2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

A computationally efficient parallel coordinate descent algorithm for MPC: implementation on a PLC

Ion Necoara and Dragos N. Clipici

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 model predictive control (MPC) for linear network systems. The basis for the new algorithm are block coordinate descent updates, that are computed in parallel, have low iteration complexity and use only local information. As a result, the algorithm is suitable for implementation on autonomous hardware with low computational power. In the case of a strongly convex objective function, we prove that the algorithm has linear rate of convergence. An MPC scheme based on this algorithm is derived, such that the computations of feasible and stabilizing inputs are distributed and cheap, and can be done by each subsystem in part. Some implementation issues of the new algorithm for MPC problems are discussed, as well as it being tested on a 4 tank laboratory setup.**

I. Introduction

Model predictive control (MPC) has become a popular advanced control technology implemented in large scale industrial plants due to its ability to handle hard input and state constraints. An MPC scheme consists of solving, at each time instant, an optimization problem whose variables are given e.g. by the inputs of the system. Once the optimal solution is computed, only the first input is injected into the system after which the whole procedure is repeated.

MPC strategies for large network systems can be split into different categories. For example, in centralized control, a single entity is responsible for the computation of all inputs and transmitting them to controllers that are dispersed in wide areas across the plant. In distributed control, the interactions between subsystems are taken into account and each subsystem takes all decisions based on local information [2], [3], [8], [9], [13], [15]. Distributed control, where the interactions between subsystems are only considered locally, has been approached in [2], [3], [7]. In [8], [15], [13] the authors considered the distributed MPC problem from the cooperative point of view, where distributed optimization algorithms are employed such that each controller optimizes a centralized controller objective. In [1], [9] dual distributed gradient algorithms based on Lagrange relaxation of the coupling constraints are presented for solving MPC problems.

MPC schemes have a well-known disadvantage of being costly when it comes to computations. More classical methods do not require powerful computational hardware and prove to be a cheaper alternative. However, many plants are comprised of subsystems that have strong couplings, for which classical controllers may not even provide stability. Due to these isssues, research for MPC schemes for embedded control has gained wider attention in the past few years. Embedded MPC is concerned with designing a control scheme that is suitable for autonomous hardware, e.g a programmable logic controller (PLC) [16], a microcontroller circuit board [18], field-programmable gate arrays (FPGAs) [5], etc. These devices differ vastly in computational power and memory capabilities. Therefore, there is a ever increasing focus on obtaining faster MPC schemes by reducing problem size through decentralization and improving the efficiency of computations. As a result, these schemes should be suitable for implementation on cheaper hardware, which often has reduced computational power.

The main contribution of this paper is the development of a parallel coordinate descent method that is computationally efficient and suitable for MPC schemes that can be e.g. implemented in hardware with limited memory and computational power such as embedded electronic circuitry. We propose an MPC strategy for a general network system in which both state and input interactions between subsystems are modeled. Stability of the MPC scheme is ensured using an appropriate terminal cost and without terminal constraints [6]. We develop an efficient parallel optimization algorithm for solving the MPC problem, with low iteration complexity and linear rate of convergence, which is deemed suitable for parallel computation hardware. The algorithm is based on block-coordinate updates for the optimization variables, under the assumption of an objective function with coordinate-wise Lipschitz continuous gradient and strong convexity, and is similar to the optimization algorithm proposed in [15], but with simpler iteration complexity and guaranteed rate of convergence. Due to the fact that the MPC optimization problem usually is terminated before convergence, the resulting MPC controller is a form of suboptimal control. However, feasibility and stability of the MPC scheme is still guaranteed via the suboptimal control theory [6], [14].

This paper is organized as follows. In Section II the model for general network systems is introduced, and the stability conditions of the MPC scheme are outlined. We present our parallel optimization algorithm in Section III and prove its convergence rate. In Section IV the algorithm is applied on a PLC for controlling our own laboratory 4 tank process and outline the control results.

II. MPC for network systems

In this paper we consider discrete-time network systems which are comprised of M interconnected subsystems,

I. Necoara and D.N. Clipici are with University Politehnica Bucharest, Automatic Control and Systems Engineering Department, 060042 Bucharest, Romania. {ion.necoara, dragos.clipici}@acse.pub.ro.

whose dynamics are defined by the following linear state space equations:

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

where $x_t^j \in \mathbb{R}^{n_j}$ and $u_t^j \in \mathbb{R}^{m_j}$ represent the state and respectively the input of jth subsystem at time t, $A^{ij} \in \mathbb{R}^{n_i \times n_j}$, $B^{ij} \in \mathbb{R}^{n_i \times m_j}$, $\mathcal{N}^i$ is the set of indices which contains the index i and that of its neighboring subsystems. A particular case of the system (1), that is frequently found in literature [1], [9], [15], has the following dynamics:

$$x_{t+1}^i = A^{ii} x_t^i + \sum_{j \in \mathcal{N}^i} B^{ij} u_t^j. \tag{2}$$

The dynamics of the entire system can be expressed as:

$$x_{t+1} = Ax_t + Bu_t,$$

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 a system of type (1) or (2) we consider local input constraints:

$$u_t^i \in U^i \quad i = 1, \cdots, M, \;\; t \geq 0, \tag{3}$$

with $U^i \subseteq \mathbb{R}^{m_i}$ compact, convex sets with the origin in their interior. We also consider local stage costs and terminal costs for each subsystem:

$$\ell^i(x_t^i, u_t^i), \;\; \ell_{\mathrm{f}}^i(x_t^i) \;\; i = 1, \ldots, M.$$

We can now formulate the MPC problem for system (1) over a prediction horizon of length N and a given initial state x:

$$\begin{aligned} V^*(x) = \min_{x_t^i, u_t^i} & \sum_{i=1}^{M} \sum_{t=0}^{N-1} \ell^i(x_t^i, u_t^i) + \ell_{\mathrm{f}}^i(x_N) \\ \text{s.t: } x_{t+1}^i = & \sum_{j \in \mathcal{N}^i} A^{ij} x_t^j + B^{ij} u_t^j, \\ & x_0^i = x^i, \; u_t^i \in U^i \;\; i = 1, \cdots, M, \;\; t \geq 0. \end{aligned} \tag{4}$$

Let us define the the global terminal cost $\ell_{\mathrm{f}}(x_t) = \sum_i \ell_{\mathrm{f}}^i(x_t^i)$. We assume that stability of this MPC scheme (4) is enforced by adapting the terminal cost ℓ_{f} and the horizon length N appropriately such that sufficient stability criteria are fulfilled [4], [6], [8], [14]. Usually, stability of MPC with quadratic stage cost and without terminal constraint is enforced if the following criteria holds: there exists a stabilizing feedback law $\kappa(\cdot)$ and a terminal cost $\ell_{\mathrm{f}}(\cdot)$ such that the following property is satisfied:

$$\begin{aligned} \ell_{\mathrm{f}}(Ax + B\kappa(x)) - & \ell_{\mathrm{f}}(x) + \kappa(x)^T R \kappa(x) \\ & + x^T Q x \leq 0 \;\; \forall x \in \mathbb{R}^n, \end{aligned} \tag{5}$$

where the matrices Q and R have a block diagonal structure and are composed of the blocks Q^i and R^i, respectively.

Keeping in line with the distributed nature of our system, the control law $\kappa(\cdot)$ and the final stage cost $\ell_{\mathrm{f}}(\cdot)$ can be computed locally (see [8] for more details).

III. Distributed MPC based on coordinate descent optimization

We now denote the input trajectory for subsystem i by:

$$\mathbf{u}^i = (u_0^i, \cdots, u_{N-1}^i),$$

and the overall input trajectory for the entire system as:

$$\mathbf{u} = (\mathbf{u}^1, \cdots, \mathbf{u}^M).$$

By eliminating the states from the dynamics (1), MPC problem (4) can be expressed as a large scale optimization problem of the form [1]:

$$\begin{aligned} f^* = & \min_{\mathbf{u}} f(\mathbf{u}^1, \ldots, \mathbf{u}^M) \\ & \text{s.t: } \mathbf{u}^i \in \mathbf{U}^i \;\; \forall i = 1, \cdots, M, \end{aligned} \tag{6}$$

where the function f is smooth and convex, whilst the convex sets $\mathbf{U}^i \subseteq \mathbb{R}^{Nm_i}$ are the Cartesian product of the sets U^i for N times.

A. Parallel Coordinate Descent Method

In this section we propose a block-coordinate descent based algorithm which permits optimization problem (6) to be solved efficiently with low computational cost per iteration and guaranteed convergence rate. We define the corresponding partition of the identity matrix:

$$I = (E^1, \ldots, E^M) \in \mathbb{R}^{Nm \times Nm},$$

where $E^i \in \mathbb{R}^{Nm \times Nm_i}$ for all $i = 1, \cdots, M$. Thus the input trajectory $\mathbf{u}$ can be represented as:

$$\mathbf{u} = \sum_{i=1}^{M} E^i \mathbf{u}^i.$$

We define the partial gradient $\nabla_i f(\mathbf{u}) \in \mathbb{R}^{Nm_i}$ 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:

$$\begin{aligned} \left\| \nabla_i f(\mathbf{u} + E^i h_i) - \nabla_i f(\mathbf{u}) \right\| \leq L_i \left\| h_i \right\| \\ \forall \; \mathbf{u} \in \mathbb{R}^{Nm}, \; h_i \in \mathbb{R}^{Nm_i}, \end{aligned}$$

where, for this paper, $\|\cdot\|$ denotes the standard Euclidean norm and $\langle \cdot, \cdot \rangle$ denotes the standard inner product on $\mathbb{R}^n$. Due to the assumption that f is coordinate-wise Lipschitz continuous, it can be easily derived that [11]:

$$\begin{aligned} f(\mathbf{u} + E^i h_i) \leq & f(\mathbf{u}) + \langle \nabla_i f(\mathbf{u}), h_i \rangle + \frac{L_i}{2} \left\| h_i \right\|^2 \\ & \forall \; \mathbf{u} \in \mathbb{R}^{Nm}, \; h_i \in \mathbb{R}^{Nm_i}. \end{aligned} \tag{7}$$

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

$$\left\| \mathbf{u} \right\|_1^2 = \sum_{i=1}^{M} L_i \left\| \mathbf{u}^i \right\|^2, \tag{8}$$

which will prove useful for estimating the rate of convergence of our algorithm. Additionally, if function f is smooth

and strongly convex with regards to $\|\cdot\|_1$ with a parameter σ_1, then [12]:

$$f(\mathbf{w}) \geq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{w}-\mathbf{v}\rangle + \frac{\sigma_1}{2}\|\mathbf{w}-\mathbf{v}\|_1^2 \quad (9)$$
$$\forall \mathbf{w}, \mathbf{v} \in \mathbb{R}^{Nm}.$$

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 (9) we also get:

$$f(\mathbf{u}+E^i h_i) \geq f(\mathbf{u}) + \langle \nabla_i f(\mathbf{u}), h_i\rangle + \frac{\sigma_1 L_i}{2}\|h_i\|^2$$
$$\forall \mathbf{u} \in \mathbb{R}^{Nm}, h_i \in \mathbb{R}^{Nm_i},$$

and combining with (7) we also deduce that $\sigma_1 \leq 1$.

Let us define the constrained coordinate update for our algorithm:

$$\mathbf{v}^i(\mathbf{u}) = \arg\min_{\mathbf{v}^i \in \mathbf{U}^i} \left\langle \nabla_i f(\mathbf{u}), \mathbf{v}^i - \mathbf{u}^i\right\rangle + \frac{L_i}{2}\left\|\mathbf{v}^i - \mathbf{u}^i\right\|^2$$
$$\mathbf{u}^i_+(\mathbf{u}) = \mathbf{u} + E^i(\mathbf{v}^i(\mathbf{u}) - \mathbf{u}^i)\ \forall i = 1, \ldots, M.$$

The optimality conditions for the previous optimization problem are:

$$\left\langle \nabla_i f(\mathbf{u}) + L_i(\mathbf{v}^i(\mathbf{u}) - \mathbf{u}^i), \mathbf{v}^i - \mathbf{v}^i(\mathbf{u})\right\rangle \geq 0\ \forall \mathbf{v}^i \in \mathbf{U}^i. \quad (10)$$

We now present our *Parallel Coordinate Descent Method*, that resembles the method in [15] but with simpler iteration update and lower computational complexity, and is a parallel version of the coordinate descent method from [11]:

Algorithm PCDM($\mathbf{u_0}$)

1) **For** $k \geq 0$**, compute in parallel** $\mathbf{v}^i(\mathbf{u}_k)$ **for** $i = 1, \ldots, M$**.**
2) **Udpate in parallel:**

$$\mathbf{u}^i_{k+1} = \frac{1}{M}\mathbf{v}^i(\mathbf{u}_k) + \frac{M-1}{M}\mathbf{u}^i_k\ \ \forall i.$$

From (10), the convexity of f and $\mathbf{u}_{k+1} = \sum_i \frac{1}{M}\mathbf{u}^i_+(\mathbf{u}_k)$ it can be easily proved that PCDM decreases the objective function at each iteration:

$$f(\mathbf{u}_{k+1}) \leq f(\mathbf{u}_k)\ \ \forall k \geq 0. \quad (11)$$

The following theorem provides the convergence rate of the Algorithm PCDM and employs standard techniques for proving the rate of convergence of the projected gradient method [12]:

Theorem 1: If function f has a coordinate-wise Lipschitz continuous gradient with constants L_i and is strongly convex with regards to $\|\cdot\|_1$ with a constant σ_1, 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 introduce the following term:

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

where $\mathbf{u}_*$ is the optimal solution of (6) and $\mathbf{u}^i_* = (E^i)^T\mathbf{u}_*$. For the next iterate we obtain:

$$r_{k+1}^2 \overset{(10)}{\leq} r_k^2 + \sum_{i=1}^M \frac{L_i}{M}(\frac{1}{M} - 2)\left\|\mathbf{v}^i(\mathbf{u}_k) - \mathbf{u}^i_k\right\|^2 + \frac{2}{M}\left\langle \nabla_i f(\mathbf{u}_k), \mathbf{u}^i_* - \mathbf{v}^i(\mathbf{u}_k)\right\rangle$$
$$\overset{\frac{1}{M}\leq 1}{\leq} r_k^2 - \frac{2}{M}\sum_{i=1}^M (\frac{L_i}{2}\left\|\mathbf{v}^i(\mathbf{u}_k) - \mathbf{u}^i_k\right\|^2 + \left\langle \nabla_i f(\mathbf{u}_k), \mathbf{v}^i(\mathbf{u}_k) - \mathbf{u}^i_k\right\rangle + \left\langle \nabla_i f(\mathbf{u}_k), \mathbf{u}^i_* - \mathbf{u}^i_k\right\rangle).$$

By convexity of f and (7) we obtain:

$$r_{k+1}^2 \leq r_k^2 - 2(f(\mathbf{u}_{k+1}) - f(\mathbf{u}_k)) + \frac{2}{M}\left\langle \nabla f(\mathbf{u}_k), \mathbf{u}^* - \mathbf{u}_k\right\rangle, \quad (12)$$

We take $\mathbf{w} = \mathbf{u}^*$ and $\mathbf{v} = \mathbf{u}_k$ in (9) and through (12) we get:

$$\frac{1}{2}r_{k+1}^2 + f(\mathbf{u}_{k+1}) - f^* \leq \frac{1}{2}r_k^2 + f(\mathbf{u}_k) - f^* - \frac{1}{M}(f(\mathbf{u}_k) - f^* + \frac{\sigma_1}{2}r_k^2). \quad (13)$$

From the strong convexity of f in (9) 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 (13) 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 convergence result 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. ■

B. Application of Algorithm PCDM to distributed suboptimal MPC

In this section we discuss some technical aspects for the implementation of our Algorithm PCDM in the case of MPC problem (4). 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 $\|x\|_P^2 = x^T P x$, the matrix $Q^i, P^i \in \mathbb{R}^{n_i \times n_i}$ 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 (4), after eliminating the dynamics, is quadratically strongly convex, having the form:

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

where $\mathbf{Q}$ is positive definite due to the assumption that all R^i are positive definite. Usually, for the dynamics (1) 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 (2) 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 neighbors). Indeed, if the dynamics of the system are given by (2), then

$$x_{t+1}^i = (A^{ii})^t x_t^i + \sum_{l=1}^{t} \sum_{j \in \mathcal{N}^i} (A^{ii})^{l-1} B^{ij} u_{t-l}^j$$

and thus the matrices $\mathbf{Q}$ and $\mathbf{W}$ have a sparse structure. 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 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:

$$\nabla_i f(\mathbf{u}) = \sum_{j \in \hat{\mathcal{N}}^i} \mathbf{Q}^{ij} \mathbf{u}^j + \sum_{j \in \mathcal{N}^i} \mathbf{W}^{ij} x^j + \mathbf{w}^i. \qquad (14)$$

Note that in the 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 (14)), and is suitable to be implemented on hardware with parallel computational abilities.

Now, if the sets $\mathbf{U}^i$ are simple (by simple we understand that the projection on this sets is easy), then computing $\mathbf{v}^i(\mathbf{u})$ 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 $\mathbf{U}^i = \left\{\mathbf{v}^i \in \mathbb{R}^{Nm_i} | \mathbf{v}_{min}^i \leq \mathbf{v}^i \leq \mathbf{v}_{max}^i\right\}$, then the complexity of computing $\mathbf{v}^i(\mathbf{u})$, once $\nabla_i f(\mathbf{u})$ is available, is $\mathcal{O}(Nm_i)$.

In turn, computing $\nabla_i f(\mathbf{u})$ has complexity $\mathcal{O}(N^2 m m_i)$ for quadratic dense functions. Further, when we have sparse Hessians (see e.g. (14)), the complexity is much lower. In conclusion, Algorithm PCDM has usually a very low computing cost per iteration, compared to other existing methods, e.g. Jacobi type algorithm presented in [15], which usually require numerical complexity $\mathcal{O}((Nm_i)^3 + N^2 m m_i)$ per iteration in case when the local quadratic problems are solved with an interior point solver. Finally, the number of iterations for finding an approximate solution can be easily predicted in our Algorithm (see Theorem 1), while in the algorithm from [15] the authors prove only asymptotic converge.

IV. Numerical Results

To demonstrate the applicability of our Algorithm PCDM, we apply the newly developed method to the MPC problem for our own laboratory installation consisting of four water tanks, with tanks 2 and 3, 1 and 4, connected to each other via outflows, whose objective is to control the level of water in each of the four tanks. The simplified continuous nonlinear model of the plant is well known [8]. 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 $(1-\gamma_b)q_b$. We can obtain a 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 $q_a^0 = q_a^{max}$, $q_b^0 = q_b^{max}$, 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 S is the cross section of the tanks, a_i are the discharge constants for each tank and $\tau_i = \frac{S}{a_i}\sqrt{\frac{2h_i^0}{g}}$, $i = 1, \ldots, 4$ are the time constants for the tanks. The discharge constants a_i, with $i = 1, \ldots, 4$ and the other parameters of the model are determined experimentally (see Table I).

Param.	Value	Unit	Description
S	0.02	m^2	Cross-section of all tanks.
a_1	$5.8e{-5}$	m^2	Discharge constant of tank 1.
a_2	$6.2e{-5}$	m^2	Discharge constant of tank 2.
a_3	$2e{-5}$	m^2	Discharge constant of tank 3.
a_4	$3.6e{-5}$	m^2	Discharge constant of tank 4.
h_1^0	0.19	m	Linearization level of tank 1.
h_2^0	0.13	m	Linearization level of tank 2.
h_3^0	0.23	m	Linearization level of tank 3.
h_4^0	0.09	m	Linearization level of tank 4.
q_a^{max}	0.39	$\frac{m^3}{h}$	Maximal and constant inflow.
q_b^{max}	0.39	$\frac{m^3}{h}$	Maximal and constant inflow.
γ_a^0	0.58		Equilibrium ratio of valve a.
γ_b^0	0.54		Equilibrium ratio of valve b.

TABLE I: Quadruple tank process parameters.

We obtain the discrete time model of form (2) via the zero-order hold method with a sample time of 5 seconds and partitioning the plant into two subsystems: the first consisting of tanks 1 and 4 and the other of tanks 2 and 3.

For the input constraints of the MPC scheme we consider the practical constraints of the ratios of the three way valves four 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 $R^i = 0.01 I_{m_i}$.

A. Embedded implementation on a Siemens S7-1200 PLC

The MPC scheme was implemented on the plant's Siemens Simatic S7-1200 PLC, in separate function block diagrams (FBDs). The 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. Although the S7-1200 lacks the hardware for parallel computations and is limited in memory and computational power, we still wish to outline the fact that Algorithm PCDM, due to its simple iterations, can still be implemented on it.

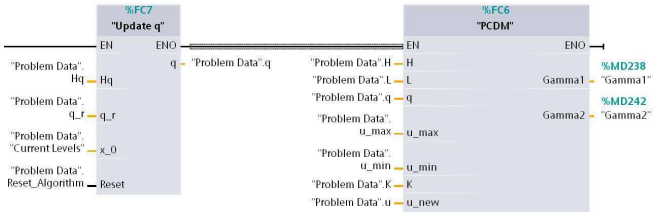

Fig. 1: Main function blocks for the PLC implementation.

There are two main function blocks for the algorithm itself, see Fig. 1, one that updates q 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 (6). The remaining function blocks are used for converting the I/O for the plant to corresponding metric values and for implementing the PI controllers required to replicate actual three-way valves. 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 $f(\mathbf{u})$ and a matrix $\mathbf{W} \in \mathbb{R}^{2N \times 4}$ necessary for updating $q(x) = \mathbf{W}x + \mathbf{w}$. Both matrices, as well as the linear control law matrix K, from $\kappa(x) = Kx$, such that (5) is satisfied, 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 on the current state of the plant and on 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.

Cycle Time	5 s		
Prediction Horizon N	10	20	30
Maximum Number of Iterations	104	39	15

TABLE II: Available number of iterations of Alg. PCDM

Prediction Horizon N	10	20	30
Used Memory (%)	59	72	88

TABLE III: PLC memory usage

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 II the number of iterations available per prediction horizon, included in the 5 seconds cycle time, are presented, whereas in Table III the memory requirements for the different prediction horizons are presented. Although the number 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. 2 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.

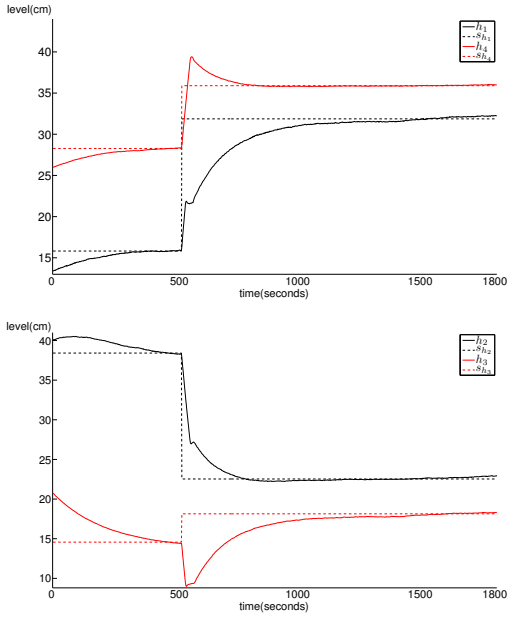

Fig. 2: Evolution of tank levels 1-4 (top), 2-3 (bottom).

To ensure disturbance free inflows for the tanks and maintain the flow ratios for each individual tank, $q_1 = \gamma_a q_a^{max}$, $q_2 = \gamma_b q_b^{max}$, $q_3 = (1-\gamma_a) q_a^{max}$, $q_4 = (1-\gamma_b) q_b^{max}$, we have implemented PI controllers on each of the four inlet valves, thus replicating actual three-way valves.

B. Parallel implementation

In this section we underline the benefits of Algorithm PCDM when it is implemented in an appropriate parallel fashion. Due to the limitations in both hardware and programming language, a proper parallel implementation of algorithm PCDM is impossible on the S7-1200. Therefore, we implemented for comparison, algorithm PCDM and that of [15] for the quadruple tank MPC scheme, corresponding to the first set point, as described in the previous section. Both

algorithms were implemented in C programming language, with parallelization ensured via MPI and linear algebra operations done with CLAPACK. Algorithm [15] 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 [17]. There are no computational drawbacks this time, the algorithms being implemented on a Ubuntu Linux platform, with 2 Intel Xeon E5310 CPUs at 1.60 GHz and 4Gb of RAM. Figure 3 outlines a comparison of the two algorithms for solving the quadruple tank MPC problem, for different prediction horizons N and available time τ in seconds, such that $\tau N = 150$ seconds, i.e. for $\tau = 0.1$ we have $N = 1500$. The bar values represent the total costs over 50 simulation steps of the MPC scheme. For the same 50 simulation steps, we outline in Table IV a comparison of the average number of iterations achieved by both algorithms, in the time τ. Note that our total cost is usually better than that of [15] when the available time is short. For a time $\tau \geq 2$, both algorithms solve the problem exactly. The times for $\tau < 2$ indicate the fact there is a performance loss in the MPC scheme, expressed as percentages in the same table. Also note that, due to its low complexity iteration, our algorithm performs more than ten times the amount of iterations than the algorithm from [15].

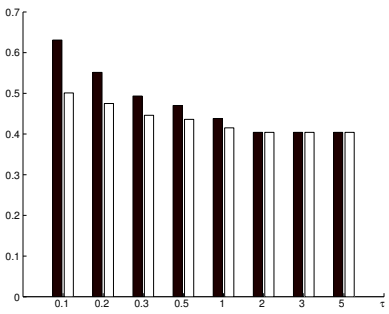

Fig. 3: Evolution of the total MPC costs for 50 MPC steps with PCDM (white) and [15] (black), for different times τ.

[15].

		PCDM	[15]
$\tau(s)$	N	Iter / Perf. Loss (%)	Iter / Perf. Loss (%)
0.1	1500	7 / 24	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 IV: Number of iterations and performance loss, for different times τ.

V. Conclusions

In this paper we have proposed a parallel optimization algorithm to solve MPC problems 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 uses only 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. Implementation results show that this algorithm is suitable both for hardware with low computational power, such as the S7-1200 PLC, and for hardware which permits parallel computations.

Acknowledgements: This research has received funding from: European Union (FP7/2007–2013) EMBOCON 248940; CNCS (project TE, no. 19/11.08.2010); ANCS (project PN II, no. 80EU/2010); POSDRU/89/1.5/S/62557.

References

[1] I. Necoara, V. Nedelcu and I. Dumitrache, *Parallel and distributed optimization methods for estimation and control in networks*, Journal of Process Control, 21(5), 756-766, 2011.

[2] E. Camponogara, D. Jia, B.H Krogh and S. Talukdar, *Distributed model predictive control*, IEEE Control Sys. Magazine, 44-52, 2002.

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

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

[5] J.L. Jerez, G.A. Constantinides and E.C. Kerrigan, *FPGA Implementation of an Interior Point Solver for Linear Model Predictive Control*, in the Proceeding of the IEEE Symposium on Field-Programmable Technology, 316-319, 2010.

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

[7] J. Liu, D. Munoz de la Pena and P.D Christofides, *Distributed model predictive control of nonlinear process systems*, AIChE Journal, 55(5), 1171-1184, 2009.

[8] I. Necoara and D. Clipici, *Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: Application to distributed MPC*, Journal of Process Control, 23(3), 243-253, 2013.

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

[10] I. Necoara, V. Nedelcu, *Inexact dual gradient methods with guaranteed primal feasibility: application to distributed MPC,* Tech. Rep., Univ. Politehnica Bucharest, 2012, `www.optimization-online.org/DB_HTML/2012/10/3627.html` .

[11] Y. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM J. Optim, 22(2), 341-362, 2012.

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

[13] R. Scattolini, *Architectures for distributed and hierarchical Model Predictive Control – A review*, J. Proc. Control, 19(5), 723-731, 2009.

[14] 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.

[15] 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.

[16] 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.

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

[18] P. Zometa, M. Kogel, T. Faulwasser and R. Findeisen, *Implementation Aspects of Model Predictive Control for Embedded Systems*, in the Proceedings of American Control Conference, 2012.