Asynchronous Cooperative method for Distributed Model Predictive Control

Alessio Maffei, Luigi Iannelli, Luigi Glielmo, Francesco Borrelli

***Abstract*— In this paper we consider a distributed solution of the model predictive control problem (DMPC), based on the block version of the Jacobi algorithm applied to the dual problem. In order to accelerate the convergence, a Nesterov's schema can be considered, but the updating rule coming out in such way is fully distributed and parallel, but synchronous. This assumption is often unrealistic in real-life large-scale systems. For this reason an asynchronous version of the method has been proposed and the convergence properties have been studied. Numerical experiments show the effectiveness of the approach by comparing it with the methods presented in the literature.**

***Index Terms*— Distributed model predictive control, networks, convex optimization.**

I. Introduction

During the recent years, the use of MPC control strategy for designing high performance model-based controllers has increased enormously due to its methodical way to optimize the behavior of dynamic systems taking into account system states and control input constraints. However, the size of the optimization problems faced today by engineers has grown and a centralized solution is sometime impossible to find. The idea behind DMPC is decomposing the global optimization problem into smaller problems assigned to a certain number of entities with limited computational capacities and local vision of the problem [1]; they have to cooperate themselves to reach the global optimum.

The problem can be easily found in many applications such as the control of large-scale power networks [2]; the integration of electrical vehicle charging control into power grid [3]; the control of water flow delivery canals [4]; and the control of air conditioning for buildings [5].

In general, the distributed iterative methods rely on the classical gradient algorithms to solve the dual optimization problem. They are suitable for both small and large scale problem and recently they have been applied to DMPC [6]. Despite their simplicity and low complexity, the gradient methods suffer from slow converge rate ($O(1/k)$), [7]. The minimization of constrained smooth functions by gradient algorithms has been accelerated by using a Nesterov's approach in [8], such that a $O(1/k^2)$ rate is achieved. That result has been used by [9] for DMPC problems.

Another way to accelerate a distributed algorithm is to apply the alternating direction method of multipliers (ADMM) [10]. They rely on the decomposability property of the augmented dual problem and they show robust convergence properties. The ADMM has been used for solving DMPC in [1] (Ch. 7) and [4].

In this paper, we study the application of an accelerated non-smooth second order Newton method for solving the DMPC problem. Starting from the dual problem we use the parallel Jacobi method to approximate the Newton descent direction [11]. The proposed approach is partially inspired by [12], where a structured block diagonal Hessian is computed offline by solving an SDP problem. However, that work suffers from scalability issues since the offline problem is solved every time the network changes, it is centralized and memory consuming, as stated by the authors. In addition, the cooperative DMPC algorithms presented in the literature require a strong synchronization among all the nodes to make the correct exchange of data. The requirement of a global clock to be instantaneously propagated to all the systems in a large-scale system is quite unrealistic and it limits practical application [13]. For this reason, we investigated an asynchronous version of the accelerated method, together with its convergence properties. The comparison with the most recent distributed methods for DMPC (i.e., [9], [10], [12]) is shown through some numerical and application examples.

II. Problem setup

We consider a linear time-invariant discrete-time system composed of N local subsystems. The state space model of the full system is

$$x(t+1) = Ax(t) + Bu(t), \quad x(0) = \hat{x},$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ are the state and input vector, respectively, at time t, obtained by stacking in one column the local states $x_i(t) \in \mathbb{R}^{n_i}$ and the manipulated variables $u_i(t) \in \mathbb{R}^{m_i}$ for each subsystem, i.e., $x(t) = [x_1(t)^\top, \dots, x_N(t)^\top]^\top$, $u(t) = [u_1(t)^\top, \dots, u_N(t)^\top]^\top$. The subsystems dynamic can be written as

$$x_i(t+1) = \sum_{j=1}^{N} \left(A_{ij} x_i(t) + B_{ij} u_i(t) \right), \quad x_i(0) = \hat{x}_i,$$

where $A_{ij} \in \mathbb{R}^{n_i \times n_j}$ and $B_{ij} \in \mathbb{R}^{n_i \times m_j}$ are the ij-th block of the full state matrix A and input matrix B, and they define the interactions among the subsystem i and the subsystem j. Indeed, it usually happens that matrices A and B are sparse, i.e., $A_{ij} = 0$ and $B_{ij} = 0$ for most $i, j \in \mathcal{V}$.

The full system can be also modeled by a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where the set of nodes $\mathcal{V} = \{1, \dots, N\}$ represents the local subsystems, and the set of links $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ models the subsystems interactions:

$$\mathcal{E} = \{(i,j) \,|\, A_{ij} \neq 0 \vee B_{ij} \neq 0, i, j \in \mathcal{V}\}. \tag{1}$$

Dipartimento di Ingegneria, Università del Sannio, Piazza Roma 21, 82100 Benevento, Italy, email: {amaffei, luigi.iannelli, luigi.glielmo}@unisannio.it

Mechanical Engineering, University of California, Berkeley, CA, email: fborrelli@berkeley.edu

The set of neighbors of node i is defined as $\mathcal{N}_i = \{j : (i,j) \in \mathcal{E}\}$. The nodes are able to exchange information among them through a communication network whose graph is assumed being coincident with $\mathcal{G}$.

For each node i local quadratic stage and terminal costs are associated: $\ell_i(x_i(t), u_i(t)) = \frac{1}{2}\left(x_i(t)^\top Q_i x_i(t) + u_i(t)^\top R_i u_i(t)\right)$, $\ell_{T,i}(x_i(T)) = \frac{1}{2}\left(x_i(T)^\top S_i x_i(T)\right)$ respectively; where $Q_i \in \mathbb{R}^{n_i \times n_i}$, $R_i \in \mathbb{R}^{m_i \times m_i}$ and $S_i \in \mathbb{R}^{n_i \times n_i}$ are assumed symmetric positive definite. This leads to the following separable cost functions for the entire plant:

$$\ell(x(t), u(t)) = \sum_{i=1}^{N} \ell_i(x_i(t), u_i(t)) = \frac{1}{2}\left(x(t)^\top Q x(t) + u(t)^\top R u(t)\right)$$
$$\ell_T(x(T)) = \sum_{i=1}^{N} \ell_{T,i}(x_i(T)) = \frac{1}{2} x(T)^\top S x(T), \tag{2}$$

with $Q = \text{blkdiag}(Q_1, \dots, Q_N)$, $R = \text{blkdiag}(R_1, \dots, R_N)$ and $S = \text{blkdiag}(S_1, \dots, S_N)$.

Moreover, the local variables $x_i(t)$ and $u_i(t)$ are required to be bounded for each time t by the box constraints:

$$\mathcal{X}_i = \{x_i(t) \,|\, \underline{x}_i \leqslant x_i(t) \leqslant \overline{x}_i\},\ \mathcal{U}_i = \{u_i(t) \,|\, \underline{u}_i \leqslant u_i(t) \leqslant \overline{u}_i\},$$

for all $i \in \mathcal{V}$, with $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ the extended real-line. These sets lead to the global constraints $x(t) \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$, $u(t) \in \mathcal{U} = \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$.

Fixing the initial state $x(0)$ and a control horizon T, the finite horizon constrained optimization problem to be solved in the DMPC scheme is

$$\begin{aligned} J_T := \min_{x,u}\ & \ell_T(x(T)) + \sum_{t=0}^{T-1} \ell(x(t), u(t)) \\ \text{s.t. } & x(t+1) = Ax(t) + Bu(t), \quad t = 0, \dots, T-1 \\ & x(t) \in \mathcal{X}\ , u(t) \in \mathcal{U}, \qquad\quad t = 0, \dots, T-1 \\ & x(T) \in \mathcal{X}, \\ & x(0) = \hat{x}. \end{aligned} \tag{3}$$

The problem (3) is then formulated as a constrained quadratic optimization problem:

$$\begin{aligned} \min_{z \in \mathcal{Z}}\ & \frac{1}{2} z^\top H z \\ \text{s.t. } & Cz = c, \end{aligned}$$

where $z = [x^\top, u^\top]^\top$, $x = [x(1)^\top, \dots, x(T)^\top]^\top$, $u = [u(0)^\top, \dots, u(T-1)^\top]^\top$, $\mathcal{Z} = \mathcal{X} \times \mathcal{U}$, $H = \text{blkdiag}\,(Q, \dots, Q, S, R, \dots, R)$ and

$$C = \begin{bmatrix} I & 0 & \cdots & 0 & -B & 0 & \cdots & 0 \\ -A & I & \cdots & 0 & 0 & -B & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -A & I & 0 & 0 & \cdots & -B \end{bmatrix}, c = \begin{bmatrix} A\hat{x} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The dual problem is then considered by introducing the multipliers $\lambda \in \mathbb{R}^{nT}$ corresponding to the dynamic constraints:

$$\hat{q}(\lambda) := \min_{z \in \mathcal{Z}} \frac{1}{2} z^\top H z - \langle \lambda, Cz - c \rangle. \tag{4}$$

We refer to $\mathcal{L}(z, \lambda) = \frac{1}{2} z^\top H z - \langle \lambda, Cz - c \rangle$ as the Lagrangian function, and $\hat{q}(\lambda)$ as the dual function.

Since $\ell_i(x_i(t), u_i(t))$ and $\ell_{T,i}(x_i(T))$ are strictly convex $\forall i, t$, if the optimization problem is feasible then the duality gap is zero [14]. Thus, we can solve the dual (4) and compute the associated primal optimal variables as:

$$z(\lambda^*) = \operatorname*{argmin}_{z \in \mathcal{Z}} \mathcal{L}(z, \lambda^*) \tag{5}$$

where λ^* is the optimal value of the dual function. By exploiting the separability properties of cost functions (2) the optimal states x_i^* and control inputs u_i^* components are:

$$\begin{aligned} x_i(t, \lambda^*) &= \operatorname*{argmin}_{y \in \mathcal{X}_i} \frac{1}{2} y^\top Q_i y + y^\top \left(\sum_{j \in \mathcal{N}_i} A_{ji}^\top \lambda_j^*(t) - \lambda_i^*(t-1) \right) \\ u_i(t, \lambda^*) &= \operatorname*{argmin}_{v \in \mathcal{U}_i} \frac{1}{2} v^\top R_i v + v^\top \left(\sum_{j \in \mathcal{N}_i} B_{ji}^\top \lambda_j^*(t) \right) \\ x_i(T, \lambda^*) &= \operatorname*{argmin}_{y \in \mathcal{X}_i} \frac{1}{2} y^\top S_i y - y^\top \lambda_i^*(T-1) \end{aligned} \tag{6}$$

where $\lambda_i(t) \in \mathbb{R}^{n_i}$ is the vector of dual variables associated to the dynamic of subsystem i at time t. The computation of the primals is naturally distributed with respect to the $\mathcal{G}$.

We are therefore interested in computing

$$\lambda^* = \arg\min_{\lambda}\ q(\lambda), \tag{7}$$

where, for notational convenience, we introduced the convex dual function $q(\lambda) \triangleq -\hat{q}(\lambda)$.

A. Non-smooth Newton's method

Newton's method is a classic iterative procedure for minimizing convex functions. Starting with an initial value λ^0, we update λ^k by minimizing $q(\lambda)$ at each step k,

$$\lambda^{k+1} = \lambda^k + \alpha d^k, \tag{8}$$

where α is a properly chosen step size and d is the descent Newton's direction. The vector d is solution of the system

$$V^k d^k = -g^k,$$

where g^k is the gradient of the negative of the dual function at the step k, i.e., $g^k = g(\lambda^k) = \nabla q(\lambda^k)$, and $V^k = V(\lambda^k)$ is the negative of the dual function Hessian.

Considering (4), we can straightforwardly compute the gradient $g(\lambda)$ as function of the primal optimizers:

$$g(\lambda^k) = Cz(\lambda^k) - c.$$

The gradient $g(\lambda)$ is Lipschitz continuous with Lipschitz constant $\kappa = \|CH^{-1}C^\top\|$ [15, Th. 7, Rem. 5].

Since g is locally Lipschitz, according to Rademacher's Th. [16], g is differentiable almost everywhere. Let D_g be the set of the point at which g is differentiable, then the generalized Hessian is the set $\partial g(\lambda) = \overline{\text{co}}\left\{\lim_{\overline{\lambda} \to \lambda, \overline{\lambda} \in D_g} \nabla g(\overline{\lambda})\right\}$, where $\overline{\text{co}}$ defines the convex hull. The Hessian of the unconstrained (smooth) version of the DMPC problem is the constant matrix $V^k = V = CH^{-1}C^\top$, which is an extreme point of $\partial g(\lambda)$. Indeed, for $\overline{\lambda} \in \{\lambda | z(\lambda) \in \mathcal{Z}\}$ we have

$$\lim_{\lambda \to \overline{\lambda}, \lambda \in D_g} \nabla g(\lambda) = V.$$

Due to the strict convexity and separability of $l(x(t), u(t))$, it follows that H^{-1} exists and is positive block diagonal. Furthermore, the structure of matrix C, that is full row rank, makes V invertible. However, the inverse V^{-1} is a dense matrix, and that leads (8) to be not distributed. For this reason, we looked for a distributed algorithm and we approximated the Hessian V with its block diagonal version (see Section III).

B. Nesterov's acceleration

The accelerated version of the Newton method for problem (7) is defined by Nesterov's method [8]:

$$\begin{aligned}\tilde{\lambda}^k &= \lambda^k + \frac{k-1}{k+2}(\lambda^k - \lambda^{k-1}),\\ \lambda^{k+1} &= \tilde{\lambda}^k - \alpha V^{-1} g(\tilde{\lambda}^k),\end{aligned} \tag{9}$$

for all k, with $\lambda^{-1} = \lambda^0$. The acceleration consists in performing a simple step of Newton's method (8) from $\tilde{\lambda}^k$ to λ^{k+1} and, using $\tilde{\lambda}^k$ as a correction of λ^k based on the previous value. The correction (9) leads to the accelerated primal variables $z(\tilde{\lambda}^k)$ in (5), so that the gradient becomes

$$g(\tilde{\lambda}^k) = Cz(\tilde{\lambda}^k) - c. \tag{10}$$

III. Distributed Non-Smooth Newton's method

The distributed method proposed here is based on the the Jacobi approach [17] applied to the accelerated Newton method. Therefore, the approximated descent directions are

$$d_i(\tilde{\lambda}^k) = -V_i^{-1} g_i(\tilde{\lambda}^k), \quad \forall i,$$

where the gradient is computed accordingly to (10) and the block components of the Hessian matrix are $V_i = \mathrm{blkdiag}(V_{t,i})$, $\forall t$, with:

$$V_{t,i} = \begin{cases} Q_i^{-1} + \sum_{j\in\mathcal{N}_i} B_{ij} R_j^{-1} B_{ij}^\top & \text{if } t = 0\\ Q_i^{-1} + \sum_{j\in\mathcal{N}_i} A_{ij} Q_j^{-1} A_{ij}^\top + B_{ij} R_j^{-1} B_{ij}^\top & \text{if } t \in [1, T\text{-}2]\\ S_i^{-1} + \sum_{j\in\mathcal{N}_i} A_{ij} Q_j^{-1} A_{ij}^\top + B_{ij} R_j^{-1} B_{ij}^\top & \text{if } t = T-1. \end{cases}$$

Therefore, the computation of the gradient g_i is completely distributed. According to the definition of C matrix, the node i requires the accelerated primals from its neighbors $\mathcal{N}_i$ to compute $g_i(\tilde{\lambda}^k)$ of (10). The Hessian blocks $V_{t,i}$ are also distributed and they can be computed off-line or whenever a node is added/removed to/from the network (plug & play).

Thus, the resulting accelerated approximated Newton's method for each node i is characterized by the iterations

$$\begin{aligned}\tilde{\lambda}_i^k &= \lambda_i^k + \frac{k-1}{k+2}(\lambda_i^k - \lambda_i^{k-1}),\\ \lambda_i^{k+1} &= \tilde{\lambda}_i^k - \alpha_i V_i^{-1} g_i(\tilde{\lambda}^k).\end{aligned} \tag{11}$$

The distributed $update\_function$ for the proposed method is given in the Algorithm 1.

Algorithm 1: $update\_function$ for node i at step k.

Data: Q_i, R_i, S_i, V_i, A_{ij}, B_{ij} $\mid$ $j \in \mathcal{N}_i$, $\lambda_i^{-1} = \lambda_i^0$;

1) receive λ_j^k from $j \in \mathcal{N}_i$;
2) compute $x_i(\tilde{\lambda}^k)$, $u_i(\tilde{\lambda}^k)$ according to (6);
3) receive $x_i(\tilde{\lambda}^k)$, $u_i(\tilde{\lambda}^k)$ from $j \in \mathcal{N}_i$;
4) compute $g_i(\tilde{\lambda}^k)$ according to (10) and update $\lambda_i^{k+1} \leftarrow \tilde{\lambda}_i^k - \alpha_i V_i^{-1} g_i(\tilde{\lambda}^k)$;

A. Convergence analysis

In order to make the convergence analysis easier, we reformulate the iterations (11) in the following form:

$$\begin{aligned}\tilde{\lambda}^k &= (1-\gamma^k)\lambda^k + \gamma^k \lambda^{k-1},\\ \lambda^{k+1} &= \tilde{\lambda}^k - \mathcal{A}\hat{V}^{-1} g(\tilde{\lambda}^k),\end{aligned} \tag{12}$$

where $\gamma^k = \frac{1-\theta^{k-1}}{\theta^k}$ with $\theta^k = \frac{k+2}{2}$, and $\mathcal{A} = \mathrm{blkdiag}(\alpha_1 I, \ldots, \alpha_N I)$ and $\hat{V} = \mathrm{blkdiag}(V_i)$, $\forall i$.

***Theorem** 1:* If the step sizes α_i are chosen as:

$$\mathcal{A} > 0 \,:\, M = \hat{V}\mathcal{A}^{-1} - \kappa I \succeq 0, \tag{13}$$

then the Algorithm 1 converges (i.e., $\lim_{k\to\infty} \lambda^k = \lambda^*$), and the convergence rate for $k \geqslant 1$ is

$$q(\lambda^k) - q(\lambda^*) \leqslant \frac{2||\lambda^0 - \lambda^*||^2_{\hat{V}\mathcal{A}^{-1}}}{(k+1)^2}$$

where $\kappa = \|CH^{-1}C^\top\|$ is the Lipschitz constant [15, Th. 7, Rem. 5], and $||y||_M = \sqrt{y^\top M y}$.

Proof: The proof is a generalization of [18]. ■

***Corollary** 1:* If the Algorithm 1 converges, then the rate of convergence for the primal variables is:

$$||z^k - z^*||_2^2 \leqslant \frac{4||\lambda^0 - \lambda^*||^2_{\hat{V}\mathcal{A}^{-1}}}{\mu_1(H)(k+1)^2}$$

with $\mu_1(H)$ the smallest eigenvalues of H.

Proof: The proof for the primals convergence rate straightforwardly follows from [9, Th.3]. ■

***Corollary** 2:* If the Algorithm 1 converges, then the equality constraint violation is bounded by:

$$||(Cz(\lambda^k) - d) - (Cz(\lambda^*) - d))\,||_2^2 \leqslant \frac{4\kappa||\lambda^0 - \lambda^*||^2_{\hat{V}\mathcal{A}^{-1}}}{(k+1)^2}.$$

Proof: From the *Descent Lemma* [17, Ch. 3] we have:

$$||\nabla q(\lambda^*) - \nabla q(\lambda^k)||_2^2 \leqslant 2\kappa(q(\lambda^k) - q(\lambda^*) - \langle \nabla q(\lambda^*), \lambda^k - \lambda^* \rangle).$$

Since λ^* is a solution of the convex dual function q, we have $\langle \nabla q(\lambda^*), \lambda^k - \lambda^* \rangle \geqslant 0$. Thus, the statement is straightforwardly derived by using Theorem (1). ■

IV. Asynchronous Distributed Algorithm

For the DMPC problem, iteration (8) leads itself to the synchronous Algorithm 1: a global clock signal k has to be shared by to all the nodes to guarantee the right exchange of information. Generally, this assumption has limits on the practical real-life large-scale systems.

An asynchronous version of the proposed DMPC algorithm is studied here. The nodes have not to wait for the neighbors states to compute the update [13], [17], i.e., the nodes keep iterating the Algorithm 1 by using the most recent neighbor's variables even though they are not updated. This allows handling the situations where the computational and the data transfer times of the nodes are not the same.

Therefore, a local time variable k_i is introduced as well as the sets K_i of times at which the node i updates its own state, i.e., $k_i \to k_i + 1$ if $k \in K_i$ [13]. We also denoted as $k_i(k)$ the value assumed by local k_i at time k, where k is artificial time variable (not known by any node). Thus, the asynchronous algorithm is characterized by the update rule:

$$\tilde{\lambda}_i^{k_i} = (1-\gamma_i^{k_i})\lambda_i^{k_i} + \gamma_i^{k_i}\lambda_i^{k_i-1} \qquad \forall i, k \in K_i \tag{14a}$$

$$\lambda_i^{k_i+1} = \tilde{\lambda}_i^{k_i} - \alpha_i V_i^{-1} g_i(\tilde{\lambda}^k) \qquad \forall i, k \in K_i. \tag{14b}$$

The vector $\tilde{\lambda}^k$ is heterogeneous in the sense that its components considered are at different time stamp, i.e., $\tilde{\lambda}_i^k = \tilde{\lambda}_i^{k_i(k)}$, while $\gamma_i^{k_i} = \frac{1-\theta^{k_i-1}}{\theta^{k_i}}$ with $\theta^{k_i} = \frac{k_i+2}{2}$.

In order to guarantee the convergence of the asynchronous algorithm we make the following assumption [13], [17]:

Assumption 1: There exists a positive integer $\overline{K}$ such that for every i and $k \geqslant 0$, at least one of the set $\{k, \ldots, k + \overline{K} - 1\}$ belongs to K_i.

The synchronous update can be seen as the asynchronous one with $k_i = \overline{K}, \forall i$, where $\overline{K}$ is the time needed by the slowest node to perform the update.

We also force the *descent property along each coordinate*:

$$\lambda_i^{k_i+1} = \lambda_i^{k_i} \quad \text{if} \quad \nabla_i q(\tilde{\lambda}^k)^\top (\lambda_i^{k_i} - \tilde{\lambda}_i^{k_i}) \leqslant 0 \quad \forall i, k \in K_i. \tag{15}$$

Indeed, the vector λ^k is heterogeneous and $q(\lambda^{k+1})$ could be greater than $q(\lambda^k)$, leading the algorithm to instability.

Theorem 2: If the step sizes α_i are chosen according to (13), then the partially asynchronous algorithm (14) under the condition (15) converges with the rate

$$q(\lambda^k) - q(\lambda^*) = O\left(\frac{1}{k^2}\right).$$

Proof: Since the gradient $g(\lambda)$ is *Lipschitz continuous* from the *Descent Lemma* [17, Ch. 3] we have:

$$q(\lambda^{k+1}) \leqslant q(\tilde{\lambda}^k) + \nabla q(\tilde{\lambda}^k)^\top (\lambda^{k+1} - \tilde{\lambda}^k) + \frac{\kappa}{2} ||\lambda^{k+1} - \tilde{\lambda}^k||_2^2$$

We remind here that the approximated block diagonal Hessian $\hat{V}$ is symmetric positive definite by definition.

Using the convexity of dual function $q(\lambda)$ (i.e., $q(\lambda^k) \geqslant q(\tilde{\lambda}^k) + \nabla q(\tilde{\lambda}^k)^\top (\lambda^k - \tilde{\lambda}^k)$), and the update (14b) we obtain:

$$q(\lambda^{k+1}) - q(\lambda^k) \leqslant -\nabla q(\tilde{\lambda}^k)^\top (\lambda^k - \tilde{\lambda}^k) - ||\lambda^{k+1} - \tilde{\lambda}^k||^2_{\hat{V}\mathcal{A}^{-1} - \frac{\kappa}{2}}, \tag{16}$$

where $\nabla q(\tilde{\lambda}^k)^\top = -(\lambda^{k+1} - \tilde{\lambda}^k)^\top \hat{V}\mathcal{A}^{-1}$ by definition.

By Assumptions 1 and the conditions (13) and (15) we ensure $q(\lambda^{k+1}) - q(\lambda^k) \leqslant 0 \, \forall k$. We can rewrite (16) component-wise as

$$q(\lambda^{k+1}) - q(\lambda^k) \leqslant \sum_{i:k \in K_i} \nabla_i q(\tilde{\lambda}^k)^\top (\tilde{\lambda}_i^{k_i} - \lambda_i^{k_i}) - ||\lambda_i^{k_i+1} - \tilde{\lambda}_i^{k_i}||^2_{V_i \alpha_i^{-1} - \frac{\kappa}{2}} \tag{17}$$

By Assumption 1 we have $\theta^k \geqslant \theta^{k_i}, \forall i, k$. Therefore, multiplying (17) left-side by $(\theta^k - 1)$ and right-side by $(\theta^{k_i} - 1)$ for each component, and adding to $\delta^{k+1} \triangleq q(\lambda^{k+1}) - q(\lambda^*) \geqslant 0$, we get:

$$\theta^k \delta^{k+1} - (\theta^k - 1)\delta^k \leqslant \sum_{i:k \in K_i} \nabla_i q(\tilde{\lambda}^k)^\top (\theta^{k_i} \tilde{\lambda}_i^{k_i} - (\theta^{k_i} - 1)\lambda_i^{k_i} - \lambda^*) - \theta^{k_i} ||\lambda_i^{k_i+1} - \tilde{\lambda}_i^{k_i}||^2_{V_i \alpha_i^{-1} - \frac{\kappa}{2}}. \tag{18}$$

Note that by (12),

$$\theta^{k+1} \tilde{\lambda}^{k+1} - (\theta^{k+1} - 1)\lambda^{k+1} = \theta^k \lambda^{k+1} - (\theta^k - 1)\lambda^k$$

and $0 < (\theta^k)^2 - \theta^k < (\theta^{k-1})^2$. The same could be written for the λ^{k_i} and θ^{k_i} from (14). Therefore, multiplying (18) right side by θ^k and left-side by θ^{k_i} each component:

$$(\theta^k)^2 \delta^{k+1} - (\theta^{k-1})^2 \delta^k \leqslant \sum_{i:k \in K^i} -\frac{1}{2} ||w_i^{k_i+1}||^2_{V_i \alpha_i^{-1}} + \frac{1}{2} ||w_i^{k_i}||^2_{V_i \alpha_i^{-1}} - \frac{1}{2} ||w_i^{k_i+1} - w^{k_i}||^2_{V_i \alpha_i^{-1} - \kappa}, \, \forall k \geqslant 1,$$

where $w_i^{k_i+1} = \theta^{k_i} \lambda_i{}^{k_i+1} - (\theta^{k_i} - 1)\lambda_i^{k_i} - \lambda^*$.

Using (13), after some manipulations, we have:

$$(\theta^k)^2 \delta^{k+1} - (\theta^{k-1})^2 \delta^k \leqslant \sum_{i:k \in K_i} -\frac{1}{2} ||w_i^{k_i+1}||^2_{V_i \alpha_i^{-1}} + \frac{1}{2} ||w_i^{k_i}||^2_{V_i \alpha_i^{-1}}. \tag{19}$$

Since (19) holds for all $k \geqslant 1$ and the matrix $V_i \succeq 0$ we can sum (19) from $1 \to k-1$ obtaining the theorem statement. ■

V. Simulation results

The synchronous DMPC solution has been compared with some of the most recent algorithms available in the literature, i.e., the distributed ADMM (*D-ADMM*) [10], the distributed Accelerated Gradient (*D-AG*) [9] with the best step size (L_2-norm) and the generalized distributed Accelerated Gradient (*D-GAG*) [12]. For the latter work, the Hessian matrix is computed offline in a centralized way by solving an SDP problem. However, it suffers from scalability issues due to its high memory usage.

Simulations are performed in MATLAB using *CVX* toolbox [19]. We set the control horizon $T = 6$ and considered two scenarios: *1)* small-scale network consisting of $N = 3$ with $n_i = 5$ and $m_i = 1$ as [12]; *2)* large-scale network of $N = 100$ and $E = 267$ with $n_i = 2$ and $m_i = 1$; both are generated according to the Watts-Strogatz model [20]. The matrices Q_i, R_i and S_i have been randomly chosen as $\mathrm{rand}(0.1, 100)$, while states and inputs have been bounded by $\mathrm{rand}(-0.5, 0.5)$ and $\mathrm{rand}(-0.15, 0.15)$, respectively.

The small-case analysis is reported in Fig. 1, where the convergence of $f(x)$ and the relative error are depicted. Similar analysis has been done for the large-scale scenario (Fig. 2), however,in this case, it has not been possible simulate the *D-GAG* algorithm due to the high memory requirements. The Algorithm 1 shows better performance for both small and large scale scenarios. The iterations needed to achieve an error $\epsilon = 0.005$ are listed in Tab.I.

Finally, the asynchronous performance has been exploited by a comparison with the synchronous one. We supposed that the slowest node takes $\overline{K} = 4$ steps to run Algorithm 1. The results are shown in Fig. 3.

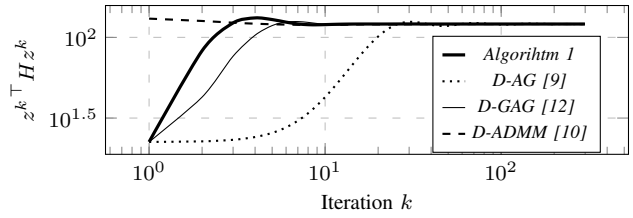
A. Example: DCOPF for power grids with ESS

Recently, the Optimal Power Flow (OPF) has emerged as a tool to optimize the power generation dispatch while satisfying the underlying network constraints [21]. The classical Alternate Current OPF (ACOPF) is a static nonlinear

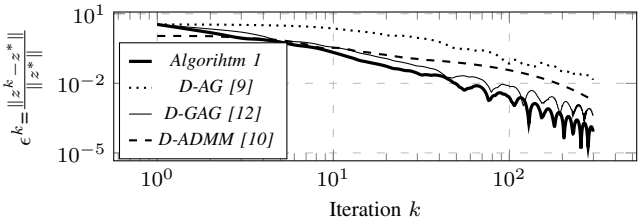
Alg.	N	ϵ	# iters Avg.	# iters Max
Alg.1	10	0.005	93.9	174
	50	0.005	179.4	234
	100	0.005	180.8	271
D-ADMM [10]	10	0.005	137.3	236
	50	0.005	306.3	479
	100	0.005	442.1	552
D-GAG [12]	10	0.005	119.3	222
D-AG L_2 [9]	10	0.005	205.8	326
	50	0.005	420.5	519
	100	0.005	458.1	605

TABLE I

ITERATIONS NEEDED TO ACHIEVE THE ϵ-ACCURACY.

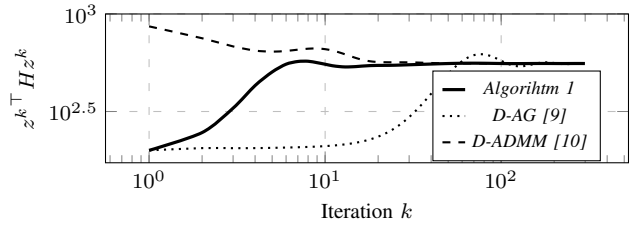


(a) *Primal objective with respect to dual iterations.*

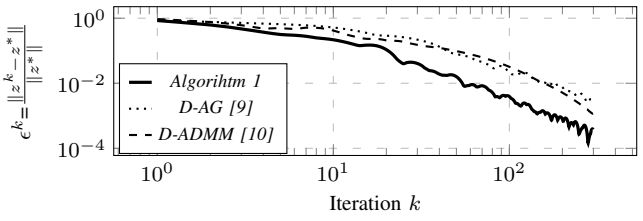


(b) *Relative error between the optimal solution z^* and z^k*

Fig. 1. Analysis of the convergence rates for small-scale scenario.



(a) *Primal objective with respect to dual iterations.*



(b) *Relative error between the optimal solution z^* and z^k*

Fig. 2. Analysis of the convergence rates for large-scale scenario.

nonconvex problem since it involves the minimization of total variable generation costs subject to nonlinear Kirchhoff's laws. However, it is usually approximated by a more tractable Direct Current OPF (DCOPF) that does not capture the reactive power and linearizes the power flow equation. The addition of power storages has introduced the opportunity to optimize, across time, the power dispatch in the electric grids [22].

The DC power injection equations are:

$$p_i(t) = -\sum_{j=1}^{N} \frac{1}{x_{ij}} \theta_{ij}(t),$$

where θ_{ij} is the phase difference between i and j and x_{ij} is reactance of line (i, j). The DC power equations can be written in matrix form as $p_t + DYD^\top \theta_t = 0$, where Y is the diagonal matrix of lines admittance $1/x_{ij}$, and $D \in \mathbb{R}^{N\times E}$ is the *incidence* matrix of the graph model $\mathcal{G}$ [23]. Therefore, the matrix $L_Y = DYD^\top \in \mathbb{R}^{N\times N}$ is the

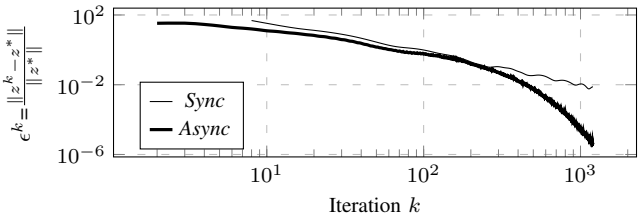


Fig. 3. Relative error of sync and async algorithms for large-scale scenario.

weighted *Laplacian matrix* of $\mathcal{G}$ and, therefore, it defines the coupling in (1).

The net power export $p_i(t)$ from node i at time t consists of the power from the node's generator $p_i^g(t)$ and battery $r_i(t)$ minus the power load $p_i^d(t)$, i.e., $p_i(t) = p_i^g(t) - p_i^d(t) + r_i(t)$. The $r_i(t)$ represents the power exchanged by the node i with its ESS at time t, it can be negative (ESS is charging) or positive (ESS is discharging). The State of Charge (SoC) of the storage is denoted by $s_i(t)$ and it evolves according to: $s_i(t+1) = s_i(t) - \Delta t\, r_i(t)$ [22], where the time axis is discretized uniformly with $\Delta t = 1\text{h}$.

Combining the above expressions leads to the following DCOPF problem with storage dynamics:

$$\begin{aligned}
\min_{s,g,\theta}\ & f_T(s(T)) + \sum_{t=0}^{T-1} f(s(t), p^g(t), \theta(t)) \\
\text{s.t.}\ & s(t+1) = s(t) + p^g(t) + L_Y\theta(t) - p^d(t), && t = 0,\dots,T-1 \\
& 0 \leqslant s(t) \leqslant \overline{s}, && t = 1,\dots,T \\
& 0 \leqslant p^g(t) \leqslant \overline{p}^g,\ \underline{\theta} \leqslant \theta(t) \leqslant \overline{\theta}, && t = 0,\dots,T-1 \\
& s(0) = \hat{s}.
\end{aligned}$$

The power grid considered is the IEEE-118 [24] and the results of the DCOPF based on DMPC are presented in Fig. 4. The position of storages, generators and loads in the network has been randomly chosen as well as the bounds $\overline{s}_i = \text{rand}(0, 0.2)\,\text{MW h}$ and $-\underline{\theta}_i = \overline{\theta}_i = 0.3\,\text{rad}$. We used $i = 1$ as reference bus with $\underline{\theta}_1 = \overline{\theta}_1 = 0\,\text{rad}$. We assumed a control horizon $T = 6\,\text{h}$ and an exact load prediction.

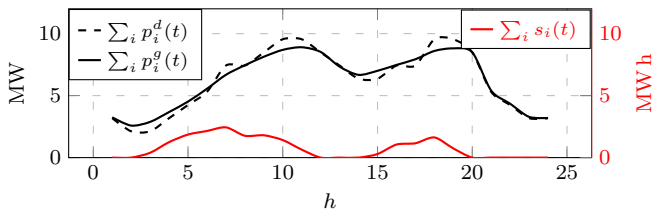
The Fig. 4(a) suggests that the implemented DCOPF uses the ESS judiciously during periods of high load demand in order to mitigate the generation and avoid more expensive peaks. Fig. 4(b) shows the relative error e^k for each DMPC iteration. To make the convergence faster, the DMPC iterations have been warmed started.

B. Example: Thermal Building Control

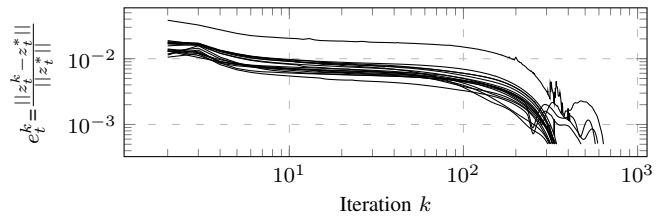
The heating, ventilation, and air conditioning (HVAC) system for buildings is an example of a decentralized control problem. The air is cooled by chilled water coils in the central Air Handling Units (AHU) and reheated by hot water coils in the Variable Air Volume (VAV) boxes in each thermal zone [5]. Therefore network communication topology that comes out for this problem has a star shape. A leaf node corresponds to a VAV box and the central hub to the AHU. The dynamic of the thermal zone $i \in \{1,\dots,N\}$ are modeled by the following model [5]:

$$C_i \dot{T}_i = u_c + u_i + \frac{T^{oa} - T_i}{R_i} + p_i^d,$$

where T_i is the temperature of zone i, T^{oa} is the outside temperature , u_i is the local reheating power input from VAV

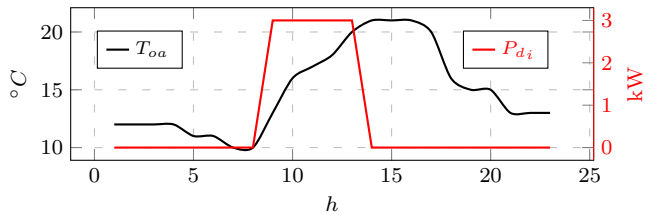


(a) *Power balancing for DCOPF with ESS*

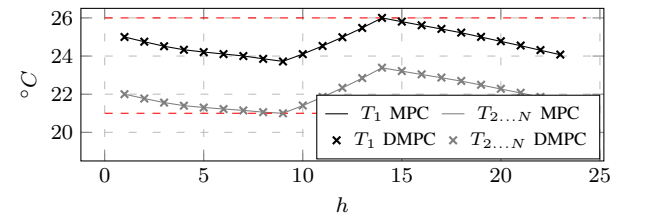


(b) *Relative error between the optimal solution z_t^* and z_t^k for each t*

Fig. 4. DCOPF with ESS based on DMCP for the IEEE-118 network.



(a) *Outside temperature*



(b) *Zone temperature*

Fig. 5. Analysis of building temperature control based MCP W/ and W/O distributed control for $N = 10$ thermal zones.

box, u_c is the cooling power input provided by the AHU system, p_i^d is the external disturbance thermal load generated by occupants, direct sunlight and electrical devices, while R_i and C_i are thermal resistive and capacitive parameters of i-th zone. The control objective is to regulate temperature in each thermal zone while minimizing energy consumption.

We used the same parameters as in [5]. The outside air temperature profile $T^{oa}(t)$ is depicted in Fig. 5(a), while, the Fig. 5(b) shows the difference between the controlled temperature in the closed loop using MPC and DMPC with the stopping condition: $||g^k|| < 0.005$.

VI. Conclusion

The purpose of this paper is to develop a distributed algorithm for solving the DMPC problems for large-scale networks in a scalable way. To this aim we proposed an asynchronous and accelerated Newton's method. The convergence of the algorithms has been studied and the effectiveness of the solution has been illustrated through numerical experiments. Finally, two real-case scenarios of the application of the proposed algorithm have been shown.

References

[1] J. M. Maestre and R. R. Negenborn, *Distributed model predictive control made easy*. Springer, 2014.

[2] H. Nong and X. Liu, "Nonlinear distributed mpc strategy with application to agc of interconnected power system," in *Control and Decision Conference (CCDC), 2013 25th Chinese*, May 2013, pp. 3935–3940.

[3] J. Rivera, P. Wolfrum, S. Hirche, C. Goebel, and H.-A. Jacobsen, "Alternating direction method of multipliers for decentralized electric vehicle charging control," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Dec 2013, pp. 6960–6965.

[4] R. Costa, J. Lemos, J. Mota, and J. Xavier, "D-admm based distributed mpc with input-output models," in *Control Applications (CCA), 2014 IEEE Conference on*, Oct 2014, pp. 699–704.

[5] Y. Ma, A. Kelman, A. Daly, and F. Borrelli, "Predictive control for energy efficient buildings with thermal storage: Modeling, stimulation, and experiments," *Control Systems, IEEE*, vol. 32, no. 1, pp. 44–64, Feb 2012.

[6] P. Giselsson and A. Rantzer, "Distributed model predictive control with suboptimality and stability guarantees," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Dec 2010, pp. 7272–7277.

[7] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.

[8] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[9] P. Giselsson, M. Doan, T. Keviczky, B. De Schutter, and A. Rantzer, "Accelerated gradient methods and dual decomposition in distributed model predictive control," *Automatica*, vol. 49, no. 3, pp. 829–833, Mar. 2013.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[11] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA, USA: Athena Scientific, 1998.

[12] P. Giselsson, "A generalized distributed accelerated gradient method for distributed model predictive control with iteration complexity bounds," in *American Control Conference (ACC), 2013*, June 2013, pp. 327–333.

[13] D. P. Bertsekas and J. N. Tsitsiklis, "Some aspects of parallel and distributed iterative algorithms - a survey," *Automatica*, vol. 27, no. 1, pp. 3 – 21, 1991.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[15] S. Richter, C. Jones, and M. Morari, "Certification Aspects of the Fast Gradient Method for Solving the Dual of Parametric Convex Programs," *Mathematical Methods of Operations Research*, vol. 77, no. 3, pp. 305–321, Jan. 2013.

[16] H. Federer, *Geometric Measure Theory*. Berlin Heidelberg: Springer-Verlag, 1969.

[17] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[18] W. Zuo and Z. Lin, "A generalized accelerated proximal gradient approach for total-variation-based image restoration," *Image Processing, IEEE Transactions on*, vol. 20, no. 10, pp. 2748–2759, Oct 2011.

[19] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[20] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 409–10, 1998.

[21] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: a bibliographic survey i," *Energy Systems*, vol. 3, no. 3, pp. 221–258, 2012.

[22] K. Chandy, S. Low, U. Topcu, and H. Xu, "A simple optimal power flow model with energy storage," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, Dec 2010, pp. 1051–1057.

[23] C. Godsil and G. Royle, *Algebraic Graph Theory*. New York, NY, USA: Springer, 2001.

[24] IEEE, "Power systems test case 118 bus," https://www.ee.washington.edu/research/pstca/, Dec. 2015.