rho_2} (w^j_k) - \frac{\alpha}{2} \| u^{j+1}_k - u^j_k \|^2 - \frac{\alpha}{2} \| v^{j+1}_k - v^j_k \|^2 + \frac{1}{\rho_1} \| y^{j+1}_k - y^j_k \|^2 + \frac{1}{\rho_2} \| z^{j+1}_k - z^j_k \|^2 \]

Substituting (A.8) and (A.9) into (A.15), it yields that
\[ \mathcal{L}_{\rho_1, \rho_2}(w_k^{j+1}) \leq \mathcal{L}_{\rho_1, \rho_2}(w_k^j) - \frac{\alpha}{2} \| q_k^{j+1} - q_k^j \|^2 \]

\[ + \frac{9}{\rho_1} (3L_2^2 + 3L_3^2 + \rho_2^2) \| \Psi_k \|^2 \| x_k^{j+1} - x_k^j \|^2 + \frac{3L_2^2}{\rho_1} \| v_k^{j+1} - v_k^j \|^2 \]

\[ + \frac{9}{\rho_1} (3L_2^2 + \rho_2^2) \| \Psi_k \|^2 \| x_k^j - x_k^{j-1} \|^2 + \frac{3L_2^2}{\rho_1} \| v_k^j - v_k^{j-1} \|^2 \]

\[ + \frac{3(L_2^2 + L_3^2)}{\rho_2} \| x_k^{j+1} - x_k^j \|^2 + \frac{3L_2^2}{\rho_2} \| x_k^j - x_k^{j-1} \|^2 \]  

(A.16)

Let

\[ V = \left( \frac{9}{\rho_1} \| \Psi_k \|^2 (3L_2^2 + \rho_2^2) + \frac{3}{\rho_2} L_3^2 \right) \| x_k^{j+1} - x_k^j \|^2 + \frac{3L_2^2}{\rho_1} \| v_k^{j+1} - v_k^j \|^2 \]

Adding \( V \) to both sides of (A.16) and rearranging gives the desired inequality

\[ \mathcal{L}_{\rho_1, \rho_2}(w_k^{j+1}) + \frac{3}{\rho_1} L_2^2 \| v_k^{j+1} - v_k^j \|^2 + \left( \frac{9}{\rho_1} \| \Psi_k \|^2 (3L_2^2 + \rho_2^2) + \frac{3}{\rho_2} L_3^2 \right) \| x_k^{j+1} - x_k^j \|^2 \]

\[ \leq \mathcal{L}_{\rho_1, \rho_2}(w_k^j) + \frac{9}{\rho_1} \| \Psi_k \|^2 (3L_2^2 + \rho_2^2) \| x_k^j - x_k^{j-1} \|^2 \]

\[ - \frac{\alpha}{2} \left( \frac{9}{\rho_1} \| \Psi_k \|^2 (3L_2^2 + L_3^2 + \rho_2^2) + \frac{3}{\rho_2} (L_2^2 + L_3^2) \right) \| \hat{w}_k^{j+1} - \hat{w}_k^j \|^2 \]

Therefore, by letting \( v_k^j = \hat{v}_k^j \) and \( x_k^j = \hat{x}_k^j \) in \( \Omega(\hat{w}_k^{j+1}) \), \( v_k^{j-1} = \hat{v}_k^j \) and \( x_k^{j-1} = \hat{x}_k^j \) in \( \Omega(\hat{w}_k^j) \), it follows that

\[ \Omega(\hat{w}_k^{j+1}) \leq \Omega(\hat{w}_k^j) - a \| \hat{w}_k^{j+1} - \hat{w}_k^j \|^2 \]  

(A.17)

where

\[ a = \frac{\alpha}{2} - 2 \left( \frac{9}{\rho_1} \| \Psi_k \|^2 (3L_2^2 + L_3^2 + \rho_2^2) + \frac{3}{\rho_2} (L_2^2 + L_3^2) \right) \]

is a positive real number according to (26). As we already noticed, \( \Omega \) satisfies the sufficient decrease condition (C1), which implies that the sequence \( \{ \Omega(\hat{w}_k^j) \}_{j \in \mathbb{N}} \) is nonincreasing.

**B. Proof of Lemma 2**

Since \( (A.2)-(A.5) \) and

\[ \Omega(\hat{w}_k^{j+1}) = \Omega(u_k^{j+1}, v_k^{j+1}, x_k^{j+1}, y_k^{j+1}, z_k^{j+1}, w_k^{j+1}) \]

\[ = \mathcal{L}_{\rho_1, \rho_2}(u_k^{j+1}, v_k^{j+1}, x_k^{j+1}, y_k^{j+1}, z_k^{j+1}) + \frac{\tau_1}{2} \| v_k^{j+1} - v_k^j \|^2 + \frac{\tau_2}{2} \| x_k^{j+1} - x_k^j \|^2, \]

the partial derivative \( \partial \Omega_u(\hat{w}_k^{j+1}) \) with respect to \( u_k \) yields the following expression

\[ \partial \Omega_u(\hat{w}_k^{j+1}) = \partial G(u_k^{j+1}) + \left[ \partial B(u_k^{j+1}) \right]^T y_k^{j+1} + \rho_1 (\partial B(u_k^{j+1}))^T (b(u_k^{j+1}) - v_k^{j+1}). \]  

(A.18)

Substituting (A.4) for \( \partial G(u_k^{j+1}) \) to (A.18) gives

\[ \partial \Omega_u(\hat{w}_k^{j+1}) \triangleq \left[ \partial B(u_k^{j+1}) \right]^T (y_k^{j+1} + \rho_1 (v_k^{j+1} - v_k^j) - \nabla \phi_1(u_k^{j+1}) + \nabla \phi_1(u_k^j) + \left[ \partial B(u_k^{j+1}) \right]^T y_k^{j+1} + \rho_1 \left[ \partial B(u_k^{j+1}) \right]^T (b(u_k^{j+1}) - v_k^{j+1}) \]

\[ + \left[ \partial B(u_k^{j+1}) \right]^T y_k^j - \left[ \partial B(u_k^{j+1}) \right]^T y_k^j \]
From the fact that \( y_{k}^{j+1} = y_{k}^{j} + \rho_{1}(b(u_{k}^{j+1}) - v_{k}^{j+1}) \), the partial derivative finally becomes
\[
\frac{\partial \Omega_{u}(\hat{w}_{k}^{j+1})}{\partial u} = \left[ \frac{\partial B(u_{k}^{j+1})}{\partial u} \right]^{T}(y_{k}^{j+1} - y_{k}^{j}) - \rho_{1} \left[ \frac{\partial B(u_{k}^{j+1})}{\partial u} \right] (v_{k}^{j+1} - v_{k}^{j}) - \nabla \phi_{1}(u_{k}^{j+1}) + \nabla \phi_{1}(u_{k}^{j})
\]
In a similar way, by (16d) and (16e), the partial derivatives \( \partial \Omega_{v}(\hat{w}_{k}^{j+1}) \) with respect to \( v_{k} \) and \( \partial \Omega_{w}(\hat{w}_{k}^{j+1}) \) with respect to \( x_{k} \) are given by
\[
\frac{\partial \Omega_{v}(\hat{w}_{k}^{j+1})}{\partial v} = -\rho_{2}\Psi_{k}^{T}(x_{k}^{j+1} - x_{k}^{j}) - \nabla \phi_{2}(v_{k}^{j+1}) + \nabla \phi_{2}(v_{k}^{j}) - (y_{k}^{j+1} - y_{k}^{j}) + \Psi_{k}(x_{k}^{j+1} - x_{k}^{j}) + \tau_{1}(v_{k}^{j+1} - v_{k}^{j})
\]
\[
\frac{\partial \Omega_{w}(\hat{w}_{k}^{j+1})}{\partial w} = -\nabla \phi_{3}(x_{k}^{j+1}) + \nabla \phi_{3}(x_{k}^{j}) + \tau_{2}(x_{k}^{j+1} - x_{k}^{j}) - (z_{k}^{j+1} - z_{k}^{j})
\]
Based on (16d) and (16e), we get the partial derivatives \( \partial \Omega_{y}(\hat{w}_{k}^{j+1}) \) and \( \partial \Omega_{z}(\hat{w}_{k}^{j+1}) \)
\[
\frac{\partial \Omega_{y}(\hat{w}_{k}^{j+1})}{\partial y} = b(u_{k}^{j+1}) - v_{k}^{j+1} \quad \text{(16a)} \quad \frac{1}{\rho_{1}}(y_{k}^{j+1} - y_{k}^{j})
\]
\[
\frac{\partial \Omega_{z}(\hat{w}_{k}^{j+1})}{\partial z} = \Phi_{k}x_{0} + \Psi_{k}v_{k}^{j+1} - x_{k}^{j+1} + 1 \quad \text{(16c)} \quad \frac{1}{\rho_{2}}(z_{k}^{j+1} - z_{k}^{j})
\]
Apparently, the partial derivatives \( \partial \Omega_{u}(\hat{w}_{k}^{j+1}) \) and \( \partial \Omega_{x}(\hat{w}_{k}^{j+1}) \) are as follows
\[
\frac{\partial \Omega_{u}(\hat{w}_{k}^{j+1})}{\partial u} = \tau_{1}(v_{k}^{j+1} - v_{k}^{j})
\]
\[
\frac{\partial \Omega_{x}(\hat{w}_{k}^{j+1})}{\partial x} = \tau_{2}(x_{k}^{j+1} - x_{k}^{j+1})
\]
Due to the continuity of \( b(\cdot) \), \( \partial B(u_{k}^{j+1}) \) is bounded. Additionally, as \( \{w_{k}^{j}\} \) is bounded by (30), the above together with (A3) implies that there exists \( \ell > 0 \) and \( c \in \partial \Omega(\hat{w}_{k}^{j+1}) \) such that \( ||c|| \leq \ell||\hat{w}_{k}^{j+1} - \hat{w}_{k}^{j+1}|| \). Thus the inequality as stated in Lemma [4] holds.

C. Proof of Lemma [5]

To verify Condition (C3), we assume that there exists a subsequence \( \{\hat{w}_{k}^{j}\} \) that converges to \( \hat{w}_{k}^{\ast} = (u_{k}^{\ast}, v_{k}^{\ast}, x_{k}^{\ast}, y_{k}^{\ast}, z_{k}^{\ast}, \hat{v}_{k}^{\ast}, \hat{x}_{k}^{\ast}) \). By the lower semicontinuity of \( \Omega \), \( \lim_{d \to \infty} \Omega(\hat{w}_{k}^{j}) \geq \Omega(\hat{w}_{k}^{\ast}) \). In addition, by the optimality we have
\[
G(u_{k}^{j}) + \frac{1}{2}\left[ y_{k}^{j+1} - \frac{1}{2}\|b(u_{k}^{j+1}) - v_{k}^{j}\|^{2} + \Delta \phi_{1}(u_{k}^{j+1}, u_{k}^{j}) \right]
\]
\[
\leq G(u_{k}^{j}) + \frac{1}{2}\left[ y_{k}^{j+1} - \frac{1}{2}\|b(u_{k}^{j}) - v_{k}^{j}\|^{2} + \Delta \phi_{1}(u_{k}^{j}, u_{k}^{j}) \right]
\]
According to the asymptotic regularity of \( \{u_{k}^{j}\} \) by Lemma [3] this implies \( \lim_{d \to \infty} G(u_{k}^{j+1}) \leq G(u_{k}^{\ast}) \). Likewise, we obtain \( \lim_{d \to \infty} F(x_{k}^{j+1}) \leq F(x_{k}^{\ast}) \), and then \( \lim_{d \to \infty} \Omega(\hat{w}_{k}^{j+1}) \leq \Omega(\hat{w}_{k}^{\ast}) \). Hence, it can be concluded that
\[
\lim_{d \to \infty} \Omega(\hat{w}_{k}^{j+1}) = \Omega(\hat{w}_{k}^{\ast})
\]
and Condition (C3) holds.

REFERENCES

[1] Frank Allgower, Rolf Findeisen, and Zoltan K Nagy. Nonlinear model predictive control: From theory to application. Journal-Chinese Institute Of Chemical Engineers, 35(3):299–316, 2004.
[2] S Joe Qin and Thomas A Badgwell. An overview of nonlinear model predictive control applications. In Frank Allgöwer and Alex Zheng, editors, Nonlinear model predictive control, volume 26, pages 369–392. Birkhäuser, 2000.
[3] Haoyang Deng and Toshiyuki Ohtsuka. A parallel newton-type method for nonlinear model predictive control. Automatica, 109:108560, 2019.
[4] Rihab Kouki, Ahmed Chemori, and Faouzi Bouani. A new fast nmpc scheme for parallel kinematic manipulators: Design and real-time experiments. In the proceedings of 2019 International Conference on Signal, Control and Communication (SCC), pages 69–75, 2019.
[5] Sebastian East and Mark Cannon. Admm for mpc with state and input constraints, and input nonlinearity. In the proceedings of 2018 Annual American Control Conference (ACC), pages 4514–4519, 2018.
[6] Stephen Boyd, Neal Parikh, and Eric Chu. Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, 2011.
[7] Wentao Tang and Prodromos Daoutidis. Distributed nonlinear model predictive control through accelerated parallel admm. In the proceedings of 2019 American Control Conference (ACC), pages 1406–1411, 2019.

[8] Attila Kozma, Carlo Savorgnan, and Moritz Diehl. Distributed multiple shooting for large scale nonlinear systems. In José M. Maestre and Rudy R. Negenborn, editors, Distributed Model Predictive Control Made Easy, pages 327–340. Springer Netherlands, 2014.

[9] Guoyin Li and Ting Kei Pong. Global convergence of splitting methods for nonconvex composite optimization. SIAM Journal on Optimization, 25(4):2434–2460, 2015.

[10] Junxiang Wang and Liang Zhao. Nonconvex generalization of alternating direction method of multipliers for nonlinear equality constrained problems. Results in Control and Optimization, 2:100009, 2021.

[11] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. Mathematical Programming, 155(1-2):57–79, 2016.

[12] Bingsheng He, Liusheng Hou, and Xiaoming Yuan. On full jacobian decomposition of the augmented lagrangian method for separable convex programming. SIAM Journal on Optimization, 24(4):2274–2312, 2015.

[13] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. Journal of Scientific Computing, 78(1):29–63, 2019.

[14] Andreea B Alexandru, Konstantinos Gatsis, Yasser Shoukry, Sanjit A Seshia, Paulo Tabuada, and George J Pappas. Cloud-based quadratic optimization with partially homomorphic encryption. IEEE Transactions on Automatic Control, 66(5):2357–2364, 2020.

[15] Yongnan Zhang, Yonghua Zhou, Huapu Lu, and Hamidou Fujita. Spark cloud-based parallel computing for traffic network flow predictive control using non-analytical predictive model. IEEE Transactions on Intelligent Transportation Systems, 2021. doi[10.1109/TITS.2021.3071862].

[16] Yu Kaneko and Toshio Ito. A reliable cloud-based feedback control system. In the proceedings of 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), pages 880–883, 2016.

[17] Takashi Tanaka, Mikael Skoglund, Henrik Sandberg, and Karl Henrik Johansson. Directed information and privacy loss in cloud-based control. In the proceedings of 2017 American Control Conference (ACC), pages 1666–1672, 2017.

[18] Alimzhan Sultagazin and Paulo Tabuada. Symmetries and isomorphisms for privacy in control over the cloud. IEEE Transactions on Automatic Control, 66(2):538–549, 2020.

[19] Per Skarinn, Johan Eker, and Karl-Erik Årzén. Cloud-based model predictive control with variable horizon. IFAC-PapersOnLine, 53(2):6993–7000, 2020.

[20] Hasan Esen, Masakazu Adachi, Daniele Bernardini, Alberto Bemporad, Dominik Rost, and Jens Knodel. Control as a service (caas) cloud-based software architecture for automotive control applications. In the proceedings of the Second International Workshop on the Swarm at the Edge of the Cloud, pages 13–18, 2015.

[21] Emrah Bıyık, James D Brooks, Hülas Sehgal, Jigar Shah, and Sahika Gency. Cloud-based model predictive building thermostatic controls of commercial buildings: Algorithm and implementation. In the proceedings of 2015 American Control Conference (ACC), pages 1683–1688, 2015.

[22] Axel Vick, Jan Guhl, and Jorg Kruger. Model predictive control as a service concept and architecture for use in cloud-based robot control. In the proceedings of 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), pages 607–612, 2016.

[23] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjosteen, Angela Jäschke, Christian A Reuter, and Martin Strand. A guide to fully homomorphic encryption. IACR Cryptol. ePrint Arch., 2015:1192, 2015.

[24] Jorge Cortés, Geir E Dullerud, Shuo Han, Jerome Le Ny, Sayan Mitra, and George J Pappas. Differential privacy in control and network systems. In the proceedings of 2016 IEEE 55th Conference on Decision and Control (CDC), pages 4252–4272, 2016.

[25] PC Weeraddana and Carlo Fischione. On the privacy of optimization. IFAC-PapersOnLine, 50(1):9502–9508, 2017.

[26] Fenghui Wang, Wenfei Cao, and Zongben Xu. Convergence of multi-block bregman admm for nonconvex composite problems. Science China Information Sciences, 61(12):1–12, 2018.

[27] Alexander Ya. Kruger. On fréchet subdifferentials. Journal of Mathematical Sciences, 116(3):3325–3358, 2003.

[28] Xi Yin Zheng and Kung Fu Ng. The fermat rule for multifunctions on banach spaces. Mathematical programming, 104(1):69–90, 2005.

[29] Pierre-Antoine Absil, Robert Mahony, and Benjamin Andrews. Convergence of the iterates of descent methods for analytic cost functions. SIAM Journal on Optimization, 16(2):531–547, 2005.

[30] Hedy Attouch and Jérôme Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. Mathematical Programming, 116(1):5–16, 2009.

[31] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soueyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-lojasiewicz inequality. Mathematics of operations research, 35(2):438–457, 2010.

[32] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. Mathematical Programming, 137(1):91–129, 2013.

[33] Levin Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR computational mathematics and mathematical physics, 7(3):200–217, 1967.

[34] Navid Rahbari Asr and Mo-Yuen Chow. Network coordinated distributed demand management for optimal large-scale charging of phvcs/phvcs. In the proceedings of 2013 IEEE Power & Energy Society General Meeting, pages 1–5, 2013.

[35] Wencong Su and Mo-Yuen Chow. Performance evaluation of a phv parking station using particle swarm optimization. In the proceedings of 2011 IEEE power and energy society general meeting, pages 1–6, 2011.
**Yaling Ma** received her B.S. degree in Automation from University of Science and Technology Beijing, China, in 2020. She is currently working forward the M.S. degree in Control Science and Engineering from Beijing Institute of Technology, Beijing, China. Her research interests include model predictive control, distribution optimization and cloud computing.

**Runze Gao** received his B.S. degree in Automation from Beijing Institute of Technology, China in 2017. He is currently working forward the Ph.D. degree in Control Science and Engineering from Beijing Institute of Technology. His research interests cover cloud control system, model predictive control and data-driven predictive control.

**Li Dai** received the B.S. degree in Information and Computing Science in 2010 and the Ph.D. degree in Control Science and Engineering in 2016, both from Beijing Institute of Technology, Beijing, China. Now she is an assistant professor in the School of Automation, Beijing Institute of Technology. Her research interests include model predictive control, distributed control, data-driven control, stochastic systems, and networked control systems.

**Jinxian Wu** received his B.S. degree in Automation from Qingdao University, Shandong, China, in 2019 and his M.S. degree in Control Science and Engineering from Beijing Institute of Technology, Beijing, China, in 2022. Now he is currently working toward the Ph.D. degree in Control Science and Engineering from Beijing Institute of Technology. His research interests include distributed model predictive control and distributed optimization.

**Yuanqing Xia** was born in Anhui, China, in 1971. He received the M.S. degree in fundamental mathematics from Anhui University, Hefei, China, in 1998, and the Ph.D. degree in control theory and control engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001. From 2002 to 2003, he was a Post-Doctoral Research Associate with the Institute of Systems Science, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing. From 2003 to 2004, he was with the National University of Singapore, Singapore, as a Research Fellow, where he researched on variable structure control. From 2004 to 2006, he was with the University of Glamorgan, Pontypridd, U.K., as a Research Fellow. From 2007 to 2008, he was a Guest Professor with Innsbruck Medical University, Innsbruck, Austria. Since 2004, he has been with the School of Automation, Beijing Institute of Technology, Beijing, first as an Associate Professor, then, since 2008, as a Professor. His current research interests include networked control systems, robust control and signal processing, and active disturbance rejection control.