ALGORITHMS FOR DISTRIBUTED MODEL PREDICTIVE CONTROL

Lucas Fraile
UID: 505034889
`lucasfrailev@gmail.com`

Alimzhan Sultangazin
UID: 005035952
`asultangazin@ucla.edu`

January 5, 2022

ABSTRACT

In this project we will consider three state-of-the-art algorithms for solving distributed model predictive control (DMPC) problems. The first algorithm utilizes dual decomposition and accelerated gradient methods in a distributed fashion. The second algorithm uses alternating direction method of multipliers (ADMM) on the primal problem. We implement the three aforementioned algorithms and compare their results with those given by CVX, which we have taken to be our benchmark. In this report, we present these results along with the discussion of the challenges we have encountered when implementing these algorithms.

1 Introduction

In the recent years, distributed control of large-scale systems has become an increasingly popular topic, because often centralized solutions can be inefficient (due to the large amount of data or the network topology) or inapplicable (the system does not have a central node). Distributed model predictive control (DMPC) is the version of a well-known MPC scheme modified to control the network of systems by using the local controllers at each system.

Model predictive control (MPC) is an optimization-based method, which calculates a sequence of inputs that drives the system to the desired state, while taking into account the dynamics and the set of imposed constraints. Typically, this is expressed as follows:

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & \frac{1}{2}\sum_{t=0}^{N-1}(x_t^T Q x_t + u_t^T R u_t) \\ \text{subject to} \quad & (x_t, u_t) \in \mathcal{X} \times \mathcal{U}, \qquad t = 0, \dots, N-1, \\ & x_{t+1} = Ax_t + Bu_t, \quad t = 0, \dots, N-2, \end{aligned} \tag{1}$$

where Q and R are positive semi-definite costs; (A, B) describe the system dynamics; $\mathcal{X}$ (resp. $\mathcal{U}$) is the set of states (resp. inputs) satisfying the imposed constraints; and x and u are the sequences of states x_t and u_t, respectively [1].

In DMPC, the entire system is divided into subsystems and each subsystem is represented by a node in the network. Moreover, each subsystem possesses a corresponding local controller. The cost in (1) is optimized by these local controllers. A local controller at a certain node can physically influence subsystems at other nodes through coupled dynamics and constraints. Therefore, for correct performance, communication between local controllers is required [2]. Each local controller must decide which input to feed into the system, based on the information it receives from its neighbors (i.e., the subsystems with which its local subsystem is coupled with via either dynamics or constraints).

The application of different distributed optimization algorithms was suggested in the DMPC literature. In [3], an accelerated gradient method is used on distributed MPC problem that is reformulated using dual decomposition. The authors of this paper argue that this brings about improvement in the convergence rate, in comparison with previous works that used regular gradient-based methods. In [2], the DMPC problem is reformulated as a global consensus problem and, then, the alternating direction method of multipliers (ADMM) is used to solve the problem in a distributed

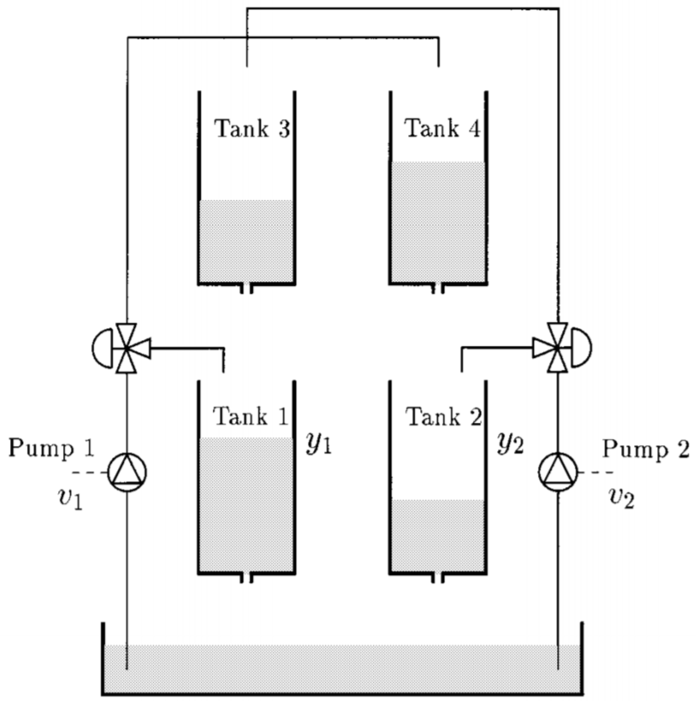

Figure 1: Schematic of the quadruple-tank process. The water levels in tanks 1 and 4 are controlled with pump 1, while the water levels in tanks 2 and 3 are controlled with pump 3. Water also falls down from the upper tanks into the lower tanks.

fashion. ADMM is applied to both the primal and the dual problem. It is said that this algorithm results in a significant reduction of the data exchange. For this project, we have chosen to implement the algorithm described in [3] and the primal ADMM described in [2]. These algorithms were tested on the quadruple-tank process system from [4]. The main metrics we have seen in the literature are the improvement of the convergence rate and the reduction of the communication load [1, 2]. Here, due to the time limitations, we mainly focus on the convergence exhibited by the algorithms (i.e., how close their solution gets to the chosen benchmark).

2 Benchmark

In this section, we describe the control benchmark that will be used to test the implemented algorithms. This benchmark system was originally proposed by Johansson in [4] and has been since widely used for testing of distributed control algorithms. As shown in Figure 1, the four tanks are filled from a storage reservoir below by means of two pumps. We denote the amount of water supplied by these pumps to the valves 1 and 2 by q_1 and q_2, respectively. Then, at each of the three-way valves, the water flow is divided based on parameters $\gamma_1, \gamma_2 \in (0, 1)$ (e.g., the flow into Tank 1 is given by $\gamma_1 q_1$ and the flow into Tank 4 is given by $(1 - \gamma_2)q_1$). We also denote the water level in Tank i by h_i for all $i \in \{1, 2, 3, 4\}$. This system has non-linear dynamics, but these dynamics can be linearized at an operating point [5]. We take this operating point to be :

$$h_1^0 = 0.65 \qquad h_2^0 = 0.66 \qquad h_3^0 = 0.65 \qquad h_4^0 = 0.66$$
$$q_1^0 = 1.63 \qquad q_2^0 = 2.00$$

We define the deviation variables:

$$x_i = h_i - h_i^0, \tag{2}$$
$$v_i = q_i - q_i^0, \quad \forall i \in \{1, 2, 3, 4\}, \tag{3}$$

and thus obtain the linearized state-space equation for the continuous dynamics with pump flows as inputs:

$$\begin{bmatrix} \dot{x}_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_1} & 0 & \frac{1}{\tau_1} & 0 \\ 0 & -\frac{1}{\tau_2} & 0 & \frac{1}{\tau_4} \\ 0 & 0 & -\frac{1}{\tau_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} \frac{\gamma_1}{S} & 0 \\ 0 & \frac{\gamma_2}{S} \\ 0 & \frac{(1-\gamma_2)}{S} \\ \frac{(1-\gamma_1)}{S} & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \tag{4}$$

where $S = 0.06$ is the cross section of the tanks and τ_i for $i \in \{1, 2, 3, 4\}$ is the time constant for Tank i. In this model, we keep the valve parameters constant at $\gamma_1 = 0.3$ and $\gamma_2 = 0.4$. Based on this model, we obtain a discrete-time model by using Tustin's method, as recommended in [5]. We further partition the resulting system into two subsystems: one consisting of tanks 1 and 3, the other - of tanks 2 and 4. We pair input from pump 2 v_2 with subsystem 1 and input from pump 1 v_1 with subsystem 2 and denote them by u_1 and u_2, respectively. In other words, the local MPC controller of subsystem 1 computes u_1 and the local MPC controller of subsystem 2 computes u_2. The resulting dynamics for two subsystems are:

$$x_{sub1}(t+1) \triangleq \begin{bmatrix} x_1(t+1) \\ x_3(t+1) \end{bmatrix} = A_{11} \begin{bmatrix} x_1(t) \\ x_3(t) \end{bmatrix} + B_{11}u_1(t) + B_{12}u_2(t) \tag{5}$$

$$x_{sub2}(t+1) \triangleq \begin{bmatrix} x_2(t+1) \\ x_4(t+1) \end{bmatrix} = A_{22} \begin{bmatrix} x_2(t) \\ x_4(t) \end{bmatrix} + B_{21}u_1(t) + B_{22}u_2(t) \tag{6}$$

Note how the dynamics of two subsystems are coupled via two inputs. This is the reason why distributed MPC is needed.

The control problem solved by MPC will be the following. We would like states x_1 and x_3 to reach a value of $r_1 = r_3 = 0.35$ (recall that the states define a deviation from the linearization level). To define the references for the inputs, u_{r1} and u_{r2}, we calculate the steady-state input that would keep the states of subsystem 1 at $x_1 = x_3 = 0.35$. To define the references for the state of subsystem 2, r_2 and r_3, we calculate the steady state of subsystem 2 for inputs u_{r1} and u_{r2}. Upon getting these values, we can define the global cost of our MPC problem to be:

$$J(x, u) = \sum_{t=0}^{N-1} ||x_{sub1}(t) - r_{sub1}||_{Q_1} + ||x_{sub2}(t) - r_{sub2}||_{Q_2} + ||u_1(t) - u_{r1}||_{R_1} + ||u_2(t) - u_{r2}||_{R_2}, \tag{7}$$

where $x(t) = [x_1(t) \quad x_2(t) \quad x_3(t) \quad x_4(t)]^T$, $r = [r_1 \quad r_2 \quad r_3 \quad r_4]^T$, and $Q = I$, $R = 0.01$ are cost functions for the states and the inputs, respectively

In addition, we have lower and upper bounds on the states (denoted by $x_{i,max}$ and $x_{i,min}$ for all $i \in \{1, 2, 3, 4\}$) and the inputs (denoted by $u_{i,max}$ and $u_{i,min}$ for all $i \in \{1, 2, 3, 4\}$), given by the volume of the tanks and power of the pumps. For all $t \geq 0$, the following is true:

$$x_{i,min} \leq x_i(t) \leq x_{i,max} \tag{8}$$

$$u_{i,min} \leq u_i(t) \leq u_{i,max} \tag{9}$$

3 Algorithm Overview

In this section, we describe in detail the algorithms that we have implemented as a part of this project to solve the problem of distributed MPC. These algorithms will be applied to the benchmark described in Section 2. Recall, that we have decided to consider the algorithm from [3], based on accelerated gradient method, and the primal ADMM algorithm from [2].

Before we can use either of those algorithms, we are required to bring them to a form, where all the variables are stacked into a single vector. Therefore, the optimization problem in (1) is reformulated into:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \frac{1}{2} y^T H y + g^T y \\ \text{subject to} \quad & A_{eq} y = B_{eq}, \\ & A_{ineq} y \leq B_{ineq}, \end{aligned} \tag{10}$$

where $y(t) = \begin{bmatrix} y_1^T & y_2^T & ... & y_M^T \end{bmatrix}^T$ and $H = \text{diag}(H_1, ..., H_M)$. Each of the y_i and H_i corresponds to a certain subsystem and has the following structure:

$$y_i = \begin{bmatrix} u_i(0) \\ x_i(0) \\ u_i(1) \\ x_i(1) \\ \vdots \\ u_i(N-1) \\ x_i(N-1) \end{bmatrix} \in \mathbb{R}^{N(n_i+m_i)}, \quad H_i = \begin{bmatrix} R_i & 0 & 0 & \dots 0 & \\ 0 & Q_i & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & R_i & 0 \\ 0 & \dots & \dots & 0 & Q_i \end{bmatrix} \in \mathbb{R}^{N(n_i+m_i)\times N(n_i+m_i)}, \tag{11}$$

where $Q_i \in \mathbb{R}^{n_i \times n_i}$ and $R_i \in \mathbb{R}^{m_i \times m_i}$ are positive definite cost matrices for the state and input of subsystem i, respectively. The linear term g has the same structure as y and is defined by the reference points r and u_r.

The equality constraints are derived from the dynamics of the system. The inequality constraints are given by the lower and upper bounds on the states and inputs. Due to the space limitations in this report, we will not be discussing the derivation of these matrices, but will only give their general structure:

$$A_{eq} = [A_{eq,ij}] \in \mathbb{R}^{((n_i+m_i)+(N-1)n_i)\times(N(n_i+m_i))}, \tag{12}$$
$$A_{ineq} = [A_{ineq,ij}] \in \mathbb{R}^{(2N(n_i+m_i))\times(N(n_i+m_i))} \tag{13}$$

.

3.1 Accelerated gradient method and dual decomposition for DMPC

Dual decomposition methods have been widely applied to distributed MPC problems (e.g., see [6]). However, they were previously largely based on regular gradient descent methods that converge at a rate $O(\frac{1}{k})$. The novelty of work in [3] lies in using the accelerated gradient method. This method has a tight lower bound of $O(\frac{1}{k^2})$ on the convergence rate [7].

In [3], the authors proceed by finding a dual problem of (10) and apply accelerated gradient method to it. Before we start describing the algorithm, let us define:

$$\mathcal{A} = \begin{bmatrix} A_{eq} \\ A_{ineq} \end{bmatrix}, \quad \mathcal{B} = \begin{bmatrix} B_{eq} \\ B_{ineq} \end{bmatrix}. \tag{14}$$

The set of constraints that unit i is responsible for is given by $\mathcal{L}_i$. Now, we can define two sets of neighbors to computational unit i:

$$\mathcal{N}_i = \{j \in \{1, ..., M\} | \exists l \in \mathcal{L}_i \text{ s.t. } a_{lj} \neq 0\} \tag{15}$$
$$\mathcal{M}_i = \{j \in \{1, ..., M\} | \exists l \in \mathcal{L}_j \text{ s.t. } a_{li} \neq 0\} \tag{16}$$

We define a dual variable $z = \begin{bmatrix} \lambda^T & \mu^T \end{bmatrix}^T$, where λ corresponds to the equality constraints and μ corresponds to the inequality constraints. For simplicity of notation, let us denote the dimensions of the constraints as follows $A_{eq} \in \mathbb{R}^{q\times n}$ and $A_{ineq} \in \mathbb{R}^{(s-q)\times n}$. The feasibility constraints on the dual variable are as follows:

$$Z = \left\{ z = (\lambda^T, \mu^T)^T \middle| \begin{array}{ll} z_l \in \mathbb{R}, & l = \{1, ..., q\} \\ z_l \geq 0, & l = \{q+1, ..., s\} \end{array} \right\} \tag{17}$$

The distributed algorithm based on accelerated gradient descent solves the following dual problem:

$$\text{sup}_{z\in Z} \left\{ -\frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1} (\mathcal{A}^T z + g) - \mathcal{B}^T z \right\} \tag{18}$$

that was derived using dual decomposition.

The algorithm is given as follows:

Algorithm 1: Distributed accelerated gradient algorithm

```
Initialize $\lambda^0 = \lambda^{-1}$, $\mu^0 = \mu^{-1}$, $y^0 = y^{-1}$ ;
For every node $i$:
for $k \geq 0$ do
    1) Primal update:
    $x_i^k = -H_i^{-1}(-\mathcal{A}_i^T z^k - g_i)$;
    $\bar{x}_i^k = x_i^k + \frac{k-1}{k+2}(x_i^k - x_i^{k-1})$;
    2) Send $\bar{x}_i^k$ to each $j \in \mathcal{M}_i$, receive $\bar{x}_j^k$ from each $j \in \mathcal{N}_i$ ;
    3) Dual update:
    $\lambda_l^{k+1} = \lambda_l^k + \frac{k-1}{k+2}(\lambda_l^k - \lambda_l^{k-1}) + \frac{1}{L}(a_l^T \bar{x}^k - b_l)$,
    where $l \in \mathcal{L}_i$ and $0 \leq l \leq q$
    $\mu_l^{k+1} = \max\{0, \mu_l^k + \frac{k-1}{k+2}(\mu_l^k - \mu_l^{k-1}) + \frac{1}{L}(a_l^T \bar{x}^k - b_l)\}$,
    where $l \in \mathcal{L}_i$ and $q+1 \leq l \leq s$
    4) Send $\{\lambda_l^{k+1}\}_{l\in\mathcal{L}_i}$, $\{\mu_l^{k+1}\}_{l\in\mathcal{L}_i}$ to each $j \in \mathcal{N}_i$, receive $\{\lambda_l^{k+1}\}_{\in\mathcal{L}_j}$, $\{\mu_l^{k+1}\}_{l\in\mathcal{L}_j}$ from each $j \in \mathcal{M}_i$
end
```

3.2 Primal ADMM for DMPC

Before we discuss how primal ADMM was implemented for DMPC, let us define the notion of neighborhood that they used in [2]. If the dynamics are given by block matrices $A = [A_{ij}]$, $B = [B_{ij}]$ with $i, j \in \{1, 2, ..., M\}$, then the set of neighbors $\mathcal{N}_i$ and $\mathcal{M}_i$ is defined as:

$$\mathcal{N}_i = \{j \in \mathcal{S} | A_{ij} \neq 0 \text{ or } B_{ij} \neq 0\} \tag{19}$$
$$\mathcal{M}_i = \{j \in \mathcal{S} | i \in \mathcal{N}_j\} \tag{20}$$

Then, the optimization problem in (10) is reformulated into the consensus form, where each subsystem has a local variable $y^i = \begin{bmatrix}(y_1^i)^T & (y_2^i)^T & \dots & (y_N^i)^T\end{bmatrix}^T$, where y_j^i is the local copy of y_j in subsystem i.

We introduce a consensus constraint to coordinate the results of optimizations at each local controller:

$$y^i - \bar{y}^i = 0, \tag{21}$$

where:

$$\bar{y}^i = \begin{bmatrix}(\bar{y}_1^i)^T & (\bar{y}_2^i)^T & \dots & (\bar{y}_N^i)^T\end{bmatrix}^T \tag{22}$$
$$\bar{y}_i = \frac{1}{|\mathcal{M}_i|} \sum_{j\in\mathcal{M}_i} y_i^j \tag{23}$$

The local copies of the variable y_i are to be averaged and transmitted to the network to find $\bar{y}^i$. This algorithm can be seen as consensus between the local optimizations and the global average.

The local optimization problem solved at each node is given by minimization of the augmented Lagrangian:

$$\begin{aligned} \underset{y^i}{\text{minimize}} \quad & \frac{1}{2}(y_i^i)^T H y_i^i + (\gamma^i)^T(y^i - \bar{y}^i) + \frac{\rho}{2}||y^i - \bar{y}^i||_2^2 \\ \text{subject to} \quad & \sum_{j\in\mathcal{N}_i} A_{eq,ij} y_j^i = B_{eq,i}, \\ & A_{ineq,ii} y_i^i \leq B_{ineq,i}, \end{aligned} \tag{24}$$

where γ^i is the Lagrange multiplier for the consensus constraint and ρ is the penalty coefficient.

Let us denote the optimal solution of (24) by $(y^i)^+$. The algorithm is given as follows:

Algorithm 2: Primal ADMM

```
Initialize $\gamma^i = 0$ and $\bar{y}^i = 0$ ;
For every node $i$:
for $k \geq 0$ do
    1) Calculate $(y^i)^+$ by solving (24);
    2) Send $(y^i_j)^+$ to all $j \in \mathcal{N}_i$;
    3) Calculate $\bar{y}_i^+$ by using (23);
    4) Send $(\bar{y}_i)^+$ to all $j \in \mathcal{M}_i$ ;
    5) Stack up $(\bar{y}^i)^+$ according to (22) ;
    6) Update $(\gamma)^+ = \gamma^i + \rho((y^i)^+ - (\bar{y}^i)^+)$
end
```

3.3 Dual ADMM for DMPC

We can formulate the dual problem to (10):

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \frac{1}{2} z^T \mathcal{A} H^{-1} \mathcal{A}^T y + \mathcal{B}^T z \\ \text{subject to} \quad & \mu_i \geq 0, \quad \forall i \in \{1, ..., M\}, \end{aligned} \tag{25}$$

where $\mathcal{A} = \begin{bmatrix} \mathcal{A}_1^T & ... & \mathcal{A}_M^T \end{bmatrix}^T$, $\mathcal{B} = \begin{bmatrix} \mathcal{B}_1^T & ... & \mathcal{B}_M^T \end{bmatrix}^T$ and $z = \begin{bmatrix} z_1^T & ... & z_M^T \end{bmatrix}^T$ with:

$$\mathcal{A}_i = \begin{bmatrix} A_{eq,i}^T & A_{ineq,i}^T \end{bmatrix}^T \tag{26}$$

$$\mathcal{B}_i = \begin{bmatrix} B_{eq,i}^T & B_{ineq,i}^T \end{bmatrix}^T \tag{27}$$

$$z_i = \begin{bmatrix} \lambda_i^T & \mu_i^T . \end{bmatrix} \tag{28}$$

These partitions correspond to equality and inequality constraints.

We define a matrix $\hat{H} = \mathcal{A} H^{-1} \mathcal{A}^T$. It is clear that this matrix is not block-diagonal. However, it has an inherent structure due to the sparsity patterns of $\mathcal{A}$ and H^{-1}. It was determined in [2] that block $\hat{\mathcal{H}}_{ij}$ is zero if $\mathcal{N}_i \cap \mathcal{N}_j = \emptyset$. It can also be seen that $\hat{\mathcal{H}}$ is positive semidefinite and symmetric.

In the dual problem, the set of neighbors for node i is given by $\hat{\mathcal{N}_i} = \{j \in \{1, ..., M\} | \mathcal{N}_i \cap \mathcal{N}_j \neq \emptyset\}$. To solve (25), we must divide it into M local subproblems. Based on the definition of neighbors in the dual problem, we can see that the optimization variable z^i at each subsystem i is given by:

$$z^i = [z^i_j]_{j \in \hat{\mathcal{N}_i}}, \tag{29}$$

where $z^i \in \mathbb{R}^{\hat{q}^i}$, $\hat{q}^i = \sum_{j \in \hat{\mathcal{N}_i}} s_j$ and z^i_j being the local copy of z_j in i.

In terms of implementation, using $\hat{H}$ as it was defined earlier posed a problem because, due to numerical errors, it was no longer positive semidefinite. To rectify this, we solved the following optimization problem to find $\hat{H}_{est}$, the closest positive semidefinite matrix to $\hat{H}$:

$$\begin{aligned} \underset{\hat{H}_{est}}{\text{minimize}} \quad & ||\hat{H}_{est} - \hat{H}||_F \\ \text{subject to} \quad & \hat{H}_{est} \geq 0. \end{aligned} \tag{30}$$

However, the solution of this problem no longer had the sparsity pattern of $\hat{H}$. As a result, this sparsity pattern had to be enforced on $\hat{H}_{est}$ to give $\hat{H}_{pd}$. By manual check, it was determined that $\hat{H}_{pd}$ was positive semidefinite. Further in the report, when $\hat{H}$ is used, it is implied that in the actual implementation the positive semidefinite matrix $\hat{H}_{pd}$ is being used.

Next, to partition problem in (25) into local problems, the following feasibility problem was solved for $\hat{H}^i$:

$$\hat{H}^i \geq 0, \tag{31}$$

$$\sum_i W^i \hat{H}^i (W^i)^T = \hat{H}, \tag{32}$$

where $W^i \in \mathbb{R}^{s \times \hat{q}^i}$ is constructed by removing from the identity matrix all block columns corresponding to agents $j \notin \hat{\mathcal{N}}_i$.

Then, the local optimization problems are given by:

$$\begin{aligned} \underset{z^i}{\text{minimize}} \quad & \frac{1}{2}(z^i)^T \hat{H}^i z^i + \mathcal{B}_i^T z_i^i + (\hat{\gamma}^i)^T (z^i - \bar{z}^i) + \frac{\rho}{2} \|z^i - \bar{z}^i\|_2^2 \\ \text{subject to} \quad & \mu_i \geq 0. \end{aligned} \tag{33}$$

We denote the optimal solution of this problem by $(z^i)^+$. The algorithm is given as follows:

Algorithm 3: Dual ADMM

```
Initialize $\hat{\gamma}^i = 0$ and $\bar{z}^i = 0$ ;
For every node $i$:
for $k \geq 0$ do
    1) Calculate $(z^i)^+$ by solving (33);
    2) Send $(z^i_j)^+$ to all $j \in \hat{\mathcal{N}}_i$;
    3) Calculate $\bar{z}_i^+ = \frac{1}{|\hat{\mathcal{N}}_i|} \sum_{j \in \hat{\mathcal{N}}_i} (z_i^j)^+$;
    4) Send $(\bar{z}_i)^+$ to all $j \in \mathcal{N}_i$ ;
    5) Stack up $(\bar{z}^i)^+ = [(\bar{z}_j)^+]_{j \in \hat{\mathcal{N}}_i}$ ;
    6) Update $(\hat{\gamma})^+ = \hat{\gamma}^i + \rho((z^i)^+ - (\bar{z}^i)^+)$
end
```

4 Simulation results and discussion

The three aforementioned algorithms were implemented in MATLAB. To make the simulation closer to reality, the execution of local optimization at the nodes was parallelized. The optimization of augmented Lagrangian in primal and dual ADMM was implemented using the built-in function *quad_prog*. This has resulted in a considerable improvement in performance, as opposed to using CVX for optimization, as the overhead was reduced greatly. CVX was still used to find $\hat{H}^i$ in the dual ADMM.

Figure 2 shows the convergence , from the top left corner towards the center of the figure, of the accelerated gradient method as the number of iterations grow. It can be seen that the state progressively gets closer to the optimal solution, which we consider is given by CVX (shown with punctuated line). The solution given in the first step reaches the set-point in three seconds yet does not respect the dynamics of the system, and in every iteration thereafter the solution gets closer to the feasible set.

Figure 3 compares the control performance of the Primal ADMM method in [2] against CVX (shown with punctuated line), which we consider to be the optimal solution. Sub-optimality is expected as this method utilizes consensus among local agents with limited global information. The algorithms were run with a receding horizon of $N = 20$.

Table 1 shows the comparison in running times between accelerated gradient method and primal ADMM. These running times are average running times over 100 optimizations. One can observe an increase in the running time with the increase of the receding horizon N. This is reasonable as the dimensionality of the optimization problem is increased. It is worth noting how the average time taken per optimization for Primal ADMM remains almost constant until the horizon jumps from 50 to 100 steps, this indicates that for this optimization problem, the time it takes to solve it depends primarily on the overhead in parallelization for horizons between 3 to 100 and on the size of the problem thereafter.

Clearly, primal ADMM algorithm converges to the solution faster than AGD. The comprehensive results for dual ADMM were not produced due to the time constraints. The main bottleneck of simulating dual ADMM lies in generation of matrices $\hat{H}^i$.

Both algorithms in [3] and [2] use some variant of the matrix $\mathcal{A}H^{-1}\mathcal{A}^T$. We have found in our simulations that this matrix is ill-conditioned and this often cause accumulation of error and divergence. This is due to the fact that typically

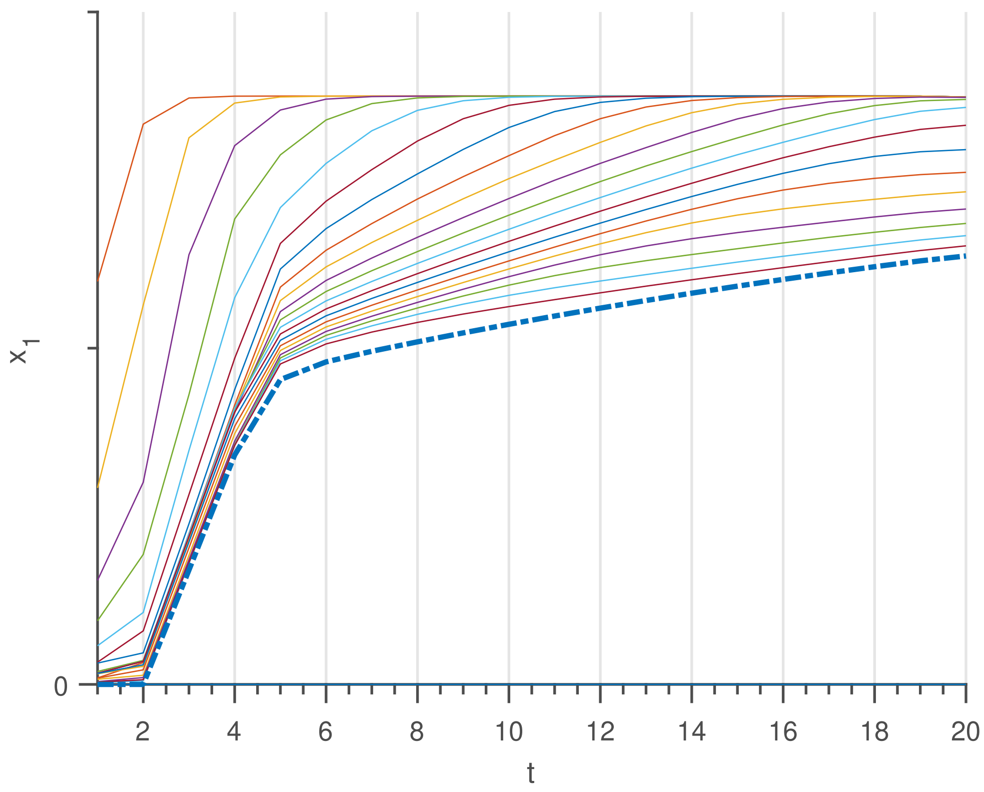

Figure 2: The evolution of the state x_1 with the increasing number of iterations dual-primal iterations in [3]

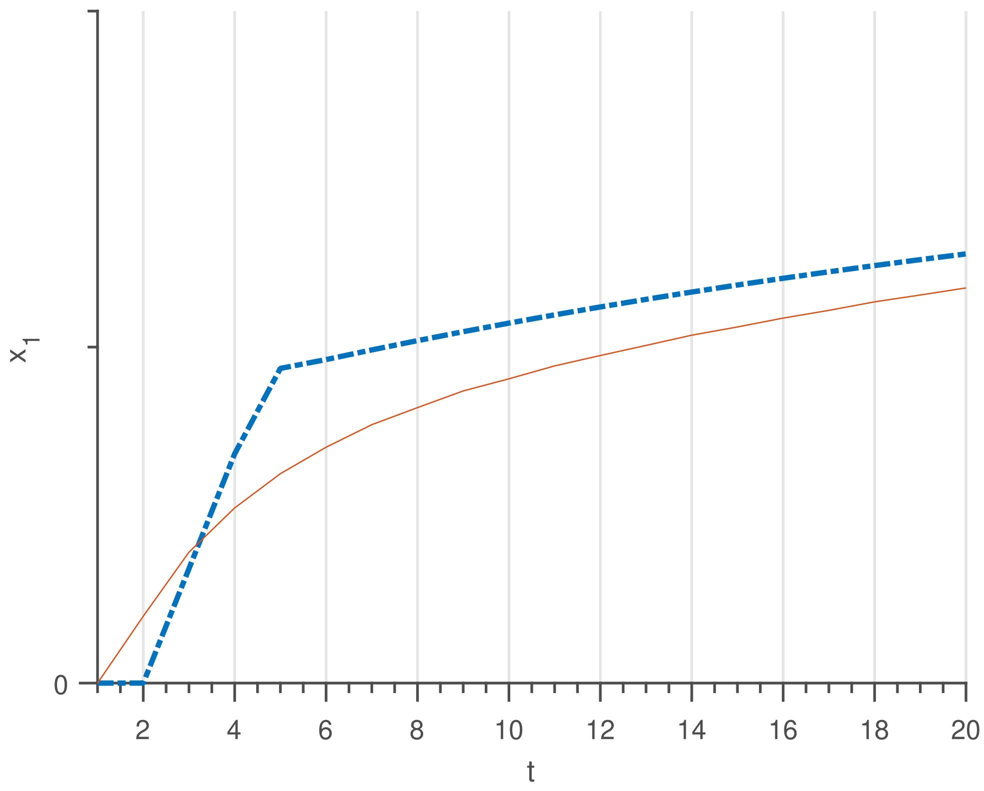

Figure 3: Performance comparison between CVX and the Primal ADMM as presented in [2]

Table 1: Comparison of running times of accelerated gradient method and primal ADMM algorithms

Algorithm \ N	3	5	10	20	50	100	200
AGD [3]	*	*	*	8592	>8592	>8592	>8592
Primal ADMM [2]	0.3880	0.3890	0.3899	0.3913	0.3803	0.4793	0.7175

in DMPC the number of constraints is greater than the number of variables and $\mathcal{A}H^{-1}\mathcal{A}^T$ has rank lower or equal than the number of variables. The simulation results in [3] were performed exclusively on problems where the number of constraints was less than the number of variables, avoiding this problem and, we believe, not reflecting the real-life applications. The benchmark used was chosen with the aim of testing these algorithms on a real world DMPC problem. That being said, we also understand that given the small amount of subsystems in the benchmark used these algorithms were not able to benefit fully from their decentralized aspects.

References

[1] P. Giselsson. A generalized distributed accelerated gradient method for distributed model predictive control with iteration complexity bounds. In *2013 American Control Conference*, pages 327–333, June 2013.

[2] 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)*, pages 6598–6603, Dec 2017.

[3] Pontus Giselsson, Minh Dang Doan, Tamás Keviczky, Bart De Schutter, and Anders Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829 – 833, 2013.

[4] K. H. Johansson. The quadruple-tank process: a multivariable laboratory process with an adjustable zero. *IEEE Transactions on Control Systems Technology*, 8(3):456–465, May 2000.

[5] I. Alvarado, D. Limon, D. Muñoz de la Peña, J.M. Maestre, M.A. Ridao, H. Scheu, W. Marquardt, R.R. Negenborn, B. De Schutter, F. Valencia, and J. Espinosa. A comparative analysis of distributed mpc techniques applied to the hd-mpc four-tank benchmark. *Journal of Process Control*, 21(5):800 – 815, 2011. Special Issue on Hierarchical and Distributed Model Predictive Control.

[6] Dang Doan, Tamás Keviczky, Ion Necoara, Moritz Diehl, and Bart De Schutter. A distributed version of han's method for dmpc of dynamically coupled systems with coupled constraints. *IFAC Proceedings Volumes*, 42(20):240 – 245, 2009. 1st IFAC Workshop on Estimation and Control of Networked Systems.

[7] Yu. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Technical report, Reports of the Academy of Sciences of the USSR, 1983.