Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC[☆]

Ion Necoara[a], Dragos Clipici[a]

[a] *University Politehnica Bucharest, Automatic Control and Systems Engineering Department, 060042 Bucharest, Romania*
(e-mail: ion.necoara@acse.pub.ro, d.clipici@acse.pub.ro)

Abstract

In this paper we propose a parallel coordinate descent algorithm for solving smooth convex optimization problems with separable constraints that may arise e.g. in distributed model predictive control (MPC) for linear network systems. Our algorithm is based on block coordinate descent updates in parallel and has a very simple iteration. We prove (sub)linear rate of convergence for the new algorithm under standard assumptions for smooth convex optimization. Further, our algorithm uses local information and thus is suitable for distributed implementations. Moreover, it has low iteration complexity, which makes it appropriate for embedded control. An MPC scheme based on this new parallel algorithm is derived, for which every subsystem in the network can compute feasible and stabilizing control inputs using distributed and cheap computations. For ensuring stability of the MPC scheme, we use a terminal cost formulation derived from a distributed synthesis. Preliminary numerical tests show better performance for our optimization algorithm than other existing methods.

Keywords: Coordinate descent optimization, parallel algorithm, (sub)linear convergence rate, distributed model predictive control, embedded control.

[☆]The research leading to these results has received funding from: the European Union, Seventh Framework Programme (FP7/2007–2013) under grant agreement no 248940; CNCSIS-UEFISCSU (project TE, no. 19/11.08.2010); ANCS (project PN II, no. 80EU/2010); Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

1. Introduction

Model predictive control (MPC) has become a popular advanced control technology implemented in network systems due to its ability to handle hard input and state constraints [20]. Network systems are usually modeled by a graph whose nodes represent subsystems and whose arcs indicate dynamic couplings. These types of systems are complex and large in dimension, whose structures may be hierarchical and they have multiple decision-makers (e.g. process control [21], traffic and power systems [7, 25], flight formation [13]).

Decomposition methods represent a very powerful tool for solving distributed MPC problems in network systems. The basic idea of these methods is to decompose the original large optimization problem into smaller subproblems. Decomposition methods can be divided in two main classes: primal and dual decomposition methods. In primal decomposition the optimization problem is solved using the original formulation and variables via methods such as interior-point, feasible directions, Gauss-Jacobi type and others [3, 5, 6, 23, 25]. In dual decomposition the original problem is rewritten using Lagrangian relaxation for the coupling constraints and the dual problem is solved with a Newton or (sub)gradient algorithm [1, 2, 4, 16, 15]. In [23, 25] cooperative based distributed MPC algorithms are proposed based on Gauss-Jacobi iterations, where asymptotic convergence to the centralized solution and feasibility for their iterates is proved. In [5, 6] non-cooperative algorithms are derived for distributed MPC problems, where communication takes place only between neighbors. In [3] a distributed algorithm based on interior-point methods is proposed whose iterates converge to the centralized solution. In [1, 4, 16, 15] dual distributed gradient algorithms based on Lagrange relaxation of the coupling constraints are presented for solving MPC problems, algorithms which usually produce feasible and optimal primal solutions in the limit. While much research has focused on a dual approach, our work develops a primal method that ensures constraint feasibility, has low iteration complexity and provides estimates on suboptimality.

Further, MPC schemes tend to be quite costly computation-wise compared with classical control methods, e.g. PID controllers, so that for these advanced schemes we need hardware with a reasonable amount of computational power that is embedded on the subsystems. Therefore, research for distributed and embedded MPC has gained momentum in the past few years. The concept behind embedded MPC is designing a control scheme that can be implemented on autonomous electronic hardware, e.g programmable logic controllers (PLC) [24] or field-programmable gate arrays (FPGAs) [10]. Such devices vary widely in both computational power and memory storage capabilities as well as cost. As a result, there has been a growing focus on making MPC schemes faster by reducing problem size and improving the computational efficiency through decentralization [21], moving block strategies (e.g. by using latent variables [8] or Laguerre functions [27]) and other procedures, allowing these schemes to be implemented on cheaper hardware with little computational power.

The main contribution of this paper is the development of a parallel coordinate descent algorithm for smooth convex optimization problems with separable constraints that is computationally efficient and thus suitable for MPC schemes that need to be implemented distributively or in hardware with limited computational power. This algorithm employs parallel block-coordinate updates for the optimization variables and has similarities to the optimization algorithm proposed in [23], but with simpler implementation, lower iteration complexity and guaranteed rate of convergence. We derive (sub)linear rate of convergence for the new algorithm whose proof relies on the Lipschitz property of the gradient of the objective function. The new parallel algorithm is used for solving MPC problems for general linear network systems in a distributed fashion using local information. For ensuring stability of the MPC scheme, we use a terminal cost formulation derived from a distributed synthesis and we eliminate the need for a terminal state constraint. Compared with the existing approaches based on an end point constraint, we reduce the conservatism by combining the underlying structure of the system with distributed optimization [9, 12, 19]. Because the MPC optimization problem is usually terminated before convergence, our

MPC controller is a form of suboptimal control. However, using the theory of suboptimal control [22] we can still guarantee feasibility and stability.

This paper is organized as follows. In Section 2 we derive our parallel coordinate descent optimization algorithm and prove the convergence rate for it. In Sections 3.1-3.2 we introduce the model for general network systems, present the MPC problem with a terminal cost formulation and provide the means for which this terminal cost can be synthesized distributively. In Sections 3.3-3.4 we employ our algorithm for distributively solving MPC problems arising from network systems and discuss details regarding its implementation. In Section 4 we compare its performance with other algorithms and test it on a real application - a quadruple water tank process.

2. A parallel coordinate descent algorithm for smooth convex problems with separable constraints

We work in $\mathbb{R}^n$ composed by column vectors. For $u, v \in \mathbb{R}^n$ we denote the standard Euclidean inner product $\langle u, v\rangle = u^T v$, the Euclidean norm $\|u\| = \sqrt{\langle u, u\rangle}$ and $\|x\|_P^2 = x^T P x$. Further, for a symmetric matrix P, we use $P \succ 0$ ($P \succeq 0$) for a positive (semi)definite matrix. For matrices P and Q, we use $\mathrm{diag}(P, Q)$ to denote the block diagonal matrix formed by these two matrices.

In this section we propose a parallel coordinate descent based algorithm for efficiently solving the general convex optimization problem of the following form:

$$f^* = \min_{\mathbf{u}^1 \in \mathbf{U}^1, \cdots, \mathbf{u}^M \in \mathbf{U}^M} f(\mathbf{u}^1, \ldots, \mathbf{u}^M), \tag{1}$$

where $\mathbf{u}^i \in \mathbb{R}^{n_{\mathbf{u}}^i}$ with $i = 1, \ldots, M$, are the decision variables, constrained to individual convex sets $\mathbf{U}^i \subset \mathbb{R}^{n_{\mathbf{u}}^i}$. We gather the individual constraint sets $\mathbf{U}^i$ into the set $\mathbf{U} = \mathbf{U}^1 \times \cdots \times \mathbf{U}^M$, and denote the entire decision variable for (1) by $\mathbf{u} = \left[(\mathbf{u}^1)^T \ldots (\mathbf{u}^M)^T\right]^T \in \mathbb{R}^{n_{\mathbf{u}}}$, with $n_{\mathbf{u}} = \sum_{i=1}^M n_{\mathbf{u}}^i$. As we will show in this section, the new algorithm can be used on many parallel computing architectures, has low computational cost per iteration and guaranteed convergence rate. We will then apply this algorithm for solving distributed MPC problems arising in network systems in Section 3.

2.1. Parallel Block-Coordinate Descent Method

Let us partition the identity matrix in accordance with the structure of the decision variable $\mathbf{u}$:

$$I_{n_{\mathbf{u}}} = \left[(E^1)^T \dots (E^M)^T\right]^T \in \mathbb{R}^{n_{\mathbf{u}} \times n_{\mathbf{u}}},$$

where $E^i \in \mathbb{R}^{n_{\mathbf{u}} \times n^i_{\mathbf{u}}}$ for all $i = 1, \cdots, M$. With matrices E^i we can represent $\mathbf{u} = \sum_{i=1}^{M} E^i \mathbf{u}^i$. We also define the partial gradient $\nabla_i f(\mathbf{u}) \in \mathbb{R}^{n^i_{\mathbf{u}}}$ of $f(\mathbf{u})$ as: $\nabla_i f(\mathbf{u}) = (E^i)^T \nabla f(\mathbf{u})$. We assume that the gradient of f is coordinate-wise Lipschitz continuous with constants $L_i > 0$, i.e:

$$\left\|\nabla_i f(\mathbf{u} + E^i h_i) - \nabla_i f(\mathbf{u})\right\| \leq L_i \left\|h_i\right\| \quad \forall \mathbf{u} \in \mathbb{R}^{n_{\mathbf{u}}},\ h_i \in \mathbb{R}^{n^i_{\mathbf{u}}}. \tag{2}$$

Due to the assumption that f is coordinate-wise Lipschitz continuous, it can be easily deduced that [17]:

$$f(\mathbf{u} + E^i h_i) \leq f(\mathbf{u}) + \langle \nabla_i f(\mathbf{u}), h_i \rangle + \frac{L_i}{2} \left\|h_i\right\|^2 \quad \forall \mathbf{u} \in \mathbb{R}^{n_{\mathbf{u}}},\ h_i \in \mathbb{R}^{n^i_{\mathbf{u}}}. \tag{3}$$

We now introduce the following norm for the extended space $\mathbb{R}^{n_{\mathbf{u}}}$:

$$\left\|\mathbf{u}\right\|_1^2 = \sum_{i=1}^{M} L_i \left\|\mathbf{u}^i\right\|^2, \tag{4}$$

which will prove useful for estimating the rate of convergence for our algorithm. Additionally, if function f is smooth and strongly convex with regards to $\|\cdot\|_1$ with a parameter σ_1, then [18]:

$$f(\mathbf{w}) \geq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle + \frac{\sigma_1}{2} \left\|\mathbf{w} - \mathbf{v}\right\|_1^2 \ \forall \mathbf{w}, \mathbf{v} \in \mathbb{R}^{n_{\mathbf{u}}}. \tag{5}$$

Note that if f is strongly convex w.r.t the standard Euclidean norm $\|\cdot\|$ with a parameter σ_0, then $\sigma_0 \geq \sigma_1 L^i_{\max}$, where $L^i_{\max} = \max_i L_i$. By taking $\mathbf{w} = \mathbf{v} + E^i h_i$ and $\mathbf{v} = \mathbf{u}$ in (5) we also get:

$$f(\mathbf{u} + E^i h_i) \geq f(\mathbf{u}) + \langle \nabla_i f(\mathbf{u}), h_i \rangle + \frac{\sigma_1 L_i}{2} \left\|h_i\right\|^2 \quad \forall \mathbf{u} \in \mathbb{R}^{n_{\mathbf{u}}}, h_i \in \mathbb{R}^{n^i_{\mathbf{u}}},$$

and combining with (3) we also deduce that $\sigma_1 \leq 1$.

We now define the constrained coordinate update for our algorithm:

$$\bar{\mathbf{v}}^i(\mathbf{u}) = \arg\min_{\mathbf{v}^i \in \mathbf{U}^i} \left\langle \nabla_i f(\mathbf{u}), \mathbf{v}^i - \mathbf{u}^i \right\rangle + \frac{L_i}{2} \left\| \mathbf{v}^i - \mathbf{u}^i \right\|^2$$

$$\bar{\mathbf{u}}^i(\mathbf{u}) = \mathbf{u} + E^i(\mathbf{v}^i(\mathbf{u}) - \mathbf{u}^i),\ i = 1, \ldots, M.$$

The optimality conditions for the previous optimization problem are:

$$\left\langle \nabla_i f(\mathbf{u}) + L_i(\bar{\mathbf{v}}^i(\mathbf{u}) - \mathbf{u}^i), \mathbf{v}^i - \bar{\mathbf{v}}^i(\mathbf{u}) \right\rangle \geq 0 \ \ \forall \mathbf{v}^i \in \mathbf{U}^i. \tag{6}$$

Taking $\mathbf{v}^i = \mathbf{u}^i$ in the previous inequality and combining with (3) we obtain the following decrease in the objective function:

$$f(\mathbf{u}) - f(\bar{\mathbf{u}}^i(\mathbf{u})) \geq \frac{L_i}{2} \left\| \bar{\mathbf{v}}^i(\mathbf{u}) - \mathbf{u}^i \right\|^2. \tag{7}$$

We now present our *Parallel Coordinate Descent Method*, that resembles the method in [23] but with simpler implementation, lower iteration complexity and guaranteed rate of convergence, and is a parallel version of the coordinate descent method from [17]:

Algorithm PCDM

Choose $\mathbf{u}_0^i \in \mathbf{U}^i$ for all $i = 1, \ldots, M$. For $k \geq 0$:

1. **Compute in parallel** $\bar{\mathbf{v}}^i(\mathbf{u}_k),\ i = 1, \ldots, M.$
2. **Update in parallel:**

$$\mathbf{u}_{k+1}^i = \frac{1}{M} \bar{\mathbf{v}}^i(\mathbf{u}_k) + \frac{M-1}{M} \mathbf{u}_k^i,\ i = 1, \ldots, M.$$

Note that if the sets $\mathbf{U}^i$ are simple (by simple we understand that the projection on these sets is easy), then computing $\bar{\mathbf{v}}^i(\mathbf{u})$ consists of projecting a vector on these sets and can be done numerically very efficient. For example, if these sets are simple box sets, i.e $\mathbf{U}^i = \left\{ \mathbf{u}^i \in \mathbb{R}^{n_{\mathbf{u}}^i} | \mathbf{u}_{min}^i \leq \mathbf{u}^i \leq \mathbf{u}_{max}^i \right\}$, then the complexity of computing $\bar{\mathbf{v}}^i(\mathbf{u})$, once $\nabla_i f(\mathbf{u})$ is available, is $\mathcal{O}(n_{\mathbf{u}}^i)$. In turn, computing $\nabla_i f(\mathbf{u})$ has, in the worst case, complexity $\mathcal{O}(n_{\mathbf{u}}^i n_{\mathbf{u}})$ for

quadratic dense functions. Thus, Algorithm PCDM has usually a very low iteration cost per subsystem compared to other existing methods, e.g. Jacobi type algorithm presented in [23], which usually require numerical complexity at least $\mathcal{O}((n^i_{\mathbf{u}})^3 + n^i_{\mathbf{u}} n_{\mathbf{u}})$ per iteration for each subsystem i, provided that the local quadratic problems are solved with an interior point solver.

From (6)-(7), convexity of f and $\mathbf{u}_{k+1} = \sum_i \frac{1}{M} \bar{\mathbf{u}}^i(\mathbf{u}_k)$ we see immediately that method PCDM decreases strictly the objective function at each iteration, provided that $\mathbf{u}_k \neq \mathbf{u}_*$, where $\mathbf{u}_*$ is the optimal solution of (1), i.e.:

$$f(\mathbf{u}_{k+1}) < f(\mathbf{u}_k) \;\; \forall k \geq 0, \; \mathbf{u}_k \neq \mathbf{u}_*. \tag{8}$$

Let f^* be the optimal value in optimization problem (1). The following theorem derives convergence rate of Algorithm PCDM and employs standard techniques for proving rate of convergence of the gradient method [17, 18]:

Theorem 1. *If function f in optimization problem* (1) *has a coordinate-wise Lipschitz continuous gradient with constants L_i as given in* (2)*, then Algorithm PCDM has the following sublinear rate of convergence:*

$$f(\mathbf{u}_k) - f^* \leq \frac{M}{M+k} \left(\frac{1}{2} r_0^2 + f(\mathbf{u}_0) - f^* \right),$$

where $r_0 = \|\mathbf{u}_0 - \mathbf{u}_*\|_1$.

Proof. We introduce the following term:

$$r_k^2 = \|\mathbf{u}_k - \mathbf{u}_*\|_1^2 = \sum_{i=1}^{M} L_i \left\langle \mathbf{u}_k^i - \mathbf{u}_*^i, \mathbf{u}_k^i - \mathbf{u}_*^i \right\rangle,$$

where $\mathbf{u}_*$ is the optimal solution of (1) and $\mathbf{u}_*^i = (E^i)^T \mathbf{u}_*$. Then, using similar derivations as in [17], we have:

$$\begin{aligned}
r_{k+1}^2 &= \sum_{i=1}^{M} L_i \left\| \frac{1}{M} \bar{\mathbf{v}}^i(\mathbf{u}_k) + (1 - \frac{1}{M}) \mathbf{u}_k^i - \mathbf{u}_*^i \right\|^2 \\
&\overset{(6)}{\leq} r_k^2 + \sum_{i=1}^{M} \frac{L_i}{M} (\frac{1}{M} - 2) \left\| \bar{\mathbf{v}}^i(\mathbf{u}_k) - \mathbf{u}_k^i \right\|^2 + \frac{2}{M} \left\langle \nabla_i f(\mathbf{u}_k), \mathbf{u}_*^i - \bar{\mathbf{v}}^i(\mathbf{u}_k) \right\rangle \\
&\overset{\frac{1}{M} \leq 1}{\leq} r_k^2 - \frac{2}{M} \sum_{i=1}^{M} (\frac{L_i}{2} \left\| \bar{\mathbf{v}}^i(\mathbf{u}_k) - \mathbf{u}_k^i \right\|^2 + \\
&\qquad \left\langle \nabla_i f(\mathbf{u}_k), \bar{\mathbf{v}}^i(\mathbf{u}_k) - \mathbf{u}_k^i \right\rangle + \left\langle \nabla_i f(\mathbf{u}_k), \mathbf{u}_*^i - \mathbf{u}_k^i \right\rangle).
\end{aligned}$$

By convexity of f and (3) we obtain:

$$r_{k+1}^2 \leq r_k^2 - 2(f(\mathbf{u}_{k+1}) - f(\mathbf{u}_k)) + \frac{2}{M}\langle \nabla f(\mathbf{u}_k), \mathbf{u}_* - \mathbf{u}_k \rangle \tag{9}$$

and adding up these inequalities we get:

$$\begin{aligned}\frac{1}{2}r_0^2 + f(\mathbf{u}_0) - f^* &\geq \frac{1}{2}r_{k+1}^2 + f(\mathbf{u}_{k+1}) - f^* + \frac{1}{M}\sum_{j=0}^{k}(f(\mathbf{u}_j) - f^*) \\ &\geq f(\mathbf{u}_{k+1}) - f^* + \frac{1}{M}\sum_{j=0}^{k}(f(\mathbf{u}_j) - f^*).\end{aligned}$$

Taking into account that our algorithm is a descent algorithm, i.e. $f(\mathbf{u}_j) \geq f(\mathbf{u}_{k+1})$ for all $j \leq k$ and by the previous inequality the proof is complete. □

Now, we derive linear convergence rate for Algorithm PCDM, provided that f is additionally strongly convex:

Theorem 2. *Under the assumptions of Theorem 1 and if we further assume that f is strongly convex with regards to $\|\cdot\|_1$ with a constant σ_1 as given in* (5)*, then the following linear rate of convergence is achieved for Algorithm PCDM:*

$$f(\mathbf{u}_k) - f^* \leq \left(1 - \frac{2\sigma_1}{M(1+\sigma_1)}\right)^k \left(\frac{1}{2}r_0^2 + f(\mathbf{u}_0) - f^*\right).$$

Proof. We take $\mathbf{w} = \mathbf{u}^*$ and $\mathbf{v} = \mathbf{u}_k$ in (5) and through (9) we get:

$$\frac{1}{2}r_{k+1}^2 + f(\mathbf{u}_{k+1}) - f^* \leq \frac{1}{2}r_k^2 + f(\mathbf{u}_k) - f^* - \frac{1}{M}(f(\mathbf{u}_k) - f^* + \frac{\sigma_1}{2}r_k^2). \tag{10}$$

From the strong convexity of f in (5) we also get:

$$f(\mathbf{u}_k) - f^* + \frac{\sigma_1}{2}r_k^2 \geq \sigma_1 r_k^2.$$

We now define $\gamma = \frac{2\sigma_1}{1+\sigma_1} \in [0,1]$ and using the previous inequality we obtain the following result:

$$f(\mathbf{u}_k) - f^* + \frac{\sigma_1}{2}r_k^2 \geq \gamma\left(f(\mathbf{u}_k) - f^* + \frac{\sigma_1}{2}r_k^2\right) + (1-\gamma)\sigma_1 r_k^2.$$

Using this inequality in (10) we get:

$$\frac{1}{2}r_{k+1}^2 + f(\mathbf{u}_{k+1}) - f^* \leq \left(1 - \frac{\gamma}{M}\right)\left(\frac{1}{2}r_k^2 + f(\mathbf{u}_k) - f^*\right).$$

Applying this inequality iteratively, we obtain the following for $k \geq 0$:

$$\frac{1}{2}r_k^2 + f(\mathbf{u}_k) - f^* \leq \left(1 - \frac{\gamma}{M}\right)^k \left(\frac{1}{2}r_0^2 + f(\mathbf{u}_0) - f^*\right),$$

and by replacing $\gamma = \frac{2\sigma_1}{1+\sigma_1}$ we complete the proof. □

The following properties follow immediately for our Algorithm PCDM.

Lemma 1. *For the optimization problem* (1)*, with the assumptions of Theorem 2, we have the following statements:*
(i) Given any feasible initial guess $\mathbf{u}_0$*, the iterates of the Algorithm PCDM are feasible at each iteration, i.e.* $\mathbf{u}_k^i \in \mathbf{U}^i$ *for all* $k \geq 0$*.*
(ii) The function f *is nonincreasing, i.e.* $f(\mathbf{u}_{k+1}) \leq f(\mathbf{u}_k)$ *according to* (8)*.*
(iii) The sub(linear) rate of convergence of Algorithm PCDM is given in Theorem 1 (Theorem 2).

3. Application of Algorithm PCDM to distributed suboptimal MPC

The Algorithm PCDM can be used to solve distributively input constrained MPC problems for network systems after state elimination. In this section we show that the MPC scheme obtained by solving approximately the corresponding optimization problem with Algorithm PCDM is stable and distributed.

3.1. MPC for network systems with terminal cost and without end constraints

In this paper we consider discrete-time network systems, which are usually modeled by a graph whose nodes represent subsystems and whose arcs indicate dynamic couplings, defined by the following linear state equations [3, 4, 21]:

$$x_{t+1}^i = \sum_{j \in \mathcal{N}^i} A^{ij} x_t^j + B^{ij} u_t^j, \qquad i = 1, \cdots, M, \tag{11}$$

where M denotes the number of interconnected subsystems, $x_t^j \in \mathbb{R}^{n_j}$ and $u_t^j \in \mathbb{R}^{m_j}$ represent the state and respectively the input of jth subsystem at time t, $A^{ij} \in \mathbb{R}^{n_i \times n_j}$, $B^{ij} \in \mathbb{R}^{n_i \times m_j}$ and $\mathcal{N}^i$ is the set of indices which contains

the index i and that of its neighboring subsystems. A particular case of (11), that is frequently found in literature [16, 23, 25], has the following dynamics:

$$x_{t+1}^i = A^{ii} x_t^i + \sum_{j \in \mathcal{N}^i} B^{ij} u_t^j. \tag{12}$$

For stability analysis, we also express the dynamics of the entire system: $x_{t+1} = Ax_t + Bu_t$, where $n = \sum_{i=1}^{M} n_i$, $m = \sum_{i=1}^{M} m_i$, $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$ and $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$. For system (11) or (12) we consider local input constraints:

$$u_t^i \in U^i \quad i = 1, \cdots, M, \;\; t \geq 0, \tag{13}$$

with $U^i \subseteq \mathbb{R}^{m_i}$ compact, convex sets with the origin in their interior. We also consider convex local stage and terminal costs for each subsystem i: $\ell^i(x^i, u^i)$ and $\ell_{\mathrm{f}}^i(x^i)$. Let us denote the input trajectory for subsystem i and the overall input trajectory for the entire system by:

$$\mathbf{u}^i = [(u_0^i)^T \cdots (u_{N-1}^i)^T]^T, \; \mathbf{u} = [(\mathbf{u}^1)^T \cdots (\mathbf{u}^M)^T]^T.$$

We can now formulate the MPC problem for system (11) over a prediction horizon of length N and a given initial state x as [20]:

$$\begin{aligned} V_N^*(x) = \min_{u_t^i \in U^i \; \forall i,t} V_N(x, \mathbf{u}) \quad & \left(:= \sum_{i=1}^{M} \sum_{t=0}^{N-1} \ell^i(x_t^i, u_t^i) + \ell_{\mathrm{f}}^i(x_N^i) \right) \\ \text{s.t: } x_{t+1}^i = & \sum_{j \in \mathcal{N}^i} A^{ij} x_t^j + B^{ij} u_t^j, \;\; x_0^i = x^i, \; i = 1, \cdots, M, \;\; t \geq 0. \end{aligned} \tag{14}$$

It is well-known that by eliminating the states using dynamics (11), the MPC problem (14) can be recast [20] as a convex optimization problem of type (1), where $n_{\mathbf{u}}^i = Nm_i$, the function f is convex (recall that we assume the stage and final costs $\ell^i(\cdot)$ and $\ell_{\mathrm{f}}^i(\cdot)$ to be convex), whilst the convex sets $\mathbf{U}^i$ are the Cartesian product of the convex sets U^i for N times. We denote the approximate solution produced by Algorithm PCDM for problem (14) after certain number of iterations with $\mathbf{u}^{\mathrm{CD}}$. We also consider that at each MPC step the Algorithm PCDM is initialized (warm start) with the shifted sequence of controllers obtained at the previous step and the feedback controller $\kappa(\cdot)$

computed in Section 3.2 below. The suboptimal MPC scheme corresponding to (14) would now be:

Suboptimal MPC scheme

Given initial state x and initial $\tilde{\mathbf{u}}^{\mathrm{CD}}$ repeat:

1. **Recast MPC problem** (14) **as opt. problem** (1)
2. **Solve** (1) **approximately with Alg. PCDM starting from $\tilde{\mathbf{u}}^{\mathbf{CD}}$ and obtain $\mathbf{u}^{\mathbf{CD}}$**
3. **Update x. Update $\tilde{\mathbf{u}}^{\mathbf{CD}}$ using warm start.**

3.2. Distributed synthesis for a terminal cost

We assume that stability of the MPC scheme (14) is enforced by adapting the terminal cost $\ell_{\mathrm{f}}(\cdot) = \sum_{i=1}^{M} \ell_{\mathrm{f}}^i(\cdot)$ and the horizon length N appropriately such that sufficient stability criteria are fulfilled [9, 12, 19]. Usually, stability of MPC with quadratic stage cost $\ell^i(x^i, u^i) = \left\|x^i\right\|_{Q^i}^2 + \left\|u^i\right\|_{R^i}^2$, where the matrices $Q^i \succeq 0$ and $R^i \succ 0$, and without terminal constraint is enforced if the following criteria holds: there exists a neighborhood of the origin $\Omega \subseteq \mathbb{R}^n$, a stabilizing feedback law $\kappa(\cdot)$ and a terminal cost $\ell_{\mathrm{f}}(\cdot)$ such that the following relations hold:

$$\begin{cases} \ell_{\mathrm{f}}(Ax + B\kappa(x)) - \ell_{\mathrm{f}}(x) + \kappa(x)^T R\kappa(x) + x^T Qx \quad \leq 0 \ \ \forall x \in \Omega, \\ \kappa(x) \in \mathbf{U}, \ Ax + B\kappa(x) \in \Omega, \end{cases} \tag{15}$$

where the matrices Q and R have a block diagonal structure and are composed of the blocks Q^i and R^i, respectively. As shown in [9, 12, 19], MPC schemes based on the condition (15) are usually less conservative than schemes based on end point constraint. Keeping in line with the distributed nature of our system, the control law $\kappa(\cdot)$ and the final stage cost $\ell_{\mathrm{f}}(\cdot)$ need to be computed locally. In this section we develop a distributed synthesis procedure to construct them locally. We choose the terminal cost for each subsystem i to be quadratic: $\ell_{\mathrm{f}}^i(x_N^i) = \left\|x_N^i\right\|_{P^i}^2$, where $P^i \succ 0$. For a locally computed $\kappa(\cdot)$, we employ distributed control laws: $u^i = F^i x^i$, i.e. $\kappa(\cdot)$ is taken linear with a block-diagonal structure. Centralized LMI formulations of (15) for quadratic terminal

costs are well-known in the literature [20]. However, our goal is to solve (15) distributively. To this purpose, we first need to introduce vectors $x^{\mathcal{N}^i} \in \mathbb{R}^{n_{\mathcal{N}^i}}$ and $u^{\mathcal{N}^i} \in \mathbb{R}^{m_{\mathcal{N}^i}}$ for subsystem i, where $n_{\mathcal{N}^i} = \sum_{j \in \mathcal{N}^i} n_j$ and $m_{\mathcal{N}^i} = \sum_{j \in \mathcal{N}^i} m_j$. These vectors are comprised of the state and input vectors of subsystem i and those of its neighbors: $x^{\mathcal{N}^i} = \left[(x^j)^T, j \in \mathcal{N}^i\right]^T$, $u^{\mathcal{N}^i} = \left[(u^j)^T, j \in \mathcal{N}^i\right]^T$.

Since our synthesis procedure needs to be distributed and taking into account that $\ell_{\mathrm{f}}(\cdot) = \sum_{i=1}^{M} \ell_{\mathrm{f}}^i(\cdot)$, we impose the following distributed structure to ensure (15) (see also [11] for a similar approach where infinity-norm control Lyapunov functions are synthesized in a decentralized fashion by solving linear programs for each subsystem) for $i = 1, \cdots, M$:

$$\ell_{\mathrm{f}}^i((x^i)^+) - \ell_{\mathrm{f}}^i((x^i)) + (F^i x^i)^T R^i F^i x^i + (x^i)^T Q^i x^i \leq q^i(x^{\mathcal{N}^i}) \; \forall x^{\mathcal{N}^i} \in \mathbb{R}^{n_{\mathcal{N}^i}} \quad (16)$$

such that: $q(x) = \sum_{i=1}^{M} q^i(x^{\mathcal{N}^i}) \leq 0$. We assume that $q^i(x^{\mathcal{N}^i})$ also have a quadratic form, with $q^i(x^{\mathcal{N}^i}) = \left\| x^{\mathcal{N}^i} \right\|^2_{W^{\mathcal{N}^i}}$, where $W^{\mathcal{N}^i} \in \mathbb{R}^{n_{\mathcal{N}^i} \times n_{\mathcal{N}^i}}$. Being a sum of quadratic functions, $q(x)$ can itself be expressed as a quadratic function, $q(x) = \|x\|^2_W$, where $W \in \mathbb{R}^{n \times n}$ is formed from the appropriate block components of matrices $W^{\mathcal{N}^i}$. Note that we do not require that matrices $W^{\mathcal{N}^i}$ be negative semidefinite. On the contrary, positive or indefinite matrices allow local terminal costs to increase so long as the global cost still decreases. This approach reduces the conservatism in deriving the matrices P^i and F^i. For obtaining P^i and F^i, we introduce matrices $E_n^i \in \mathbb{R}^{n_i \times n}$, $E_m^i \in \mathbb{R}^{m_i \times m}$, $J_n^{\mathcal{N}^i} \in \mathbb{R}^{n_{\mathcal{N}^i} \times n}$, $J_m^{\mathcal{N}^i} \in \mathbb{R}^{m_{\mathcal{N}^i} \times m}$ such that $x^i = E_n^i x$, $u^i = E_m^i u$, $x^{\mathcal{N}^i} = J_n^{\mathcal{N}^i} x$ and $u^{\mathcal{N}^i} = J_m^{\mathcal{N}^i} u$. We now define the matrices $A^{\mathcal{N}^i} = E_n^i A (J_n^{\mathcal{N}^i})^T$, $B^{\mathcal{N}^i} = E_n^i B (J_m^{\mathcal{N}^i})^T$ and $F^{\mathcal{N}^i} = J_m^{\mathcal{N}^i} F (J_n^{\mathcal{N}^i})^T$, as to express the dynamics (11) for subsystem i: $x_{t+1}^i = (A^{\mathcal{N}^i} + B^{\mathcal{N}^i} F^{\mathcal{N}^i}) x_t^{\mathcal{N}^i}$. Using these notations we can now recast inequality (16) as:

$$\begin{aligned} &(A^{\mathcal{N}^i} + B^{\mathcal{N}^i} F^{\mathcal{N}^i})^T P^i (A^{\mathcal{N}^i} + B^{\mathcal{N}^i} F^{\mathcal{N}^i}) \\ &- J_n^{\mathcal{N}^i} (E_n^i)^T (P^i + Q^i + (F^i)^T R^i F^i) E_n^i (J_n^{\mathcal{N}^i})^T \preceq W^{\mathcal{N}^i}. \end{aligned} \quad (17)$$

The task of finding suitable P^i, F^i and $W^{\mathcal{N}^i}$ matrices is now reduced to the

following optimization problem:

$$\min_{P^i,F^i,W^{\mathcal{N}^i},\delta}\{\delta:\ \text{MI (17)},\ i=1,\cdots,M,\quad W\preceq\delta I\}. \tag{18}$$

It can be easily observed that if the optimal value $\delta^*\leq 0$, consequently $W\leq 0$ and (15) holds. This optimization problem, in its current nonconvex form, cannot be solved efficiently. However, it can be recast as a sparse SDP if we can reformulate (17) as an LMI. We need now to make the assumption that all the subsystems have the same dimension for the states, i.e. $n_i=n_j$ for all i,j. Subsequently, we introduce the well-known linearizations: $P^i=(S^i)^{-1}$, $F^i=Y^iG^{-1}$ and a series of matrices that will be of aid in formulating the LMIs:

$$\begin{aligned}
&G^{\mathcal{N}^i}=I_{|\mathcal{N}^i|}\otimes G,\ \ G^{\mathcal{N}^i\setminus i}=[0\ \ I_{|\mathcal{N}^i|-1}\otimes G],\ S^{\mathcal{N}^i}=\operatorname{diag}(S^i,\mu_iI_{(n_{\mathcal{N}^i}-n_i)})\\
&Y^{i,j}=F^jG,\ j\in\mathcal{N}^i\setminus i,\ \ Y^{\mathcal{N}^i}=\operatorname{diag}(Y^i,Y^{i,j})=F^{\mathcal{N}^i}G^{\mathcal{N}^i},\\
&T^{\mathcal{N}^i}=\begin{bmatrix}A^{\mathcal{N}^i}G^{\mathcal{N}^i}+B^{\mathcal{N}^i}Y^{\mathcal{N}^i}\\ G^{\mathcal{N}^i\setminus i}\end{bmatrix},\ T^i=\begin{bmatrix}(Q^i)^{\frac{1}{2}}G & 0\\ (R^i)^{\frac{1}{2}}Y^i & 0\end{bmatrix},
\end{aligned}$$

where the 0 blocks are of appropriate dimensions[1].

Lemma 2. *If the following SDP:*

$$\min_{G,S^i,Y^i,Y^{i,j},\tilde{W},\mu^i,\delta}\ \delta \tag{19}$$

$$\begin{aligned}
s.t:\quad &\begin{bmatrix}G^{\mathcal{N}^i}+(G^{\mathcal{N}^i})^T-S^{\mathcal{N}^i}+\tilde{W}^{\mathcal{N}^i} & * & *\\ T^{\mathcal{N}^i} & S^{\mathcal{N}^i} & *\\ T^i & 0 & I\end{bmatrix}\succ 0 \qquad (20)\\
&Y^{i,j}=Y^j\ \ \forall j\in\mathcal{N}^i,\ \ i=1,\cdots,M,\quad \tilde{W}\preceq\delta I,
\end{aligned}$$

has an optimal value $\delta^*\leq 0$, then (15) *holds*[2].

[1] By I_n we denote the identity matrix of size $n\times n$, by $\otimes$ we denote the standard Kronecker product and by $|\mathcal{N}^i|$ the cardinality of the set $\mathcal{N}^i$.

[2] By $*$ we denote the transpose of the symmetric block of the matrix.

PROOF. From (20) we observe that $S^{\mathcal{N}^i} \succ 0$, so that $(S^{\mathcal{N}^i} - G^{\mathcal{N}^i})^T (S^{\mathcal{N}^i})^{-1} (S^{\mathcal{N}^i} - G^{\mathcal{N}^i}) \succeq 0$, which in turn implies

$$G^{\mathcal{N}^i} + (G^{\mathcal{N}^i})^T - S^{\mathcal{N}^i} \preceq (G^{\mathcal{N}^i})^T (S^{\mathcal{N}^i})^{-1} G^{\mathcal{N}^i}. \tag{21}$$

If we apply the Schur complement to (20), we obtain:

$$0 \preceq G^{\mathcal{N}^i} + (G^{\mathcal{N}^i})^T - S^{\mathcal{N}^i} + \tilde{W}^{\mathcal{N}^i} - (T^{\mathcal{N}^i})^T (S^{\mathcal{N}^i})^{-1} T^{\mathcal{N}^i} - (T^i)^T T^i$$

and by (21) we get $(G^{\mathcal{N}^i})^{-T} \left[(T^{\mathcal{N}^i})^T (S^{\mathcal{N}^i})^{-1} T^{\mathcal{N}^i} + (T^i)^T T^i \right] (G^{\mathcal{N}^i})^{-1} - (S^{\mathcal{N}^i})^{-1} \preceq (G^{\mathcal{N}^i})^{-T} \tilde{W}^{\mathcal{N}^i} (G^{\mathcal{N}^i})^{-1}$, which is equivalent to (17) if we consider $W^{\mathcal{N}^i} = (G^{\mathcal{N}^i})^{-T} \tilde{W}^{\mathcal{N}^i} (G^{\mathcal{N}^i})^{-1}$. □

There exist in literature many optimization algorithms (see e.g. [14]) for solving distributively sparse SDP problems in the form (19).

3.3. Stability of the MPC scheme

We will consider the cost function of the MPC problem $V_N(x, \mathbf{u}^{\mathrm{CD}})$ as a Lyapunov function, using the standard theory for suboptimal control (see e.g. [12, 19, 20, 22, 23] for similar approaches). We also consider that at each MPC step the Algorithm PCDM is initialized (warm start) with the shifted sequence of controllers obtained at the previous step and the feedback controller $\kappa(\cdot)$ computed in Section 3.2 such that (15) is satisfied and we denote it by $(\tilde{\mathbf{u}}^{\mathrm{CD}})^+$. The next state is also denoted with x^+. Assume also that $\kappa(\cdot)$, $\ell_f(\cdot)$ and $\alpha > 0$ are chosen such that, together with the set

$$\Omega = \{x \in \mathbb{R}^n : \ell_f(x) \leq \alpha\},$$

satisfies (15). Then, using Theorem 3 from [12] we have that our MPC controller stabilizes asymptotically the system for all initial states $x \in X_N$, where

$$X_N = \{x \in \mathbb{R}^n : V_N^*(x) \leq Nd + \alpha\},$$

such that $V_N(x, \mathbf{u}^{CD}) \leq Nd + \alpha$, where $d > 0$ is a parameter for which we have $\ell(x, \mathbf{u}) \geq d$ for all $x \notin \Omega$. Clearly, this MPC scheme is locally stable with a region of attraction X_N.

3.4. Distributed implementation of the MPC scheme based on Algorithm PCDM

In this section we discuss some technical aspects for the distributed implementation of the MPC scheme derived above when using Algorithm PCDM to solve the control problem (14). Usually, in the linear MPC framework, the local stage and final cost are taken of the following quadratic form:

$$\ell^i(x^i, u^i) = \left\|x^i\right\|^2_{Q^i} + \left\|u^i\right\|^2_{R^i}, \quad \ell^i_{\mathrm{f}}(x^i) = \left\|x^i\right\|^2_{P^i},$$

where the matrices $Q^i, P^i \in \mathbb{R}^{n_i \times n_i}$ are positive semidefinite, whilst matrices $R_i \in \mathbb{R}^{m_i \times m_i}$ are positive definite. We also assume that the local constraints sets U^i are polyhedral. In this particular case, the objective function in (14), after eliminating the dynamics, is quadratically strongly convex, having the form [20]:

$$f(\mathbf{u}) = 0.5\ \mathbf{u}^T\mathbf{Q}\mathbf{u} + (\mathbf{W}x + \mathbf{w})^T\mathbf{u},$$

where $\mathbf{Q}$ is positive definite due to the assumption that all R^i are positive definite. Usually, for the dynamics (11) the corresponding matrices $\mathbf{Q}$ and $\mathbf{W}$ obtained after eliminating the states are dense and despite the fact that Algorithm PCDM can perform parallel computations (i.e. each subsystem needs to solve small local problems) we need all to all communication between subsystems. However, for the dynamics (12) the corresponding matrices $\mathbf{Q}$ and $\mathbf{W}$ are sparse and in this case in our Algorithm PCDM we can perform distributed computations (i.e. the subsystems solve small local problems in parallel and they need to communicate only with their neighborhood subsystems as detailed below). Indeed, if the dynamics of the system are given by (12), then

$$x^i_{t+1} = (A^{ii})^t x^i_t + \sum_{l=1}^{t} \sum_{j \in \mathcal{N}^i} (A^{ii})^{l-1} B^{ij} u^j_{t-l}$$

and thus the matrices $\mathbf{Q}$ and $\mathbf{W}$ have a sparse structure (see also [3]). Let us define the *neighborhood subsystems* of a certain subsystem i as $\hat{\mathcal{N}}^i = \mathcal{N}^i \cup \{l : l \in \mathcal{N}^j, j \in \bar{\mathcal{N}}^i\}$, where $\bar{\mathcal{N}}^i = \{j : \ i \in \mathcal{N}^j\}$, then the matrix $\mathbf{Q}$ has all the (i,j) block matrices $\mathbf{Q}^{ij} = 0$ for all $j \notin \hat{\mathcal{N}}^i$ and the matrix $\mathbf{W}$ has all the block

matrices $\mathbf{W}^{ij} = 0$ for all $j \notin \bar{\mathcal{N}}^i$, for any given subsystem i. Thus, the ith block components of ∇f can be computed using only local information:

$$\nabla_i f(\mathbf{u}) = \sum_{j \in \hat{\mathcal{N}}^i} \mathbf{Q}^{ij}\mathbf{u}^j + \sum_{j \in \bar{\mathcal{N}}^i} \mathbf{W}^{ij}x^j + \mathbf{w}^i. \tag{22}$$

Note that in Algorithm PCDM the only parameters that we need to compute are the Lipschitz constants L_i. However, in the MPC problem, L_i does not depend on the initial state x and can be computed locally by each subsystem as: $L_i = \lambda_{\max}(\mathbf{Q}^{ii})$. From the previous discussion it follows immediately that the iterations of Algorithm PCDM can be performed in parallel using distributed computations (see (22)).

Further, our Algorithm PCDM has a simpler implementation of the iterates than the algorithm from [23]: in Algorithm PCDM the main step consists of computing local projections on the sets $\mathbf{U}^i$ (in the context of MPC usually these sets are simple and the projections can be computed in closed form); while in the algorithm from [23] this step is replaced with solving local dense QP problems with the feasible set given by $\mathbf{U}^i$ (even in the context of MPC this local QP problems cannot be solved in closed form and an additional QP solver needs to be used). Finally, the number of iterations for finding an approximate solution can be easily predicted in our Algorithm (see Theorems 1 and 2), while in the algorithm from [23] the authors prove only asymptotic converge.

4. Numerical Results

Since our Algorithm PCDM has similarities with the algorithm from [23], in this section we compare these two algorithms on controlling a laboratory setup with DMPC (4 tank process) and on MPC problems for random network systems of varying dimension.

4.1. Quadruple tank process

To demonstrate the applicability of our Algorithm PCDM, we apply this newly developed method for solving the optimization problems arising from the

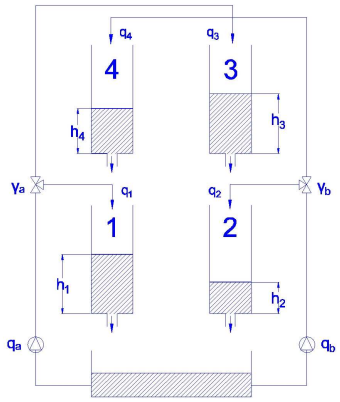

Figure 1: Quadruple tank process diagram.

MPC problem for a process consisting of four interconnected water tanks, see Fig. 1 for the process diagram, whose objective is to control the level of water in each of the four tanks. For this plant, there are two types of system inputs that can be considered: the pump flows, when the ratios of the three way valves are considered fixed, or the ratios of the three way valves, whilst having fixed flows from the pumps. In this paper, we consider the latter option, with the valve ratios denoted by γ_a and γ_b, such that tanks 1 and 3 have inflows $\gamma_a q_a$ and $(1-\gamma_a)q_a$, while tanks 2 and 4 have inflows $\gamma_b q_b$ and $(1-\gamma_b)q_b$. The simplified continuous nonlinear model of the plant is well known [1]. We use the following notation: h_i are the levels and a_i are the discharge constants of tank i, S is the cross section of the tanks, γ_a, γ_b are the three-way valve ratios, both in $[0, 1]$, while q_a and q_b are the pump flows.

Param	S	a_1	a_2	a_3	a_4	h_1^0	h_2^0	h_3^0	h_4^0	$q_{a/b}^{max}$	γ_a^0	γ_b^0
Value	0.02	$5.8e{-5}$	$6.2e{-5}$	$2e{-5}$	$3.6e{-5}$	0.19	0.13	0.23	0.09	0.39	0.58	0.54
Unit	m^2	m^2	m^2	m^2	m^2	m	m	m	m	$\frac{m^3}{h}$		

Table 1: Quadruple tank process parameters.

The discharge constants a_i, with $i = 1, \ldots, 4$ and the other parameters of the

model are determined experimentally from our laboratory setup (see Table 1). We can obtain a linear continuous state-space model by linearizing the nonlinear model at an operating point given by h_i^0, γ_a^0, γ_b^0, and the maximum inflows from the pumps, with the deviation variables $x^i = h_i - h_i^0$, $u^1 = \gamma_a - \gamma_a^0$, $u^2 = \gamma_b - \gamma_b^0$:

$$\frac{dx}{dt} = \begin{bmatrix} -\frac{1}{\tau_1} & 0 & 0 & \frac{1}{\tau_4} \\ 0 & -\frac{1}{\tau_2} & \frac{1}{\tau_3} & 0 \\ 0 & 0 & -\frac{1}{\tau_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_4} \end{bmatrix} x + \begin{bmatrix} \frac{q_a^{max}}{S} & 0 \\ 0 & \frac{q_b^{max}}{S} \\ \frac{-q_a^{max}}{S} & 0 \\ 0 & \frac{-q_b^{max}}{S} \end{bmatrix} u,$$

where $\tau_i = \frac{S}{a_i}\sqrt{\frac{2h_i^0}{g}}$, $i = 1, \ldots, 4$, is the time constant for tank i.

Using zero-order hold method with a sampling time of 5 seconds we obtain the discrete time model of type (12), with the partition $x^1 \leftarrow \begin{bmatrix} x^1 & x^4 \end{bmatrix}^T$ and $x^2 \leftarrow \begin{bmatrix} x^2 & x^3 \end{bmatrix}^T$. For the input constraints of the MPC scheme we consider the practical constraints of the ratios of the three way valves for our plant, i.e $u^i \in [0.15,\ 0.8] - \gamma_0^i$, where γ_0^i is the linearization input. Due to the fact that our plant has overflow sensors fitted to the tanks and an emergency shutoff program, we do not introduce constraints for the states. For the stage cost we have taken the weighting matrices to be $Q^i = I_{n_i}$ and $R^i = 0.01 I_{m_i}$.

4.2. Implementation of the MPC scheme using MPI

In this section we underline the benefits of Algorithm PCDM when it is implemented in an appropriate fashion for the quadruple tank MPC scheme. We implemented for comparison, Algorithm PCDM and that of [23]. Both algorithms were implemented in C programming language, with parallelization ensured via MPI and linear algebra operations done with CLAPACK. Algorithm [23] requires solving, at each step, 2 QP problems in parallel, problems which cannot be solved in closed form. For solving these QP problems, we use the *qpip* routine of the QPC toolbox [26]. The algorithms were implemented on a PC, with 2 Intel Xeon E5310 CPUs at 1.60 GHz and 4Gb of RAM. For the MPC problem in this subsection we control the plant such that the levels and inputs will reach those of the steady state linearization values h^0 and γ^0.

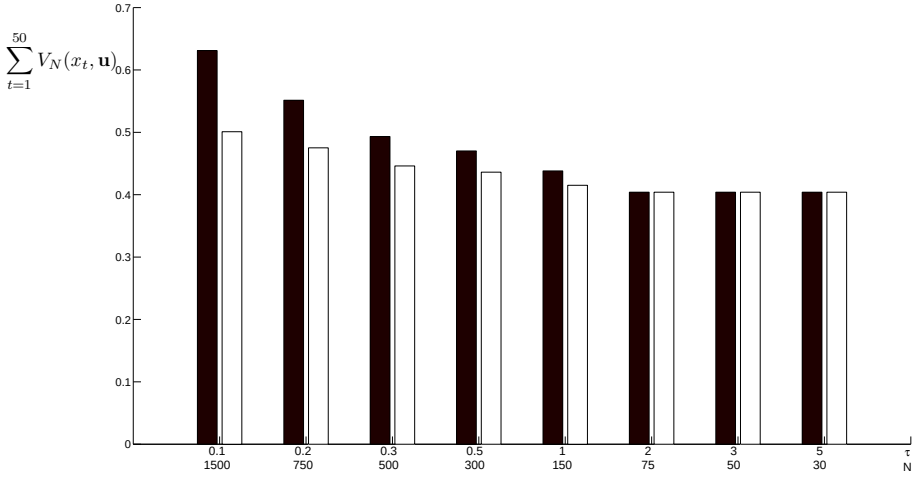

Figure 2: Total costs for 50 MPC steps with PCDM (white) and [23] (black).

Figure 2 outlines a comparison of the two algorithms for solving this quadruple tank MPC problem, considering a total prediction time of 150 seconds, for different prediction horizons N and available time τ, such that $\tau N = 150$ seconds. The bar values represent the total sum $\sum_{t=1}^{50} V_N(x_t, \mathbf{u})$, where $\mathbf{u}$ is calculated either with PCDM or with the algorithm from [23]. For the same 50 simulation steps, we outline in Table 2 a comparison of the average number of iterations achieved by both algorithms and the performance loss, i.e a percentile difference between the suboptimal cost achieved in Figure 2 (i.e. $\sum_{t=1}^{50} V_N(x_t, \mathbf{u})$) and the optimal costs that were precalculated with Matlab's quadprog (i.e. $\sum_{t=1}^{50} V_N^*(x_t)$), both for the time τ and prediction horizon N. Note that our total cost is usually better than that of [23] when the available time is short and for $\tau \geq 2$ both algorithms solve the corresponding optimization problem exactly. Also note that, due to its low complexity iteration, our algorithm performs more than ten times the amount of iterations than the algorithm from [23].

		PCDM	[23]
τ	N	Iter / Perf. Loss (%)	Iter / Perf. Loss (%)
0.1	1500	7 / 23.9	1 / 60.18
0.2	750	30 / 17.59	2 / 36.48
0.3	500	240 / 10.42	5 / 22.1
0.5	300	1803 / 7.94	22 / 16.36
1	150	12244 / 2.74	258 / 8.44
2	75	67470 / 0	2495 / 0
3	50	153850 / 0	8663 / 0
5	30	382810 / 0	38110 / 0

Table 2: Number of iterations and performance loss, for different times τ.

4.3. Implementation of the MPC scheme using Siemens S7-1200 PLC

Due to the limitations, in both hardware and programming language, of the S7-1200 PLC, any proper implementation of a distributed optimization algorithm, in the sense of distributed computations and passing information between processes running on different cores, cannot be undertaken on it. However, to illustrate the fact that our PCDM algorithm is suitable for control devices with limited computational power and memory, we implemented it in a centralized manner for an MPC scheme in order to control the quadruple tank plant. We note that S7-1200 PLC is considered an entry-level PLC, with 50 KB of main memory, 2 MB of load memory (mass storage) and 2 KB of backup memory. There are two main function blocks for the algorithm itself, one that updates $q(x) = \mathbf{W}x + \mathbf{w}$ in the quadratic objective function f given the current levels of the four tanks and one in which Algorithm PCDM is implemented for solving problem (1). Both blocks contain Structured Control Language which corresponds to IEC 1131.3 standard. The remaining function blocks are used for converting the I/O for the plant to corresponding metric values. The elements of the problem which occupy the most memory is the $\mathbf{Q} \in \mathbb{R}^{2N\times 2N}$ matrix of the objective function f and matrix $\mathbf{W} \in \mathbb{R}^{2N\times 4}$ for updating $q(x)$. Both matrices

Cycle Time	5 s		
Prediction Horizon N	10	20	30
Maximum Number of Iter.	104	39	15
Used Memory (%)	59	72	88

Table 3: Available number of iterations and memory usage of Alg. PCDM

are precomputed offline using Matlab and then stored in the work memory using Data Blocks. The components of the problem which require updating are the input trajectory vectors $\mathbf{u}^i$ and the vector $q(x)$ of the objective function $f(\mathbf{u})$ which is dependent of the current state of the plant and of the current set point. The evolution of the tank levels and input ratios of the plant are recorded in Matlab on the plant's PC workstation, via an OPC server and Ethernet connection. In accordance with the imposed sample time of 5 seconds, the cycle time of the S7-1200 PLC is also limited to this interval.

Due to this cycle time, the limited size of the S7-1200's work memory and its processing speed, the number of iterations of the Algorithm PCDM that can be computed are also limited. In Table 3 the number of iterations available per prediction horizon, included in the 5 seconds cycle time, and the memory requirements for these prediction horizons are presented. Although the numbers of computed iterations seem small, we have found in practice that the suboptimal MPC scheme still stabilizes the quadruple tank process and ensures set point tracking. The results of the control process are presented in Fig. 3 for a prediction horizon $N = 20$: the continuous lines represent the evolution of water levels in each of the four tanks and the inputs, while the dashed lines are their respective set points. We choose two set points. We first let the plant get near its first setpoint, after which we choose a new set point which is an equilibrium point for the plant. As it can be observed from the figure, the MPC scheme still steers the process to the respective set points.

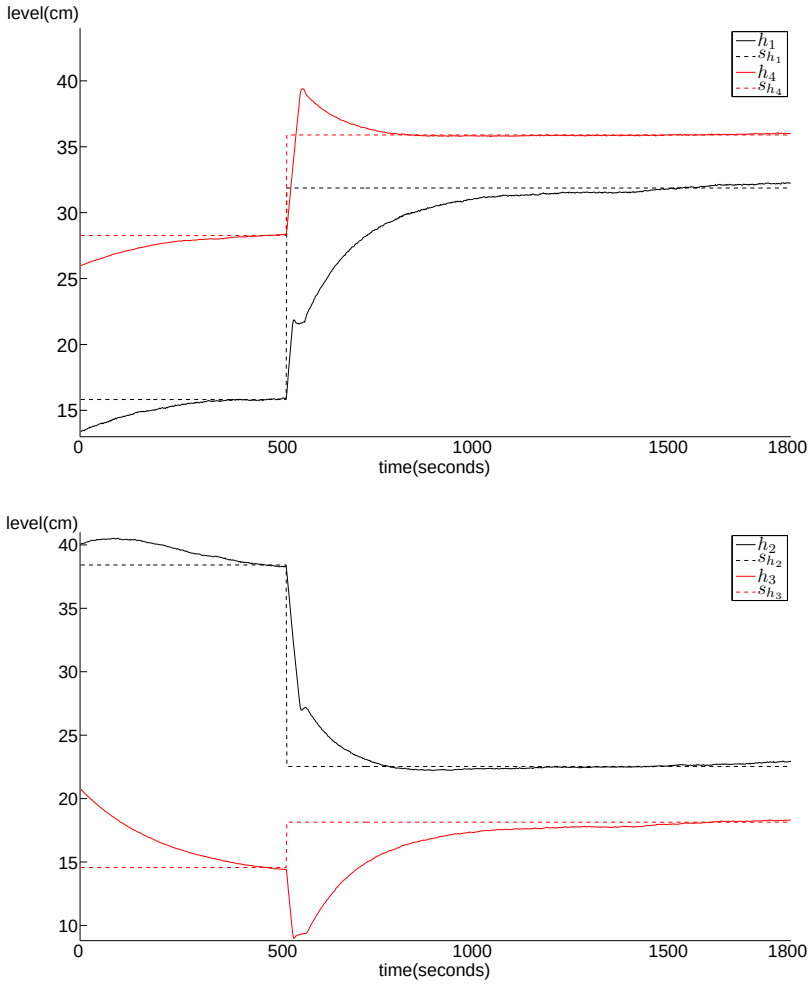

Figure 3: Evolution of tank levels 1-4 (top), 2-3 (bottom), with continuous lines, against their respective set points, with dashed lines.

4.4. Implementation of MPC scheme for random network systems

We now wish to outline a comparison of results between algorithm PCDM and that of [23] when solving QP problems arising from MPC for random network systems. Both algorithms were implemented in the same manner as described in Section 4.2. We considered random network systems with dynamics (11) generated as follows: the entries of system matrices A^{ij} and B^{ij} are taken from a normal distribution with zero mean and unit variance. Matrices A^{ij}

are then scaled, so that they become neutrally stable. Matrices $Q^i \succeq 0$ and $R^i \succ 0$ are random. The input variables are constrained to lie in box sets whose boundaries are generated randomly. The terminal cost matrices P^i are taken to be the solution of the SDP problem given in Lemma 2. For each subsystem the number of inputs is taken $m_i = 5$ or $m_i = 10$. We let the prediction horizon range between $N = 6$ to $N = 120$. The subsystems are arranged in a ring, i.e. $\mathcal{N}^i = \{i-1, i, i+1\}$. We first considered $M = 8$ subsystems, matching the number of cores on our PC. Parallel implementation was also carried out for $M = 16$ subsystems, with each core of the PC running two processes. The resulting random QP problems have $p = MNm_i$ variables. The stopping criterion for each algorithm is $f(\mathbf{u}_k) - f^* \leq 0.001$, with f^* being precomputed for each problem using Matlab's quadprog. For each prediction horizon, 10 simulations were run, starting from different random initial states.

		PCDM		[23]		PCDM centralized	Quadprog
M	p	CPU (s)	Iter	CPU (s)	Iter	CPU (s)	CPU (s)
8	480	0.47	1396	1.904	682	0.663	1.08
	960	2.21	2839	21.52	1475	9.15	3.57
	3200	256.4	8671	911.2	4197	265.3	39.8
	4800	857.2	12750	7864.4	6182	1114	139.9
	9600	2223.1	16950	*	*	3125	307.5
16	480	4.36	2600	4.66	1615	0.99	0.97
	960	15.02	4792	25.18	2798	14.17	3.21
	3200	377.6	13966	612.8	8462	423.1	41.3
	4800	1524.7	23539	3061.7	14241	2161.1	134.03
	9600	3415.1	29057	*	*	4773	308.4

Table 4: CPU processing time in seconds and number of iterations for algorithms PCDM and [23].

Table 4 presents the average CPU time in seconds for the execution of each algorithm. It illustrates that Algorithm PCDM, with its design for distributed

computations and simple iterations, usually performs better than that in [23], where the assumption is that for each iteration, a QP problem of size p/M needs to be solved. The entries with $*$ denote that the algorithm would have taken over 5 hours to complete. Also note that our implementation of the algorithm from [23], for problems of larger dimensions, i.e starting with $p = 3200$, takes less time for it to complete if the problem is divided between $M = 16$ subsystems than $M = 8$. This is due to the fact that the solver *qpip* takes much more time to solve problems of size 600 in the case of $p = 4800$ and $M = 8$ than problems of size 300 for $p = 4800$ and $M = 16$. Also, the transmission delays between subsystems are negligible in comparison with these *qpip* times. We have also implemented Algorithm PCDM in a centralized manner, i.e. without using MPI and, as can be seen from the table, we gain speedups of computation when the algorithm is parallelized. Algorithm PCDM is outperformed by Matlab's quadprog, but do note that quadprog is not designed for distributed implementation and there are no transmission delays between processes.

5. Conclusions

In this paper we have proposed a parallel optimization algorithm for solving smooth convex problems with separable constraints that may arise e.g in MPC for general linear systems comprised of interconnected subsystems. The new optimization algorithm is based on the block coordinate descent framework but with very simple iteration complexity and using local information. We have shown that for strongly convex objective functions it has linear convergence rate. An MPC scheme based on this optimization algorithm was derived, for which every subsystem in the network can compute feasible and stabilizing control inputs using distributed computations. An analysis for obtaining local terminal costs from a distributed viewpoint was made which guarantees stability of the closed-loop interconnected system. Preliminary numerical tests show that this algorithm is suitable for MPC applications, especially those with hardware that has low computational power.

References

[1] I. Alvarado, D. Limon, D. Munoz de la Pena, 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.

[2] D.P. Bertsekas and J. Tsitsiklis, *Paralel and distributed computation: Numerical Methods*, Prentice Hall, 1989.

[3] E. Camponogara, H.F. Scherer, *Distributed Optimization for Model Predictive Control of Linear Dynamic Networks With Control-Input and Output Constraints*, IEEE Transactions on Automation Science and Engineering, 8(1), 233–242, 2011.

[4] M.D. Doan, T. Keviczky and B. De Schutter, *A distributed optimization-based approach for hierarchical model predictive control of large-scale systems with coupled dynamics and constraints*, Proceedings of Conference on Decision and Control, 5236–5241, 2011.

[5] W.B. Dunbar, *Distributed receding horizon control of dynamicall coupled nonlinear systems*, IEEE Transactions on Automatic Control, 52(7), 1249–1263, 2007.

[6] M. Farina and R. Scattolini, *Distributed predictive control: a non-cooperative algorithm with neighbor-to-neighbor communication for linear systems*, Automatica, to appear, 2012.

[7] D. N Godbole, F. H Eskafi and P. P Varaiya, *Automated Highway Systems*, Proceedings of 13th IFAC World Congress, 121–126, 1996.

[8] M. Golshan, J.F. MacGregor, M.J. Bruwer and P. Mhaskar, *Latent Variable Model Predictive Control (LV-MPC) for trajectory tracking in batch processes*, Journal of Process Control, 20(4), 538–550, 2010.

[9] B. Hu and A. Linnemann, *Toward Infinite-Horizon Optimality in Nonlinear Model Predictive Control*, IEEE Transactions on Automatic Control, 47(4), 679–682, 2002.

[10] J.L. Jerez, K.-V. Ling, G.A. Constantinides and E.C. Kerrigan, *Model predictive control for deeply pipelined field-programmable gate array implementation: algorithms and circuitry*, IET Control Theory and Applications, 6(8), 1029–1041, 2012.

[11] A. Jokic and M. Lazar, *On Decentralized Stabilization of Discrete-time Nonlinear Systems*, Proceedings of American Control Conference, 5777–5782, 2009.

[12] D. Limon, T. Alamo and E.F. Camacho, *Stable Constrained MPC without Terminal Constraint*, Proceedings of American Control Conference, 4893–4898, 2003.

[13] P. Massioni and M. Verhaegen, *Distributed Control for Identical Dynamically Coupled Systems: A Decomposition Approach*, IEEE Transactions on Automatic Control, 54(1), 124–135, 2009.

[14] M.V. Nayakkankuppam, *Solving large-scale semidefinite programs in parallel*, Mathematical Programming, 109, 477–504, 2007.

[15] I. Necoara, V. Nedelcu and I. Dumitrache, *Parallel and distributed optimization methods for estimation and control in networks*, Journal of Process Control, 21, 756-766, 2011.

[16] I. Necoara, D. Doan and J. A. K. Suykens, *Application of the proximal center decomposition method to distributed model predictive control*, Proceedings of the Conference on Decision and Control, 2900–2905, 2008.

[17] Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM Journal of Optimization, 22(2), 341–362. 2012.

[18] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer, 2004.

[19] J.A. Primbs and V. Nevistic, *A New Approach to Stability Analysis for Constrained Finite Receding Horizon Control Without End Constraints*, IEEE Transactions on Automatic Control, 45(8), 1507–1512, 2000.

[20] J.B. Rawlings and D.Q. Mayne, *Model Predictive Control: Theory and Design*, Nob Hill Publishing, 2009.

[21] R. Scattolini, *Architectures for distributed and hierarchical Model Predictive Control – A review*, Journal of Process Control, 19(5), 723–731, 2009.

[22] P.O.M. Scokaert, D.Q. Mayne and J.B. Rawlings, *Suboptimal model predictive control (feasibility implies stability)*, IEEE Transactions on Automatic Control, 44(3), 648–654, 1999.

[23] B. T. Stewart, A.N. Venkat, J.B. Rawlings, S. Wright and G. Pannocchia, *Cooperative distributed model predictive control*, Systems & Control Letters, 59, 460–469, 2010.

[24] G. Valencia-Palomo and J.A. Rossiter *Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information*, ISA Transactions, 50, 92–100, 2011.

[25] A.N Venkat, I.A. Hiskens, J.B Rawlings and S. Wright, *Distributed MPC strategies with application to power system automatic generation control*, IEEE Transactions on Control Systems Technology, 16(6), 1192–1206, 2008.

[26] A. Wills, QPC - Quadratic Programming in C, University of Newcastle, Australia, *http://sigpromu.org/quadprog/index.html*.

[27] L. Wang, *Discrete model predictive control design using Laguerre functions*, Journal of Process Control, 14, 131–142, 2004.