Distributed Model Predictive Control With Hierarchical Architecture For Communication: Application In Automated Irrigation Channels

Alireza Farhadi and Ali Khodabandehlou

A. Farhadi is Assistant Professor in the Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran. Email: afarhadi@sharif.edu.

A. Khodabandehlou was a M.Sc. student in the Department of Electrical Engineering, Sharif University of Technology.

This work is supported by the research office of Sharif University of Technology.

Abstract

This paper is concerned with a distributed model predictive control method that is based on a distributed optimization method with two-level architecture for communication. Feasibility (constraints satisfaction by the approximated solution), convergence and optimality of this distributed optimization method are mathematically proved. For an automated irrigation channel, the satisfactory performance of the proposed distributed model predictive control method in attenuation of the undesired upstream transient error propagation and amplification phenomenon is illustrated and compared with the performance of another distributed model predictive control method that exploits a single-level architecture for communication. It is illustrated that the distributed model predictive control that exploits a two-level architecture for communication has a better performance by better managing communication overhead.

Keywords- Distributed model predictive control, large-scale linear systems, networked control system optimization.

I. Introduction

A. Motivation and Background

In large-scale systems, the total number of constraints and decision variables can be very large. In many cases this means the computation overhead (the time spent computing the optimal solution) using centralized model predictive control methods at each receding horizon may not be practical. Towards overcoming this computational scalability problem, in [1] a Distributed Model Predictive Control (DMPC) method for solving a constrained linear quadratic optimal control problem with single-level architecture for communication is proposed which exploits the computational power often available at each sub-system in the network. This distributed control method consists of two steps: 1) Initialization and 2) Iterated (parallel) computation and communication for exchanging updates of components of the overall decision variable between distributed computing resources.

To provide scope for managing the communication overheads, the authors of [1] proposed in [2] a distributed optimization method that exploits a hierarchical (two-level) architecture for communication (see Fig.1) and a three-step algorithm including an extra outer iterate step. The distributed decision makers are grouped into q disjoint (non-overlapping) neighborhoods. Exchange of information between decision makers within a neighborhood occurs after each update, whereas the exchange of information between neighborhoods is limited to be less frequent. Within a neighborhood, each decision maker frequently updates its local component of the overall decision variable by solving an optimization problem of reduced size. The updated value is then communicated to all other neighboring decision makers. This intra-neighborhood update and communication is referred as an inner iterate. In addition to inner iterates, updates of decision variables from other neighborhoods are received periodically. These are referred to as outer iterates. Between outer iterates, distributed decision makers continue to compute and refine the local approximation of the optimal solution, with fixed values for decision variables from outside the neighborhood. In [1] the authors mathematically proved feasibility (constraints satisfaction by the approximated solution), convergence and optimality of the two-step algorithm. However, in [2] the authors assumed these properties for the three-step optimization algorithm, and they did not provide any mathematical proofs for feasibility, convergence and optimality under the hierarchical exchange of updates.

B. Paper Contributions

This paper aims to further develop the results of [2] by providing mathematical proofs for feasibility, convergence and optimality of the distributed optimization method presented therein [2] that exploits a two-level architecture for communication. Then, a distributed model predictive control method which is based on this distributed optimization method is proposed. This method is applied to automated irrigation channel; and the satisfactory performance of this distributed model predictive control method in attenuation of the undesired upstream transient error propagation and amplification phenomenon is illustrated and compared with the performance of the distributed model predictive control method of [1] that exploits a single-level architecture for communication. It is illustrated that the proposed distributed model predictive control method that exploits a two-level architecture for communication has a better performance by better managing communication overhead.

The literature on the subject of distributed model predictive control is quite rich [3]-[13]. In [4] the authors designed a Lyapunov - based distributed model predictive control method for nonlinear systems that take asynchronous measurements and delays into account. In [5] the authors designed an iterative Lyapunov - based distributed model predictive control method for large-scale nonlinear systems subject to asynchronous, delayed state feedback. In comparison, the distributed model predictive control method of this paper is an iterative Jacobi- based method that uses a synchronous communication architecture. In [6] the authors proposed an iterative dual decomposition approach for distributed model predictive control of linear systems. The authors also presented a stopping condition to distributed optimization algorithm that keeps the number of iterations as small as possible and gives a feasible, stabilizing solution. There are also many papers that have used distributed model predictive control for irrigation channels, such as [10]-[13]. In [10] the authors presented an accelerated gradient - based distributed model predictive control method for linear systems; and they illustrated a successful application of this method to the power reference tracking problem of an hydro power valley system. An hydro power valley system may consist of several irrigation channels/rivers and lakes and exhibits nonlinear and large-scale dynamics and a globally coupled cost functional that prevents distributed methods to be applied directly. The authors in [10] proposed a linearization and approximation technique that enabled them to apply the developed gradient - based distributed model predictive control method for power reference tracking of hydro power valley systems. In terms of application, the control objective of [10] is different from the objective of this paper, which is the improvement of the transient response of automated irrigation channels that automatically regulate water levels.

[11]-[13] also presented distributed model predictive control methods for water level regulation in irrigation channels for delivering the required amount of water in the right time and place. In [11] the authors presented the use of a serial iterative distributed model predictive control method for water level regulation. However in terms of application, the objective of [11]-[13] is different from the objective of this paper; because, unlike [11]-[13], this paper presents a distributed model predictive control method as the secondary control layer in addition of the existing first control layer, which automates the irrigation channels, to improve the performance of the existing automated irrigation channels. Nevertheless, [11]-[13] use a single -layer control via distributed model predictive methods.

C. Notations

Throughout certain conventions are used: $\mathbf{0}$ denotes the zero vector, $||\cdot||$ the Euclidean norm and $\mathbb{R}$ denotes the set of real numbers. '$\doteq$' means 'by definition is equivalent to', $\mathbb{N}\doteq\{1, 2, 3, ...\}$ and $'$ denotes matrix/vector transpose. $a << b$ means $a > 0$ is much smaller than $b > 0$.

D. Paper Organization

The paper is organized as follows: Section II briefly describes the distributed optimization method of [2], and proofs for feasibility, convergence and optimality of this method is presented in Section III. In Section IV, the distributed model predictive control method that is based on the distributed optimization method of [2] is presented and in Section V the satisfactory performance of this method in attenuation of the undesired upstream transient error propagation and amplification phenomenon in automated irrigation channels is illustrated and compared with the performance of the distributed model predictive control method of [1]. In Section VI the paper is concluded by summarizing the contributions of the paper and direction for future research.

II. Distributed Optimization Method with Hierarchical Architecture for Communication

The distributed optimization method of [2] is concerned with n interacting sub-systems: S_1, S_2, ... , S_n each equipped with a decision maker with limited computational power for solving the following optimization problem

$$\min_{(u_1,...,u_n)} \Big\{ J(\mathbf{g}, u_1, ..., u_n),\ u_i \in \mathcal{U}_i, \forall i \Big\}. \tag{1}$$

Here, g is a collection of known vectors, $J \geq 0$ is a finite-horizon quadratic cost functional of decision variables with horizon length N, for each $i = 1, 2, ..., n$, $u_i \in \mathbb{R}^{Nm_i}$ is the decision

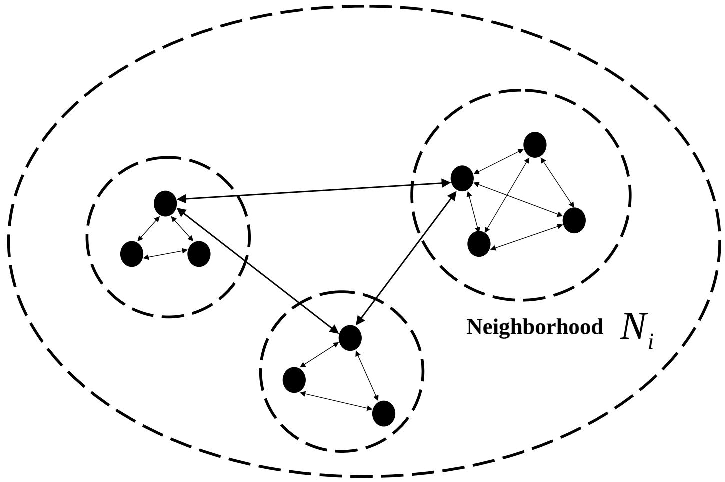

Fig. 1. Two-level architecture for exchanging information between distributed decision makers.

variable associated with sub-system S_i and $\mathcal{U}_i$ is a closed convex subset of the Euclidean space $\mathbb{R}^{Nm_i}$ that includes zero vector.

For the simplicity of presentation, without loss of generality, the dependency of the cost functional J on g is dropped. Throughout, it is assumed that decision makers have knowledge of known parameters described by g and also the expression for the cost functional J in (1). To manage the communication overhead, the distributed optimization method of [2] uses a two-level architecture for exchanging information between distributed decision makers. This communication architecture involves a collection of disjoint neighborhoods of sub-systems (see Fig. 1). In each neighborhood at least one decision maker is selected as the neighborhood cluster head such that all the sub-systems of the neighborhood and also all the sub-systems of the nearest neighboring neighborhood are within the effective communication range of the neighborhood cluster head so that the communication graph between cluster heads is connected. That is, there is a communication path between a cluster head to any other cluster heads. A simple communication architecture is obtained by implementing a Time Division Multiple Access (TDMA)/Orthogonal Frequency Division Multiple Access (OFDMA) scheme[14] as follows: Decision makers in different neighborhoods broadcast their updated decision variables simultaneously without collision using OFDMA scheme; and within a neighborhood, decision makers exchange their updated decision variables without collision using TDMA scheme. The coordination between cluster heads is also achieved by implementing a TDMA scheme as proposed in [15]. This is a synchronous communication. Asynchronous communication is also possible via exchanging flags between decision makers. When a decision maker receives all required information from all other decision makers in its neighborhood, it broadcasts a flag to all other decision makers in its neighborhood to inform them that it is ready to update its decision variable. Also, when this decision maker receives flags of all other decision makers in its neighborhood, it knows that it is time to update its decision variable. Similarly, asynchronous communication between cluster heads is possible via exchanging flags between cluster heads.

Without loss of generality, suppose sub-systems S_1, S_2,...,S_n are distributed into q disjoint neighborhoods, as follows: $\mathcal{N}_1 = \{S_1, .., S_{l_1}\}$, $\mathcal{N}_2 = \{S_{l_1+1}, ..., S_{l_2}\}$, ..., $\mathcal{N}_q = \{S_{l_{q-1}+1}, ..., S_n\}$. Then, the distributed optimization method of [2] approximates the solution of the optimization problem (1) by taking the following three steps:

- *Initialization:* The information exchange between neighborhoods at outer iterate $t \in \{0, 1, 2, ...\}$ makes it possible for every sub-system $\mathcal{S}_i$ to initialize its local decision variable as $h_i^0 = u_i^t \in \mathbb{R}^{Nm_i},\ i \in \{1, ..., n\}$, where $u_i^0 \in \mathcal{U}_i$ are chosen arbitrarily at $t = 0$.
- *Inner Iterate:* Between every two successive outer iterates there are $\bar{p}$ inner iterates. Every sub-system $\mathcal{S}_i$ of the neighborhood $\mathcal{N}_e$ $(e = 1, 2, ..., q)$ performs $\bar{p}$ inner iterates simultaneously with other sub-systems, as follows:

  For each inner iterate $p \in \{0, 1, ..., \bar{p} - 1\}$, sub-system $\mathcal{S}_i$ first updates its decision variable via

  $$h_i^{p+1} = \pi_i h_i^* + (1 - \pi_i) h_i^p, \tag{2}$$

  where π_i are chosen subject to $\pi_i > 0,\ \sum_{j=1}^{l_1} \pi_j = 1,\ \ \ldots\ \ , \sum_{j=l_{q-1}+1}^{n} \pi_j = 1$ and $h_i^* \doteq \text{argmin}_{h_i \in \mathcal{U}_i} J(h_1^0, ..., h_{l_{e-1}}^0, h_{l_{e-1}+1}^p, ..., h_i, ..., h_{l_e}^p, h_{l_e+1}^0, ..., h_n^0)$ (note that $l_0 = 0$, $l_q = n$). Then, it trades its updated decision variable h_i^{p+1} with all other sub-systems in its neighborhood $\mathcal{N}_e$.
- *Outer Iterate:* After $\bar{p}$ inner iterates, there is an outer iterate update as follows:

  $$u_i^{t+1} = \lambda_i h_i^{\bar{p}} + (1 - \lambda_i) u_i^t, \tag{3}$$

  where $u_i^t = \begin{pmatrix} u_i^{t\prime}[0] & u_i^{t\prime}[1] & . & . & . & u_i^{t\prime}[N-1] \end{pmatrix}' \in \mathbb{R}^{Nm_i}$, $u_i^t[j] \in \mathbb{R}^{m_i}$, $j = 0, 1, 2, ..., N-1$, and λ_i, $i = 1, 2, ..., n$, are chosen subject to $\lambda_i > 0,\ \lambda_1 = ... = \lambda_{l_1},\ \lambda_{l_1+1} = ... = \lambda_{l_2},\ ..., \lambda_{l_{q-1}+1} = ... = \lambda_{l_q}$ $(\lambda_{l_q} = \lambda_n)$, $\lambda_{l_1} + \lambda_{l_2} + ... + \lambda_{l_q} = 1$. Then, there is an outer iterate communication, in which the updated decision variables u_i^{t+1} are shared between all neighborhoods; and subsequently, between all sub-systems.

As will be shown in the next section, by increasing t, $(u_1^t, u_2^t, ..., u_n^t)$ converges to $(u_1^*, u_2^*, ..., u_n^*)$, which is the optimal solution of the optimization problem (1). Hence, for a large enough $\bar{t}$, $(u_1^{\bar{t}}, u_2^{\bar{t}}, ..., u_n^{\bar{t}})$ is an approximation of the optimal solution.

We now define communication overhead, computation overhead and computational latency of the above three-step algorithm as follows.

Definition 2.1: (Communication Overhead) Communication overhead is defined as the total time spent for exchanging information between distributed decision makers for approximating the solution of the optimization problem (1) by $u_i^{\bar{t}}$.

Definition 2.2: (Computation Overhead) At a given inner iteration define the decision maker with the longest processing time as the dominating decision maker at this iteration. Then, computation overhead is defined as the summation of the processing times of the dominated decision maker for approximating the solution of the optimization problem (1) by $u_i^{\bar{t}}$.

Definition 2.3: (Computational Latency) Computational latency is the summation of the communication overhead and computation overhead.

Remark 2.4: For the one neighborhood case (i.e., $q = 1, \bar{p} = 1$), the above three-step algorithm is reduced to the two-step algorithm of [1], in which it only involves initialization step and inner iterate updates (2) with $p = t$ followed by outer iterate communication.

III. Feasibility, Convergence and Optimality Results

In this section, it is shown that given a feasible initialization (i.e., $u_i^0 \in \mathcal{U}_i$), the iterates (3) are feasible (i.e., $u_i^t \in \mathcal{U}_i$, $t \in \{0, 1, 2, ...\}$), the cost functional is non-increasing for each outer iterate (and so converges as $t \to \infty$), and the iterates $(u_1^t, ..., u_n^t)$ converge to the optimal solution $(u_1^*, ..., u_n^*)$ of the constrained optimization problem (1). Note that as $J \geq 0$ is quadratic and the constraint sets are convex, there exists a unique optimal solution $(u_1^*, ..., u_n^*)$. Feasibility and convergence properties are shown for a general convex finite-horizon cost functional $J(u_1, ..., u_n)$; however, for optimality it is also assumed that the cost functional is quadratic.

Feasibility, convergence and optimality proofs presented here closely follow those given in [1]. The changes are required to develop new proofs are as follows:

- For feasibility proof, first we need to prove feasibility of inner iterates, and then feasibility of outer iterates as proved in [1].

- For convergence proof, we need to show that the cost functional is non-increasing at each inner iterate between each two successive outer iterates in addition of showing that the cost functional is non-increasing at each outer iterate as proved in [1] .

- For optimality proof, as we consider the general case with n sub-systems subject to inner and outer iterates update and communication, a more detailed and complicated proof must be presented compared with the proof presented in [1] for optimality, which is concerned with a system with only two sub-systems subject to outer iterates update and communication.

Proposition 3.1: (Feasibility) Given the above convex finite-horizon cost functional J, convex control constraint sets $\mathcal{U}_i$ and a feasible initialization, the inner and outer iterates (2) and (3) are feasible.

Proof: By assumption, the initialization, $h_i^0 = u_i^0$ is feasible. Since $\mathcal{U}_1$, ..., $\mathcal{U}_n$ are convex, the convex combination (2) with $p = 0$ implies that $(h_1^1, ..., h_n^1)$ is feasible. Feasibility for $p \in \{1, ..., \bar{p} - 1\}$ follows similarly by induction. Now as u_i^0 and $h_i^{\bar{p}}$ are feasible, the convex combination (3) with $t = 0$ implies that $(u_1^1, ..., u_n^1)$ is feasible. Subsequently, the feasibility for $t > 1$ and each $p \in \{0, 1, ..., \bar{p}-1\}$ between every two successive outer iterates follows similarly.

Next we show the convergence of the cost functional J under the solution (3).

Proposition 3.2: (Convergence) Given a feasible initialization, convex finite-horizon cost functional $J(u_1^t, ..., u_n^t)$ is non-increasing at each outer iterate $t \in \{0, 1, 2, ...\}$ and converges as $t \to \infty$.

Proof: For each $t \in \{0, 1, 2, ...\}$, the cost functional satisfies the following:

$$
\begin{aligned}
&J(u_1^{t+1}, ..., u_n^{t+1}) \\
&= J\Big(\lambda_{l_1}(h_1^{\bar{p}}, ..., h_{l_1}^{\bar{p}}, u_{l_1+1}^t, ..., u_n^t) + ... + \lambda_{l_m}(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}}, ..., h_{l_m}^{\bar{p}}, u_{l_m+1}^t, ..., u_n^t) \\
&+ ... + \lambda_{l_q}(u_1^t, ..., u_{l_{q-1}}^t, h_{l_{q-1}+1}^{\bar{p}}, ..., h_n^{\bar{p}})\Big) \\
&\leq \lambda_{l_1} J(h_1^{\bar{p}}, ..., h_{l_1}^{\bar{p}}, u_{l_1+1}^t, ..., u_n^t) + ... + \lambda_{l_m} J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}}, ..., h_{l_m}^{\bar{p}}, u_{l_m+1}^t, ..., u_n^t) \\
&+ ... + \lambda_{l_q} J(u_1^t, ..., u_{l_{q-1}}^t, h_{l_{q-1}+1}^{\bar{p}}, ..., h_n^{\bar{p}}), \qquad (4)
\end{aligned}
$$

where the equality follows from (3), and the inequality follows from the convexity of the cost functional. Now, define

$$
J_m \doteq J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}}, ..., h_{l_m}^{\bar{p}}, u_{l_m+1}^t, ..., u_n^t), \quad m \in \{1, 2, ..., q\}, \qquad (5)
$$

whereby it is understood that $J_1 = J(h_1^{\bar{p}}, ..., h_{l_1}^{\bar{p}}, u_{l_1+1}^t, ..., u_n^t)$ and $J_q = J(u_1^t, ..., u_{l_{q-1}}^t, h_{l_{q-1}+1}^{\bar{p}}, ..., h_n^{\bar{p}})$. Note that $l_q = n$.

Then, J_m satisfies the following bound

$$\begin{aligned}
&J_m = J\Big(u_1^t, ..., u_{l_{m-1}}^t, \pi_{l_{m-1}+1}(h_{l_{m-1}+1}^*, h_{l_{m-1}+2}^{\bar{p}-1}, ..., h_{l_m}^{\bar{p}-1}) + ...\\
&+\pi_{l_m}(h_{l_{m-1}+1}^{\bar{p}-1}, ..., h_{l_m-1}^{\bar{p}-1}, h_{l_m}^*), u_{l_m+1}^t, ..., u_n^t\Big)\\
&\leq \pi_{l_{m-1}+1} J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^*, h_{l_{m-1}+2}^{\bar{p}-1}, ..., h_{l_m}^{\bar{p}-1}, u_{l_m+1}^t, ..., u_n^t) + ...\\
&+\pi_{l_m} J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}-1}, ..., h_{l_m-1}^{\bar{p}-1}, h_{l_m}^*, u_{l_m+1}^t, ..., u_n^t)\\
&\leq \Big(\sum_{j=l_{m-1}+1}^{l_m} \pi_j\Big) J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}-1}, ..., h_{l_m}^{\bar{p}-1}, u_{l_m+1}^t, ..., u_n^t)\\
&= J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}-1}, ..., h_{l_m}^{\bar{p}-1}, u_{l_m+1}^t, ..., u_n^t),
\end{aligned}$$

where $h_{l_{m-1}+1}^*$, $h_{l_{m-1}+2}^*$, $...h_{l_m}^*$ have been generated at inner iterate $\bar{p}$, the first equality follows from (2) for $p = \bar{p}$, the first inequality follows from the convexity of the cost functional, the second inequality follows from the fact that the cost functional J for h_j^*, $j = l_{m-1} + 1$,...,l_m, is not greater than J for $h_j^{\bar{p}-1}$, and the second equality follows from the fact that $\sum_{j=l_{m-1}+1}^{l_m} \pi_j = 1$. By following a similar argument, it can be shown for $m \in \{1, 2, ..., q\}$ that

$$\begin{aligned}
J_m &\leq J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}-1}, ..., h_m^{\bar{p}-1}, u_{l_m+1}^t, ..., u_n^t)\\
&\leq J(u_1^t, ..., u_{l_{m-1}}^t, h_{l_{m-1}+1}^{\bar{p}-2}, ..., h_{l_m}^{\bar{p}-2}, u_{l_m+1}^t, ..., u_n^t) \leq ... \leq J(u_1^t, ..., u_n^t). \qquad (6)
\end{aligned}$$

Consequently, from (4), (6) it follows that

$$J(u_1^{t+1}, ..., u_n^{t+1}) \leq \Big(\lambda_{l_1} + \lambda_{l_2} + ... + \lambda_{l_q}\Big) J(u_1^t, ..., u_n^t) = J(u_1^t, ..., u_n^t).$$

That is, the cost $J(u_1^t, ..., u_n^t)$ is non-increasing at each outer iterate t. Hence, the non-negative cost functional J converges as $t \to \infty$ by the monotone convergence theorem [16].

Now, in the following proposition using the contradiction argument we show that the convergent point $\bar{J}$ is the optimal value of the cost functional, i.e., $\bar{J} = J(u_1^*, ..., u_n^*)$; and the iterates $(u_1^t, ..., u_n^t)$ converge to the unique optimal solution $(u_1^*, ..., u_n^*)$, as $t \to \infty$.

Proposition 3.3: (Optimality) Given a feasible initialization, strictly convex and quadratic cost J, and closed convex control constraint sets $\mathcal{U}_i$, the cost $J(u_1^t, ..., u_n^t)$ converges to the optimal cost $J(u_1^*, ..., u_n^*)$, and the iterates $(u_1^t, ..., u_n^t)$ converge to the unique optimal solution $(u_1^*, ..., u_n^*)$, as $t \to \infty$.

Proof: For the clarity of the proof we first present the sketch of the proof: From convergence result, it follows that the cost converges to some $\bar{J} \geq 0$ and all iterates belong to a sequentially compact set. Hence, there must exist one sub-sequence $(u_1^t, ..., u_n^t)_{t \in \mathcal{T}}$, $\mathcal{T} \subset \{1, 2, 3, ...\}$ and an accumulation point $(\bar{u}_1, ..., \bar{u}_n)$ such that this sub-sequence converges to this point. Then, using a contradiction argument it is shown that this accumulation point must be the optimal solution. Now, as this point is an arbitrary accumulation point and $(u_1^t, ..., u_n^t)_{t\in\mathcal{T}}$ is also an arbitrary convergent sub-sequence of the sequence $(u_1^t, ..., u_n^t)_{t\geq 0}$, from the above analysis, it is concluded that every convergent sub-sequence of the sequence $(u_1^t, ..., u_n^t)_{t\geq 0}$ converges to the same limit, which is the optimal solution. Therefore, it must be the case that the entire sequence $(u_1^t, ..., u_n^t)_{t\geq 0}$ with the property of $\lim_{t\to\infty} J(u_1^t, ..., u_n^t) = \bar{J}$ which is made of convergent sub-sequences, converges to the optimal solution.

Now, the details of the proof are as follows: From Proposition 3.2, it follows that the cost converges to some $\bar{J} \geq 0$. Because J is quadratic and strictly convex, its sub level sets $Lev_{\leq a}(J)$ are compact and bounded for all $a \geq 0$. Therefore, all iterates belong to the compact and bounded set $Lev_{\leq J(u_1^0,...,u_n^0)}(J) \cap \mathcal{U}_1 \times ... \times \mathcal{U}_n$. Hence, there is at least a sub-sequence $\mathcal{T} \subset \{1, 2, 3, ...\}$ and one accumulation point $(\bar{u}_1, ..., \bar{u}_n)$ such that $(u_1^t, ..., u_n^t)_{t\in\mathcal{T}}$ converge to $(\bar{u}_1, ..., \bar{u}_n)$ and $\lim_{t\in\mathcal{T}, t\to\infty} J(u_1^t, ..., u_n^t) = J(\bar{u}_1, ..., \bar{u}_n) = \bar{J}$.

Suppose for the purpose of contradiction that $\bar{J} \neq J(u_1^*, ..., u_n^*)$, and therefore $(\bar{u}_1, ..., \bar{u}_n) \neq (u_1^*, ..., u_n^*)$. Because $J(.)$ is convex, we have:

$$\nabla J(\bar{u}_1, ..., \bar{u}_n)'(U^* - \bar{U}) \leq \Delta J \doteq J(u_1^*, ..., u_n^*) - J(\bar{u}_1, ..., \bar{u}_n), \tag{7}$$

where $U^* = (\, u_1^{*'} \quad . \quad . \quad . \quad u_n^{*'}\,)'$ and $\bar{U} = (\, \bar{u}_1' \quad . \quad . \quad . \quad \bar{u}_n{}'\,)'$. ∇J can be partitioned as follows: $\nabla J(\bar{u}_1, ..., \bar{u}_n) = (\, \nabla_{u_1} J(\bar{u}_1, ..., \bar{u}_n)' \quad . \quad . \quad . \quad \nabla_{u_n} J(\bar{u}_1, ..., \bar{u}_n)'\,)'$. Then, from the inequality (7) it follows that at least one of $\nabla_{u_i} J(\bar{u}_1, ..., \bar{u}_n)' \times (u_i^* - \bar{u}_i)$ must be less than or equal to $\frac{\Delta J}{n}$. For simplicity, without loss of generality, suppose $i = 1$. Then, by applying Taylor's theorem to $J(u_1^t + \epsilon(u_1^* - u_1^t), u_2^t, ..., u_n^t)$, $\epsilon > 0$, we have:

$$J(u_1^t + \epsilon(u_1^* - u_1^t), u_2^t, ..., u_n^t) = J(u_1^t, ..., u_n^t) + \epsilon \nabla J(u_1^t, .., u_n^t)' \begin{pmatrix} u_1^* - u_1^t \\ \mathbf{0} \\ . \\ . \\ . \\ \mathbf{0} \end{pmatrix} + \mathcal{O}(\epsilon^2). \tag{8}$$

By the definition of the convergence of $J(u_1^t, ..., u_n^t)$ and $(u_1^t, ..., u_n^t)_{t\in\mathcal{T}}$ there exists a $t_1 \in \mathcal{T}$ such that for all $t \in \mathcal{T}$ so that $t \geq t_1$, the difference between $J(u_1^t, ..., u_n^t)$ and $J(\bar{u}_1, ..., \bar{u}_n)$ and also the difference between $(u_1^t, ..., u_n^t)$ and $(\bar{u}_1, ..., \bar{u}_n)$ are negligible. Hence, as $\nabla_{u_1} J(\bar{u}_1, ..., \bar{u}_n)' \times$

$(u_1^* - \bar{u}_1) \leq \frac{\Delta J}{n}$, for all $t \geq t_1$, we have the following inequality:

$$J(u_1^t,...,u_n^t) + \epsilon \nabla J(u_1^t,..,u_n^t)' \begin{pmatrix} u_1^* - u_1^t \\ \mathbf{0} \\ . \\ . \\ . \\ \mathbf{0} \end{pmatrix} + \mathcal{O}(\epsilon^2) \leq J(\bar{u}_1,...,\bar{u}_n) + \frac{\epsilon \Delta J}{n} + \mathcal{O}(\epsilon^2). \qquad (9)$$

Now, from (4) and (6) it follows that

$$J(u_1^{t+1}, u_2^{t+1}, ..., u_n^{t+1}) \leq \lambda_{l_1} J_1 + \lambda_{l_2} J_2 + ... + \lambda_{l_q} J_q \leq J(u_1^t, u_2^t, ..., u_n^t),$$

where J_m, $m = 1, 2, ..., q$, are given in (5). From these inequalities and as $\lim_{t\in\mathcal{T},t\to\infty} J(u_1^{t+1}, u_2^{t+1},$ $..., u_n^{t+1}) = \bar{J}$ and $\lim_{t\in\mathcal{T},t\to\infty} J(u_1^t, u_2^t, ..., u_n^t) = \bar{J}$, it follows that

$$\lim_{t\in\mathcal{T},t\to\infty} (\lambda_{l_1} J_1 + \lambda_{l_2} J_2 + ... + \lambda_{l_q} J_q) = \bar{J}. \qquad (10)$$

From (6) and the definition of the convergence of $J(u_1^t, ..., u_n^t)$ it follows that for all $t \in \mathcal{T}$ so that $t \geq t_1$, we have the following inequalities: $J_m \leq \bar{J}$, $\forall m \in \{1, 2, ..., q\}$. This means that for each m there exists an $\epsilon_m \geq 0$ such that $\lim_{t\in\mathcal{T},t\to\infty} J_m = \bar{J} - \epsilon_m$. Hence, from (10) it follows that $\sum_{m=1}^{q} \lim_{t\in\mathcal{T},t\to\infty} \lambda_{l_m} J_m (= \bar{J} - \sum_{m=1}^{q} \lambda_{l_m} \epsilon_m) = \bar{J}$. From this equality and as $\lambda_{l_m} > 0$ it follows that $\epsilon_m = 0$, $\forall m \in \{1, 2, ..., q\}$ giving the following result: $\lim_{t\in\mathcal{T},t\to\infty} J_m = \bar{J}$, $\forall m \in \{1, 2, ..., m\}$. This result combined with (6) for $m = 1$ similarly results in the following:

$$\lim_{t\in\mathcal{T},t\to\infty} J(h_1^p, h_2^p, ..., h_{l_1}^p, u_{l_1+1}^t, ..., u_n^t) = \bar{J}, \ \forall p \in \{1, 2, ..., \bar{p}\}. \qquad (11)$$

Now, convexity of $J(.)$ results in the following inequality:

$$\begin{aligned} J(h_1^1, h_2^1, ..., h_{l_1}^1, u_{l_1+1}^t, ..., u_n^t) &= J(\pi_1(h_1^*, u_2^t, ..., u_{l_1}^t) + \pi_2(u_1^t, h_2^*, ..., u_{l_1}^t) + ... \\ &\quad +\pi_{l_1}(u_1^t, u_2^t, ..., h_{l_1}^*), u_{l_1+1}^t, ..., u_n^t) \\ &\leq \pi_1 J(h_1^*, u_2^t, ..., u_n^t) + \pi_2 J(u_1^t, h_2^*, ..., u_n^t) + ... \\ &\quad +\pi_{l_1} J(u_1^t, ..., h_{l_1}^*, u_{l_1+1}^t, ..., u_n^t) \\ &\leq (\pi_1 + \pi_2 + ... + \pi_{l_1}) J(u_1^t, ..., u_n^t) = J(u_1^t, ..., u_n^t), \end{aligned}$$

where $h_1^* = \text{argmin}_{h_1\in\mathcal{U}_1} J(h_1, u_2^t, ..., u_n^t)$ and h_2^*,...,$h_{l_1}^*$ are defined similarly. Consequently, from (11) and as $\pi_1, \pi_2, ..., \pi_{l_1} > 0$, it follows from a similar argument that

$$\lim_{t\in\mathcal{T},t\to\infty} J(h_1^*, u_2^t, ..., u_n^t) = \bar{J}. \qquad (12)$$

Now, consider the left hand side of (8). As $\mathcal{U}_1$ is a convex set and $u_1^t + \epsilon(u_1^* - u_1^t) = (1-\epsilon)u_1^t + \epsilon u_1^*$ is a convex combination of $u_1^t, u_1^* \in \mathcal{U}_1$ for $\epsilon \in (0,1]$, it follows for $\epsilon \in (0,1]$ that $u_1^t + \epsilon(u_1^* - u_1^t) \in \mathcal{U}_1$. Hence, as $h_1^* = \text{argmin}_{h_1 \in \mathcal{U}_1} J(h_1, u_2^t, ..., u_n^t)$, it is evident for $\epsilon \in (0,1]$ that $J(h_1^*, u_2^t, ..., u_n^t) \leq J(u_1^t + \epsilon(u_1^* - u_1^t), u_2^t, ..., u_n^t)$. Consequently, from (12) for all $t \in \mathcal{T}$ so that $t \geq t_1$ the following inequality holds:

$$\bar{J} \leq J(u_1^t + \epsilon(u_1^* - u_1^t), u_2^t, ..., u_n^t). \tag{13}$$

Hence, from (8), (9) and (13), for sufficiently small $\epsilon > 0$ and all $t \in \mathcal{T}$ so that $t \geq t_1$, it follows that $\bar{J} \leq \bar{J} + \frac{\epsilon \Delta J}{n}$. But, as $\Delta J < 0$, from the inequality $\bar{J} \leq \bar{J} + \frac{\epsilon \Delta J}{n}$ it follows that $\bar{J} < \bar{J}$ giving a contradiction. Therefore, by contradiction we have $J(\bar{u}_1, ..., \bar{u}_n) = \bar{J} = J(u_1^*, ..., u_n^*)$ and $(\bar{u}_1, ..., \bar{u}_n) = (u_1^*, ..., u_n^*)$. Now, as $(\bar{u}_1, ..., \bar{u}_n)$ was an arbitrary accumulation point and $(u_1^t, ..., u_n^t)_{t \in \mathcal{T}}$ was an arbitrary convergent sub-sequence of the sequence $(u_1^t, ..., u_n^t)_{t \geq 0}$, from the above analysis it is concluded that every convergent sub-sequence of the sequence $(u_1^t, ..., u_n^t)_{t \geq 0}$ converges to the same limit of $(u_1^*, ..., u_n^*)$. Therefore, it must be the case that the entire sequence $(u_1^t, ..., u_n^t)_{t \geq 0}$ with the property of $\lim_{t \to \infty} J(u_1^t, ..., u_n^t) = \bar{J}$ which is made of convergent sub-sequences, converges to the optimal solution $(u_1^*, ..., u_n^*)$.

IV. Distributed Model Predictive Control Method

Because the distributed optimization method of previous sections is concerned with a convex optimization problem with quadratic cost functional of decision variables, in this section we propose a distributed model predictive control method with linear dynamics, convex constraint sets and quadratic cost functional. As will be shown in this section this distributed model predictive control problem can be written in terms of a convex optimization problem with a quadratic cost; and hence, the distributed optimization method of previous sections can be used to solve it.

This section is concerned with a dynamic system with n distributed interacting linear time invariant sub-systems S_i, $i = 1, 2, ..., n$, of the following form, in which each of them is equipped with a decision maker that generates the decision variable u_i.

$$S_i : \begin{cases} x_i[k+1] = A_i x_i[k] + B_i u_i[k] + \sum_{j=1, j \neq i}^{n} M_j x_j[k] + N_j u_j[k], \\ y_i[k] = C_i x_i[k], \\ z_i[k] = D_i x_i[k], \\ k = \{0, 1, 2, ...\} \text{is the time instant.} \end{cases} \tag{14}$$

For the system (14) we are concerned with the Linear Quadratic (LQ) constrained optimal control problem (15) subject to the dynamics of sub-systems (14) and the operational constraints

$x_i[k] \in \mathcal{X}_i$ and $u_i[k] \in \mathcal{G}_i$, where $\mathcal{X}_i$ is a closed convex subset of the real Euclidean space with dimension $n_i > 0$ (i.e., $\mathcal{X}_i \subset \mathbb{R}^{n_i}$) modeling constraint set on the ith state variable; and $\mathcal{G}_i$ is a closed convex subset of $\mathbb{R}^{m_i}$ modeling constraint set on the ith decision variable.

$$\min_{(u_1,u_2,...,u_n)} \{J_L(x[0], r, u_1, ..., u_n), u_i[k] \in \mathcal{G}_i, x_i[k] \in \mathcal{X}_i, \forall i, k, \text{subject to}(14)\} \tag{15}$$

$$J_L(x[0], r, u_1, ..., u_n) = \sum_{i=1}^{n} \sum_{k=0}^{L-1} ||y_i[k] - r_i||_Q^2 + ||u_i[k] - u_i[k-1]||_R^2 + ||z_i[k]||_P^2, \ (u_i[-1] = \mathbf{0}), \tag{16}$$

where $r \doteq (\, r_1' \quad r_2' \quad . \quad . \quad . \quad r_n' \,)'$, r_i is the desired value for the output y_i,

$$x[0] \doteq (\, x_1'[0] \quad x_2'[0] \quad . \quad . \quad . \quad x_n'[0] \,)'$$

is the initial state vector, $|| \cdot ||$ is the Euclidean norm and $Q = Q' \geq 0$, $R = R' > 0$ and $P = P' \geq 0$ are weighting matrices.

Remark 4.1: As the optimal control problem (15) is a constrained problem and the horizon length L is long, the receding horizon idea must be used to solve this problem. That is, at each time instant k the following associated optimal control problem with the cost functional (17) with the horizon length $N << L$ must be solved.

$$\min_{(u_1,u_2,...,u_n)} \{J(x[k], r, u_1, ..., u_n), u_i[j] \in \mathcal{G}_i, x_i[j] \in \mathcal{X}_i, \forall i, j = k, ..., k+N-1, \text{subject to}(14)\}$$

$$J(x[k], r, u_1, u_2, ..., u_n) = \sum_{i=1}^{n} \sum_{j=k}^{k+N-1} ||y_i[j] - r_i||_Q^2 + ||u_i[j] - u_i[j-1]||_R^2 + ||z_i[j]||_P^2. \tag{17}$$

The solution to this optimal control problem is the vectors

$$u_i^* = (\, u_i^{*'}[k] \quad u_i^{*'}[k+1] \quad . \quad . \quad . \quad u_i^{*'}[k+N-1] \,)', i = 1, 2, ..., n,$$

in which only the first components, i.e., $u_i^*[k]$s, are applied by distributed decision makers and this procedure is repeated for the next time instant. Note that in the proposed method at each time instant k, $x_i[k]$, $i = 1, 2, ..., n$, are measured and shared between all decision makers so that at each receding horizon $x_i[k]$s are known to each decision maker.

By expanding the dynamic model (14) in terms of $x[k] \doteq (\, x_1'[k] \quad x_2'[k] \quad . \quad . \quad . \quad x_n'[k] \,)'$ and the decision variables $u_i[j]$s, $j = k, k+1, ..., k+N-1$, and substituting $x_i[j]$, $y_i[j]$ and $z_i[j]$ rewritten in terms of $x[k]$ and decision variables $u_i[j]$ in the cost functional (17), it is written as a quadratic function of decision variables. Furthermore, as $\mathcal{X}_i$, $i = 1, 2, ..., n$, are closed convex sets, their Cartesian product $\mathcal{X}_1 \times \mathcal{X}_2 \times ... \times \mathcal{X}_n$ is a closed convex set [17]. Consequently, as affine functions preserves closeness and convexity of sets [17], the closed convex state constraint set $\mathcal{X}_i$

on the time invariant dynamics (14) imposes additional constraint on decision variables. That is, $u_i \in \mathcal{H}_i(x[k])$. Therefore, for each i the set of control constraint for the associated LQ problem with the cost functional (17) is $(u_i \in)\mathcal{G}_i \cap \mathcal{H}_i(x[k])$, where $\mathcal{G}_i \cap \mathcal{H}_i(x[k])$ is a closed convex set. Hence, the optimization problem of the previous sections with $\mathbf{g} = (x[k], r)$ is applicable to the associated LQ problem with the cost functional (17).

Having that, in the proposed distributed model predictive control method, at each receding horizon with horizon length N, the distributed optimization method of previous sections is used with $t = \{0, 1, 2, ..., \bar{t}-1\}$ to solve the associated LQ problem. The initialization of the distributed optimization method at each receding horizon is based on the warm start. That is, at time instant $k+1$, the initialization is as follows

$$u_i^0 = (\, u_i^{\bar{t}'}[k+1] \quad u_i^{\bar{t}'}[k+2] \quad . \quad . \quad . \quad u_i^{\bar{t}'}[k+N-1] \quad \mathbf{0}' \,)',$$

where $u_i^{\bar{t}}[k+j] \in \mathbb{R}^{m_i}$, $j = 0, 1, 2, ..., N-1$, are the solution of the distributed optimization problem at the previous time instant k. Note that at time instant $k = 0$, $u_i^0 \in \mathcal{U}_i$, $i = 1, 2, ..., n$, are chosen arbitrary. Then, for each time instant k the three-step optimization algorithm of the previous sections are repeated $\bar{t}$ times until the vectors $u_i^{\bar{t}} = (\, u_i^{\bar{t}'}[k] \quad u_i^{\bar{t}'}[k+1] \quad . \quad . \quad . \quad u_i^{\bar{t}'}[k+N-1] \,)'$, $i = 1, 2, ..., n$, are generated, in which only the first components $u_i^{\bar{t}}[k]$s are applied to the distributed dynamic system by distributed decision makers.

Remark 4.2: To guarantee the recursive feasibility, the proper terminal constraints can be included to each associated LQ problem with horizon length N [18].

V. Automated Irrigation Channels and Undesired Upstream Transient Error Propagation and Amplification Phenomenon

To illustrate the satisfactory performance of the above distributed model predictive control method with two-level architecture for communication, this method is applied in this section to an automated irrigation channel and its satisfactory performance in attenuating the undesired upstream transient error propagation and amplification phenomenon is illustrated and compared with the performance of the distributed model predictive control method of [1] that exploits a single-level architecture for communication.

A. An Automated Irrigation Channel

An automated irrigation channel consists of a collection of interconnected pools (see Fig 2). Each pool is equipped with an overshot gate, a modem for wireless communication, sensors,

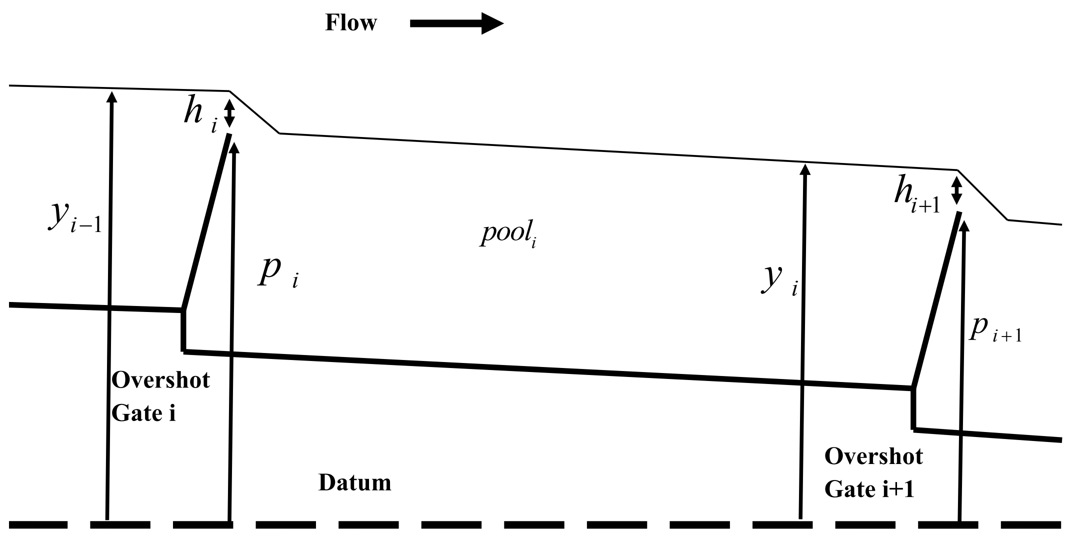

Fig. 2. An automated irrigation channel.

actuators and a processing device (decision maker). Open irrigation channels have been traditionally modeled by the St. Venant equations which are nonlinear hyperbolic partial differential equations [19]. However, as shown in [20], [21] around desired set points open irrigation channels can be modeled with high accuracy by linear dynamics using mass-balance principle and system identification techniques. Because in this section the objective is to maintain the downstream water levels of open irrigation channels pools around desired set points with small variation, linear model is used for open irrigation channels.

The dynamics of each sub-system (pool) in an automated irrigation channel in continuous time domain is described, as follows (see Fig. 2)[22].

$$\begin{aligned}
\dot{y}_i(t) &= C_i^{in} z_i(t-\tau_i) + C_i^{out} z_{i+1}(t) + C_i^{out} d_i(t), \quad i = 1, 2, ..., n, \\
z_i &\doteq h_i^{\frac{3}{2}}, \quad h_i = y_{i-1} - p_i, \quad z_{i+1} \doteq h_{i+1}^{\frac{3}{2}}, \quad h_{i+1} = y_i - p_{i+1}, \\
d_i &= \frac{\bar{d}_i}{\gamma_{i+1}}, \quad d_n = \frac{\bar{d}_n}{\gamma_n},
\end{aligned} \tag{18}$$

where $\alpha_i > 0$ (measured in meter square - m^2) is a constant which depends on the pool surface area, $y_i \geq 0$ (measured in meter Above Height Datum - mAHD) is the downstream water level at the ith pool, $h_i \geq 0$ (measured in meter) is the head over upstream gate (the ith gate), h_{i+1} is the head over downstream gate (the $i+1$th gate), $p_i \geq 0$ (measured in meter) is the position of the ith gate, τ_i (measured in minutes) is the fixed transport delay, $\bar{d}_i \geq 0$ (measured in meter cube per minutes - m^3/min) is the off-take flow rate disturbance taken at the end of pool i by user, and γ_i as well as C_i^{in} (measured in $m^{\frac{-1}{2}}/min$) and C_i^{out} (measured in $m^{-\frac{1}{2}}/min$) are constant.

The equation (18) can be written in terms of the storage (integrator) equation (19) and transport (delay) equation (20), as follows.

$$\dot{y}_i(t) = C_i^{in}\tilde{z}_i(t) + C_i^{out}z_{i+1}(t) + C_i^{out}d_i(t). \tag{19}$$

$$\tilde{z}_i(t) = z_i(t - \tau_i). \tag{20}$$

The storage equation (19) can be directly converted to discrete time model using the zero order hold technique, while the transport equation (20) is converted to discrete time model by introducing $\frac{\tau_i}{T}$ states as follows $x_{i,2}(kT) = z_i(kT - \frac{\tau_i}{T})$, ... , $x_{i,\frac{\tau_i}{T}+1}(kT) = z_i(kT - T)$, where the sampling period T is the biggest common factor of pools transport delays (note that $x_{i,1}(kT) = y_i(kT)$, $k \in \{1, 2, 3, ...\}$). Following the above conversions, the equivalent discrete time model describing the dynamics of the ith sub-system/pool is given by the following discrete time state space model with the state variable $b_i[k]$.

$$\begin{cases} b_i[k+1] = \check{A}_i b_i[k] + \check{B}_i z_i[k] + \check{D}_i z_{i+1}[k] + \check{F}_i d_i[k], \\ \quad y_i[k] = \check{C}_i b_i[k] \qquad\qquad i = 1, 2, ..., n \end{cases}$$

In an automated irrigation channel, PI controllers $z_i(s) = C_i(s)e_i(s)$, $C_i(s) = \frac{K_i T_i s + K_i}{T_i F_i s^2 + T_i s}$, $e_i = u_i - y_i$ are designed to stabilize an automated irrigation channel around the pre-defined reference signals u_is by attenuating the effects of off-take flow disturbances. Now, by finding the corresponding discrete time transfer function $C_i(z)$ and then the corresponding state space representation, we have the following discrete time representation for the PI controllers

$$\begin{cases} \xi_i[k+1] = \bar{A}_i \xi_i[k] + \bar{B}_i e_i[k], \;\; \xi_i[0] = \mathbf{0}, \\ \quad z_i[k] = \bar{C}_i \xi_i[k]. \end{cases}$$

Consequently, by defining the augmented state variable $x_i[k] = \begin{pmatrix} b_i[k] \\ \xi_i[k] \end{pmatrix}$, the dynamics of the automated irrigation network is given by (21).

$$S_i : \begin{cases} x_i[k+1] = A_i x_i[k] + B_i u_i[k] + F_i d_i[k] + v_i[k], \\ \qquad y_i[k] = C_i x_i[k], \\ \qquad z_i[k] = D_i x_i[k], \end{cases} \tag{21}$$

Pool	$C_i^{in}(\frac{m^{-1/2}}{min})$	$C_i^{out}(\frac{m^{-1/2}}{min})$	$\tau_i(min)$	$\gamma_i(\frac{m^3}{min})$	K_i	T_i	F_i
1 (pool 2 of the East Goulburn)	0.01072	-0.01034	36	2330	0.585	539	47.2
2 (pool 5 of the East Goulburn)	0.01169	-0.00833	28	1210	0.679	366	43.4
3 (pool 8 of the East Goulburn)	0.01065	-0.01599	15	351	0.892	355	31.2
4 (pool 9 of the East Goulburn)	0.08457	-0.0853	1	527	1.31	26.1	3.1

TABLE I

NUMERICAL VALUES FOR PARAMETERS DESCRIBING AUTOMATED POOLS 2,5, 8 AND 9 OF THE EAST GOULBURN MAIN IRRIGATION CHANNEL [18].

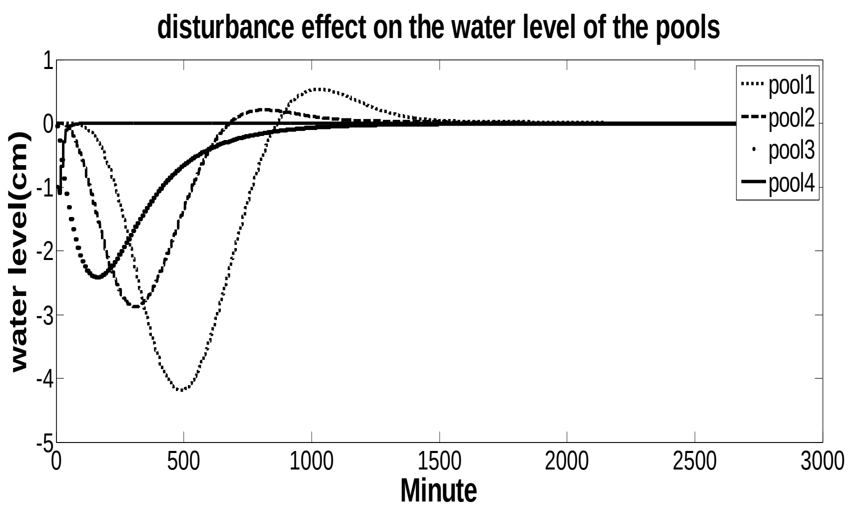

Fig. 3. The upstream transient error propagation and amplification phenomenon.

for $i = 1, 2, ..., n$ and $k \in \{0, 1, 2, ..., \}$. In the above dynamic model $v_i[k] = M_i x_{i+1}[k]$ represents the cascade interconnection, $x_i \in \mathbb{R}^{n_i}$ is the state variable of dimension $n_i \in \mathbb{N} \doteq \{1, 2, 3, ...\}$, $u_i \in \mathbb{R}$ is the reference set point, $y_i \in \mathbb{R}$ and $z_i \in \mathbb{R}$ are variables to be controlled, and $d_i \in \mathbb{R}$ is a known off-take disturbance for the ith sub-system.

For the purpose of illustration, in this section we consider an automated irrigation channel consisting of pools 2,5,8 and 9 of the East Goulburn main irrigation channel located in Victoria, Australia. Numerical values for parameters describing these automated pools are given in Table I [18].

Fig. 3 illustrates the response of this automated irrigation channel to an off-take disturbance with the value of $\bar{d}_4 = 8\frac{m^3}{min}$ applied to the last pool (pool 9 of the East Goulburn irrigation channel) for the first 135 minute (note that for simulations, the desired steady state values for water levels are set to be $1m$ above datum in Fig. 2 and for simulations the datum for the water levels are moved to the desired steady state water levels).

Fig. 3 illustrates the upstream transient error propagation and amplification phenomenon due to interplay between off-take flow disturbance and transport delay in automated irrigation channels. At their worst, the undesirable transient characteristics can result in instability and performance degradation due to actuator limitations. One way to mitigate such effect is to equip automated irrigation channels with a supervisory controller that properly manages reference set points u_is of local PI controllers by solving a quadratic constrained optimal control problem [18]. To formulate this problem, we use the following augmented state space representation for the distributed dynamic model (21).

$$\begin{cases} x[k+1] = Ax[k] + Bu[k] + Fd[k], \\ y[k] = Cx[k], \\ z[k] = Dx[k], \end{cases} \tag{22}$$

where

$$x[k] = \begin{pmatrix} x_1'[k] & x_2'[k] & \dots & x_n'[k] \end{pmatrix}', \quad u[k] = \begin{pmatrix} u_1[k] & u_2[k] & \dots & u_n[k] \end{pmatrix}',$$

$$d[k] = \begin{pmatrix} d_1[k] & d_2[k] & \dots & d_n[k] \end{pmatrix}', \quad y[k] = \begin{pmatrix} y_1[k] & y_2[k] & \dots & y_n[k] \end{pmatrix}',$$

$$z[k] = \begin{pmatrix} z_1[k] & z_2[k] & \dots & z_n[k] \end{pmatrix}'.$$

As the supervisory controller can have a larger time step than the time step of local PI controllers, which is set to be $T = 1$ minute for simulation study, the time step for supervisory controller is set to be ST, $S \in \mathbb{N}$. Hence, the dynamic model for supervisory controller is obtained by taking the model re-sampling approach, which involves holding the inputs to the system constant for the whole new sample period, and aggregating the dynamic (22) across the new sample period, as follows:

$$\begin{cases} x[k+1] = A^S x[k] + (\sum_{j=0}^{S-1} A^{S-j-1} B) u[k] + (\sum_{j=0}^{S-1} A^{S-j-1} F d[Sk+j]), \\ y[k] = Cx[k], \\ z[k] = Dx[k], \\ k = \{0, 1, 2, 3, \dots\}. \end{cases} \tag{23}$$

After obtaining the re-sampled model, the number of states in the re-sampled model (23) is reduced while maintaining the input-output behavior using balanced truncation [23]. Consequently, the obtained reduced model for the supervisory controller has the following representation

$$\begin{cases} \hat{x}[k+1] = \hat{A}\hat{x}[k] + \hat{B}u[k] + \hat{d}[k], \\ y[k] = \hat{C}\hat{x}[k], \\ z[k] = \hat{D}\hat{x}[k], \\ k = \{0, 1, 2, 3, ...\}, \end{cases} \tag{24}$$

where $\hat{d}[k]$ represents the effect of known off-take disturbances on the reduced model. Now, the supervisory controller manages reference set points u_is of local PI controllers by solving the following quadratic constrained optimal control problem [18].

$$\min_{u=(u_1,\dots,u_n)} J_L(\hat{x}[0], \hat{d}_0^{L-1}, r, u)$$
$$\text{subject to (24) and} \left\{ \begin{array}{c} y_i[k] \in [W_i, H_i],\ \ u_i[k] \in [W_i, H_i] \\ z_i[k] \in [E_i, Z_i] \end{array} \right\} \forall i \in [1, n],\ \ k \in [0, L-1], \tag{25}$$

where L is a measure of irrigation season length, the interval $[W_i, H_i]$ is the admissible region for water-levels y_is and also decision variables u_is, the interval $[E_i, Z_i]$ is the admissible region for variable z_i which is a measure of water flow rate, and

$$J_L(\hat{x}[0], \hat{d}_0^{L-1}, r, u) = \sum_{i=1}^{n} \sum_{k=0}^{L-1} ||y_i[k] - r_i||_Q^2 + ||u_i[k] - u_i[k-1]||_R^2 + ||z_i[k]||_P^2 \quad (u_i[-1] = 0). \tag{26}$$

Here $||.||$ denotes the Euclidean norm (i.e., $||z||_P^2 \doteq z'Pz$), $\hat{x}[0]$ is the vector of known initial states, $\hat{d}_0^{L-1} \doteq \{\hat{d}[k]\}_{k=0,1,\dots,L-1}$, where $\hat{d}[k] = (\, \hat{d}_1'[k] \quad . \quad . \quad . \quad \hat{d}_n'[k] \,)'$ is a collection of known vectors which represent the effects of off-take disturbances, $r = (\, r_1 \quad \dots \quad r_n \,)'$ is the vector of desired steady state values for y_is, and $Q, P \geq 0,\ R > 0$ are weighting matrices. The first norm in the cost functional (26) penalizes deviation of water levels from the corresponding desired values, and the second norm penalizes large changes in the input vector for the local PI controllers; and therefore, it tries to provide a smooth input trajectory. The last norm tries to minimize the input flow rates as z_is are measures of input flow rates; and therefore, it is desirable to make them as small as possible to keep water in reservoir as much as possible.

Remark 5.1: In automated irrigation channels, u_is produced by the optimization problem (25) are the set points for the distributed PI controllers. PI controllers are tuned so that water levels y_is follow these set points. Since the ith water level y_i must be within the bound $y_i \in [W_i, H_i]$ and is supposed to follow the set point u_i, the set point u_i must be limited in the same bound $u_i \in [W_i, H_i]$.

Remark 5.2: As the optimal control problem (25) is a constrained problem and the season length L is long, the receding horizon idea must be used to solve the constrained optimal control problem (25). For large-scale automated irrigation channels (i.e., when n is large) the computation overhead for solving the optimal control problem at each receding horizon using centralized techniques is very large. As shown in [24] the computation overhead using centralized technique is of the order of $\mathcal{O}(n^4)$; while the overhead using the distributed method with either single-level or two-level architecture for communication is $\mathcal{O}(n)$. Hence, for large-scale automated irrigation channels, a practical way to solve the constrained optimal control problem (25) using full computational capacity of existing local decision makers is to implement the distributed model predictive controller with either two-level or single-level architecture for communication. Using these controllers at each time instant k by solving a constrained optimization problem with horizon length $N << L$, the set points u_is for the time instant k are obtained and they are applied to automated irrigation channels by distributed decision makers.

Remark 5.3: In automated irrigation channels, for a given vector $\hat{x}[k]$ of measured states at time instant k, collection of known vectors $\hat{d}_k^{k+N-1} \doteq \{\hat{d}[j]\}_{j=k,\ldots,k+N-1}$, $(N = \min(\bar{N}, L-k))$ and vector of desired steady state values for references r, the following cost functional

$$J(\hat{x}[k], \hat{d}_k^{k+N-1}, r, u) \doteq \sum_{i=1}^{n} \sum_{j=k}^{k+N-1} ||y_i[j] - r_i||_Q^2 + ||u_i[j] - u_i[j-1]||_R^2 + ||z_i[j]||_P^2 \tag{27}$$

subject to the dynamic model (24) is a quadratic function of inputs u_i, $i \in \{1, 2, ..., n\}$, as the dynamic model (24) for automated irrigation channels are linear. Moreover, as $[W_i, H_i], [E_i, Z_i] \subset \mathbb{R}$ are closed convex sets, and linear transformations preserve closeness and convexity [17], the inputs (i.e. decision variables) in the following constrained optimization problem, belong to closed convex constraint sets. Hence, with $\mathbf{g} = (\hat{x}[k], \hat{d}_k^{k+N-1}, r)$ the following optimization problem

$$\min_{u=(u_1,\ldots,u_n)} J(\hat{x}[k], \hat{d}_k^{k+N-1}, r, u)$$

$$\text{subject to (24) and} \left\{ \begin{array}{c} y_i[j] \in [W_i, H_i],\ u_i[j] \in [W_i, H_i] \\ z_i[j] \in [E_i, Z_i] \end{array} \right\} \forall i \in [1, n],\ j \in [k, k+N-1],$$

is of the form of the general optimization problem (1). Hence, the proposed distributed model predictive control method can be used to solve the optimal control problem (25).

B. Simulation Results

To illustrate the satisfactory performance of the distributed model predictive controller with two-level architecture for communication in attenuation of the undesired upstream transient error propagation and amplification phenomenon, this controller is applied to the automated irrigation channel with four pools with numerical values as given in Table I. The performance of this controller is also compared with the performance of the distributed model predictive controller with single-level architecture for communication [1].

In this section it is assumed that the above automated irrigation channel is subject to an off-take disturbance with the value of $\bar{d}_4 = 8\frac{m^3}{min}$ for the first 135 minute (the first 15 time steps of the supervisory controller). It is also assumed that $x[0] = \mathbf{0}$, $r = \mathbf{0}$, $S = 9$, $L = 240$, $\bar{N} = 10$, $[W_i, H_i] = [-0.2m, 0.2m]$ and $[E_i, Z_i] = [0, 0.75^{3/2}]$. Note that by applying balanced truncation, the reduced model has only 22 states instead of 92 states. Also, a similar method as used in [18] is used to guarantee the recursive feasibility of DMPCs. To apply DMPC with two-level architecture for communication, two neighborhoods are considered. The first neighborhood includes the first and the second decision makers and the second neighborhood includes the third and fourth decision makers.

Decision makers in different neighborhood broadcast their updated decision variables simultaneously without collision, e.g., using Orthogonal Frequency Division Multiple Access (OFDMA) scheme[14]. And, within a neighborhood, decision makers exchange their updated decision variables without collision using TDMA scheme, which allocates 2.5 second to each decision maker to broadcast its data to all other decision makers in its neighborhood. Hence, the communication load for each inner iterate communication is 5 second. However, for outer iterate communication multi-hopping is required that induces communication delay such that the outer iterate communication load is 50 second. For the simulation study, $\bar{p}$ is set to be 10 and $\bar{t} = 1$. Hence, the communication overhead of the DMPC with two-level architecture for communication is $\bar{p} \times 5 + \bar{t} \times 50 = 10 \times 5 + 1 \times 50 = 100$ second., and the overhead of the DMPC with single-level architecture for communication is $\bar{p} \times 50 = 500$ second.

To compare the performance of DMPCs with single-level and two-level communication architectures for communication in water level regulations, we compare their responses with computational latency with their responses without computational latency for water level regulation; because the responses of both methods for water level regulation without computational latency are almost identical.

For simulation purposes, MATLAB *quadprog.m* solver is used, which is interfaced via YALMIP

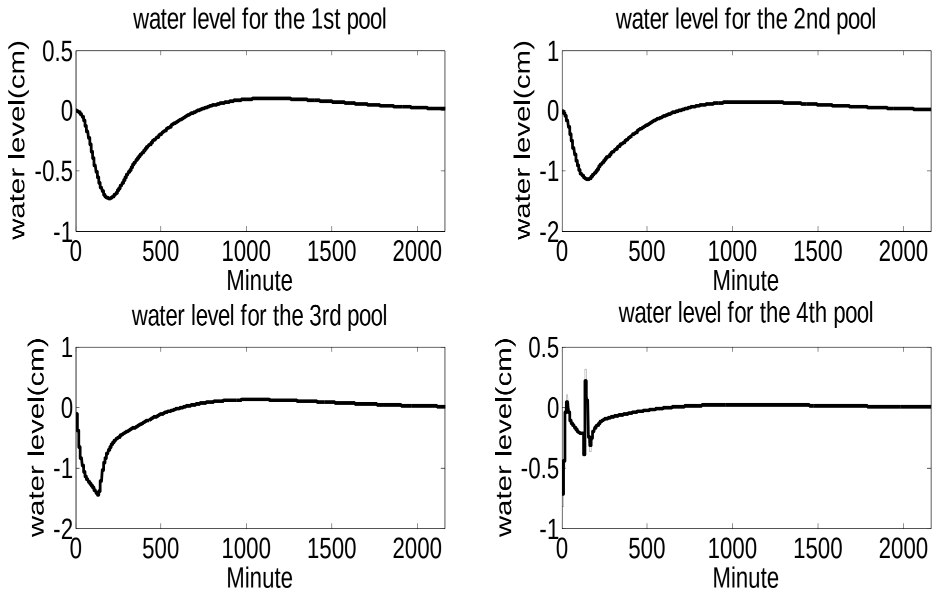

Fig. 4. The DMPC with two-level architecture for communication with computational latency of 113 second. Solid line: without computational latency, dotted line: with latency. As is clear from this figure there is no mismatch between these two responses; and the magnitude of transient errors between water levels and the desired values decreases as we move towards upstream pools. This indicates that the DMPC with two-level architecture for communication attenuates the upstream transient error propagation and amplification phenomenon.

[25] to compute the optimal controls numerically. The computer hardware is a Dell Inspiron laptop computer, processor: Intel(R) Core (TM) i5 CPU M450 at 2.40GHz, with 32Bits operating system. Note that the computation overhead calculation in the following simulation study captures what calculation will be done in parallel by calculating the computation time of each decision maker of a neighborhood in a given inner iterate update; and then choosing the computation time of the decision maker with the largest computation time as the computation time of neighborhood in that inner iterate.

Fig. 4 illustrates the responses of the DMPC with two-level architecture for communication without and with considering the computational latency. As the computation overhead in average is 13second, its computational latency in average is 113second, which is almost $2/9$th of the time step of 9minute. As is clear from Fig. 4, the response with the computational latency is the same as the response without latency. This result is expected because the computational latency here is almost 4 times smaller than the time step. As is clear from Fig. 4, the magnitude of transient errors between water levels and the desired values decreases as we move towards upstream

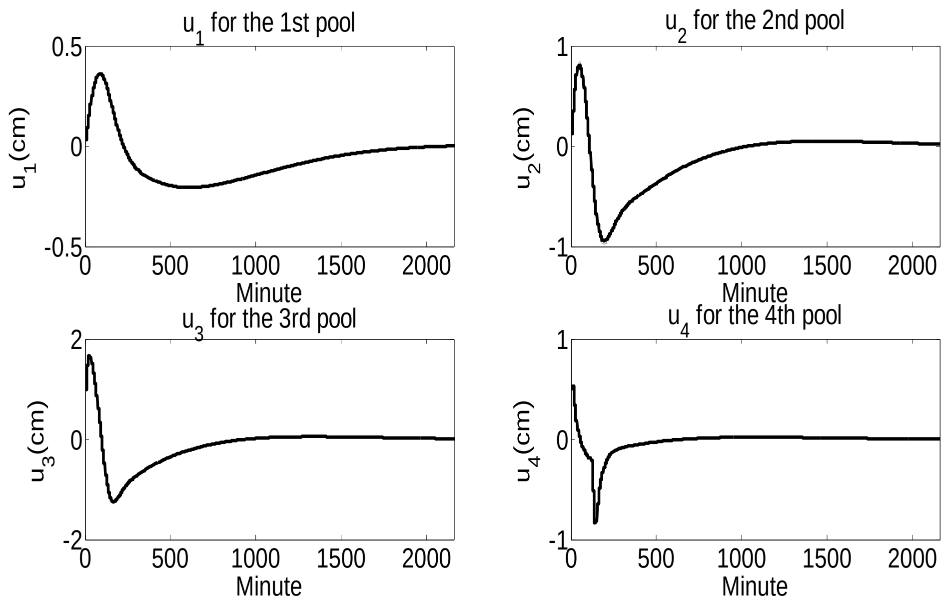

Fig. 5. Set-point trajectories delivered by the DMPC with two-level architecture for communication. Solid line: without computational latency, dotted line: with latency. As is clear from this figure there is no mismatch between these two trajectories.

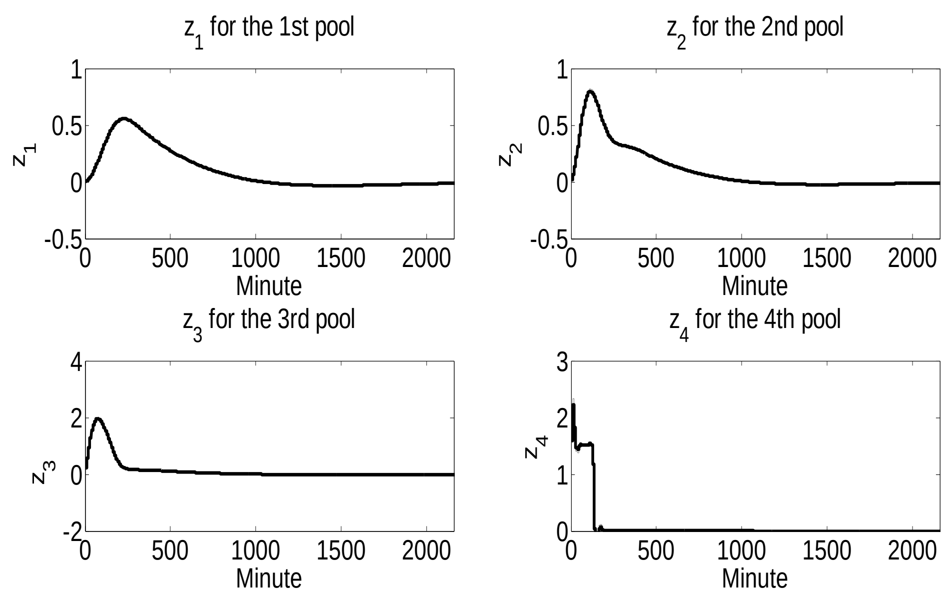

Fig. 6. z_is (measures of flows between pools delivered by the DMPC with two-level architecture for communication). Solid line: without computational latency, dotted line: with latency. As is clear from this figure there is no mismatch between z_is for these two cases.

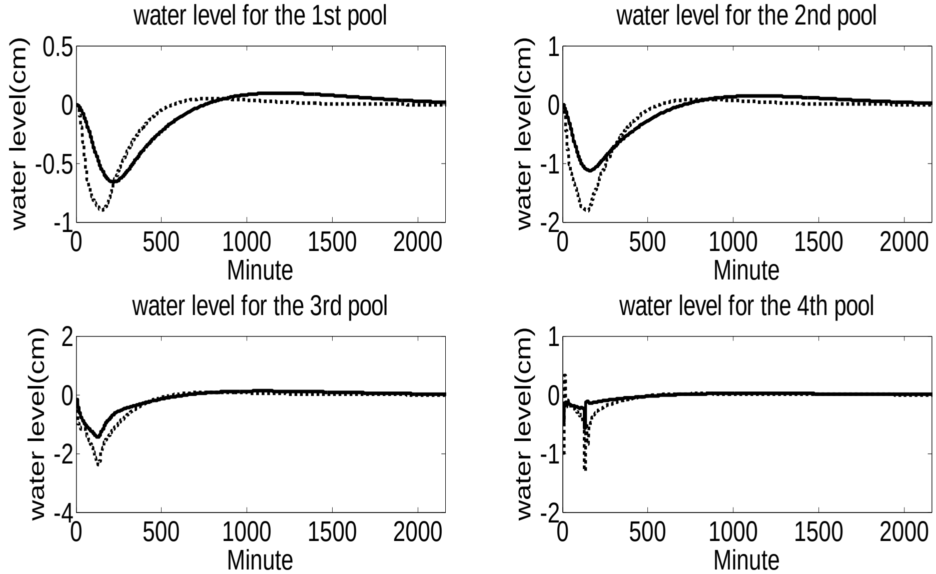

Fig. 7. The DMPC with single-level architecture for communication with computational latency of 515 second. Solid line: without computational latency, dotted line: with latency. This figure illustrates that the performance of the case with computational latency (dotted line) in disturbance rejection is worst than the performance of the case without latency (solid line).

pools. This indicates that the DMPC with two-level architecture for communication attenuates the upstream transient error propagation and amplification phenomenon. Fig. 5 illustrates the set-point trajectories delivered by the DMPC with two-level architecture for communication; and Fig. 6 illustrates z_is trajectories, which are measures of input flows (input flow $= C_i^{in} z_i$) for this case.

Fig. 7 illustrates the responses of the DMPC with single-level architecture for communication without and with considering the computational latency. Here, the average computation overhead is 15second, and hence the computational latency in average is 515second. Fig. 7 clearly illustrates that the performance of the case with the computational latency in disturbance rejection is worst than the performance of the case without the computational latency. For the first pool, the deviation of the response with computational latency from the response without latency is up to 0.25 cm, for the second pool is up to 0.75cm, for the third pool is up to 1cm and for the last pool is up to 1.2 cm. This results is expected because of large computational latency here. Fig. 8 illustrates the set-point trajectories delivered by the DMPC with single-level architecture; and Fig. 9 illustrates z_is for this case. Fig. 4 and Fig. 7 illustrate that the DMPC with two-level architecture has a performance better than the performance of the DMPC with single-level architecture by better managing communication overhead. Fig. 10 illustrates the response

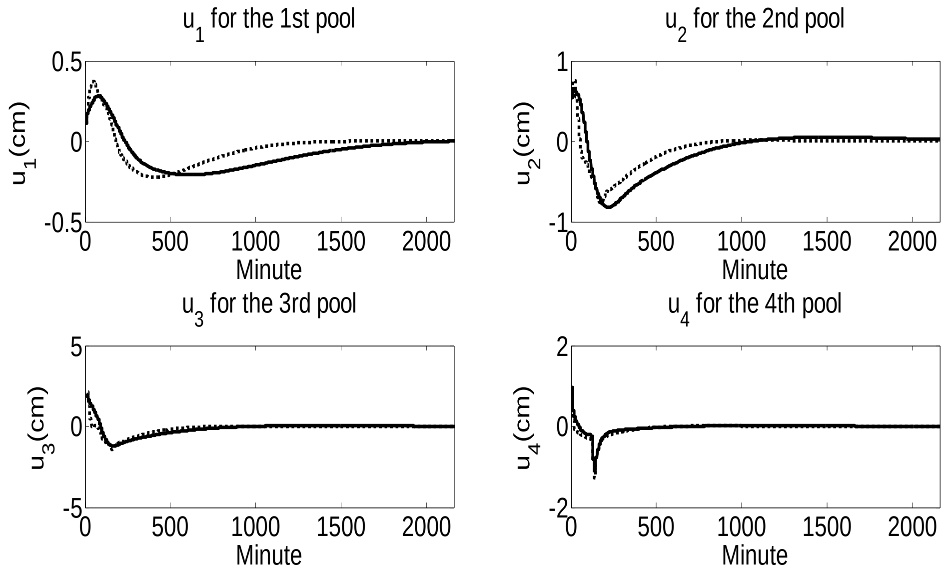

Fig. 8. The set-point trajectories delivered by the DMPC with single-level architecture for communication. Solid line: without computational latency, dotted line: with latency.

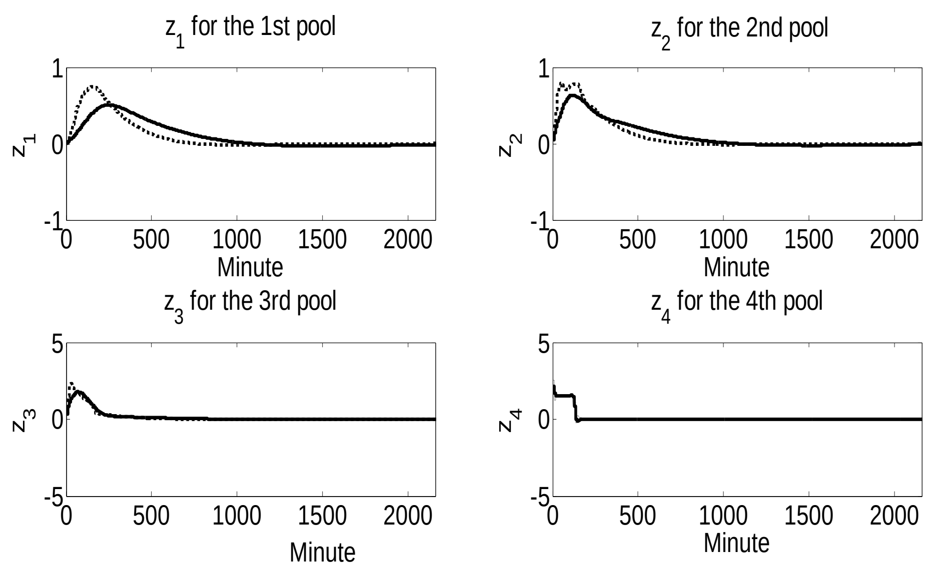

Fig. 9. z_is (measures of flows between pools delivered by the DMPC with single-level architecture for communication). Solid line: without computational latency, dotted line: with latency.

of DMPC with single-level architecture for communication without computational latency and with computational latency of 113 second. From this figure and Fig. 4 it follows that for the same computational latency, the response with computational latency of the DMPC with two-level architecture for communication is better than the response of the DMPC with single-level architecture for communication in disturbance rejection. This perhaps is due to the extra flexibility introduced by parameter λ_i in the DMPC with two-level architecture for communication.

Now, for the sensitivity analysis of communication overhead and to quantify the deviation of the response with computational latency from the ideal response without computational latency, we introduce the sum absolute error criterion as follows: $SAE = \sum_{i=1}^{4} \sum_{t=0}^{SL} |y_i^{wo}(t) - y_i^{w}(t)|$, where y_i^{wo} denotes the response without computational latency and y_i^{w} the response with computational latency. For DMPC with single-level architecture and outer iterate communication load of 1, 5, 10, 30, 50 second, the sum absolute error is 0.000091657, 14.9757, 14.6678, 13.5228, 8.1418, respectively. For DMPC with two-level architecture for communication and outer iterate communication load of 50 second and different values for inner iterate communication, the sum absolute error is shown in Table II. As is clear from this table, even with inner iterate communication load of 50 second (that is, when the ratio of inner iterate communication load versus outer iterate communication load is one) the SAE of the DMPC with two-level architecture is much smaller than that of the DMPC with single-level architecture. Note that for this case the communication overhead of the DMPC with two-level architecture is bigger than that of the DMPC with single-level architecture. This indicates that in the presence of computational latency, the response of the DMPC with two-level architecture is much closer to the ideal response without computational latency. Fig. 11 illustrates the response of the DMPC with two-level architecture for communication without and with computational latency (due to communication overhead and computation overhead) when the inner iterate communication load is 50 second. As is clear from this figure, although the communication overhead for this case is 550 second; and hence, the computational latency is high (563 second), the response with latency is very close to the ideal response without latency. The above analysis illustrates that for the simulated conditions, the DMPC with two-level architecture for communication has a better performance in disturbance rejection over whole communication overhead except in very small overhead where the performance of two methods are almost identical.

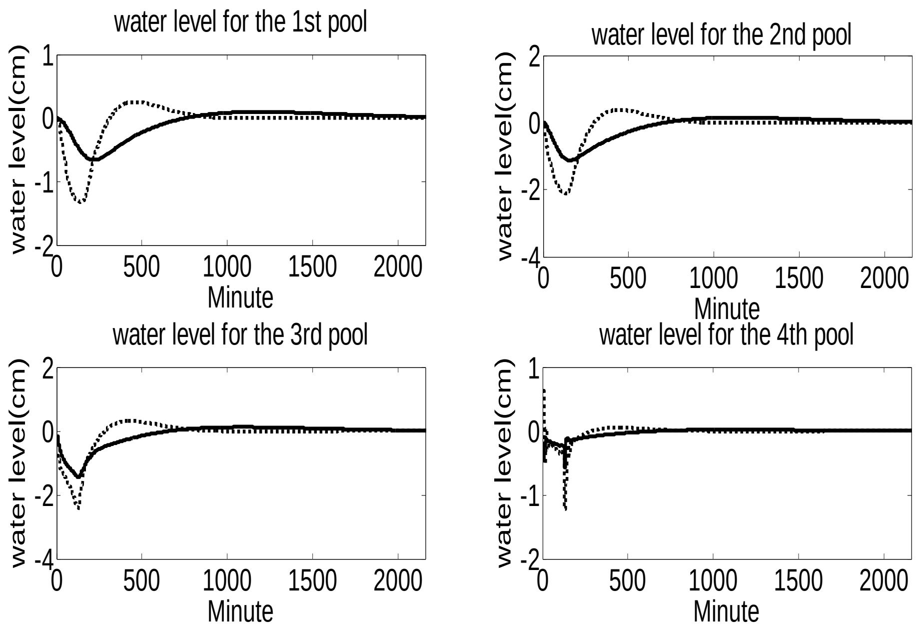

Fig. 10. The DMPC with single-level architecture for communication with computational latency of 113 second. Solid line: without computational latency, dotted line: with latency.

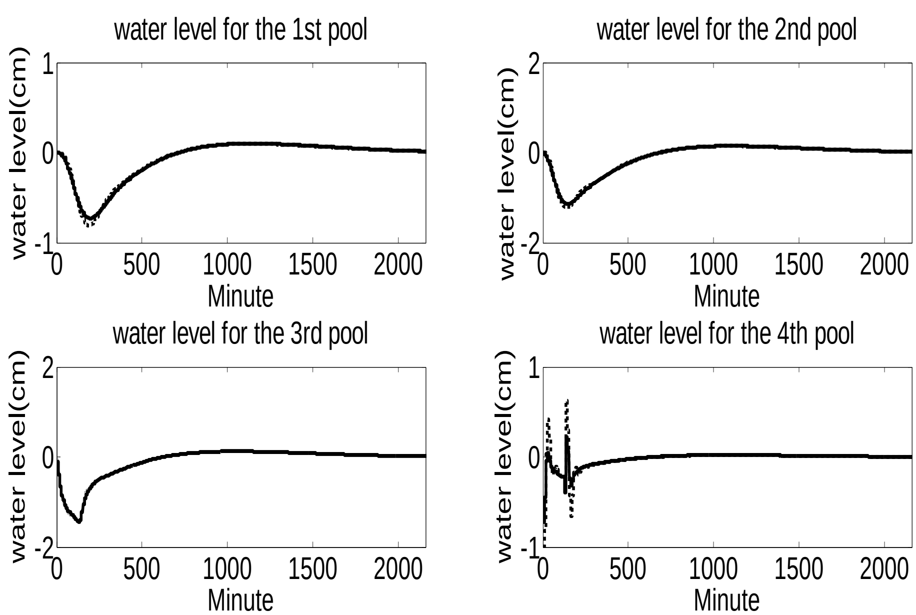

Fig. 11. The DMPC with two-level architecture for communication with computational latency of 563 second. Solid line: without computational latency, dotted line: with latency.

Inner iterate communication load (second)	Communication overhead (second)	SAE
1	60	0.0097
5	100	0.1955
10	150	0.2950
30	350	0.59
50	550	0.8777

TABLE II

SAE FOR DMPC WITH TWO-LEVEL ARCHITECTURE FOR COMMUNICATION WITH OUTER ITERATE COMMUNICATION LOAD OF 50 SECOND, $\bar{p} = 10$ AND DIFFERENT VALUES FOR INNER ITERATE COMMUNICATION LOAD.

VI. CONCLUSION AND DIRECTION FOR FUTURE RESEARCH

Feasibility, convergence and optimality of the distributed optimization method of [2] that exploits a two-level architecture for communication were mathematically proved. For an automated irrigation channel, the satisfactory performance of the distributed model predictive control method that is based on this distributed optimization method, was illustrated and compared with the performance of the distributed model predictive control of [1] that exploits a single-level architecture for communication. It was illustrated that the former method has a better performance by better managing communication overhead. For future it is interesting to address the problem of identifying disjoint neighborhoods for a given distributed system with arbitrary topology so that the fastest convergence rate to the optimal solution is achieved. Also, it is interesting to compute the communication overhead for a given system and develop techniques for exchanging information between sub-systems and neighborhoods with minimum communication overhead. It is also interesting to study the effects of parameters π_js and λ_is in the quality of response. These problems are left for future investigation.

REFERENCES

[1] Stewart, B. T., Venkat, A. N., Rawlings, J. B., Wright, S. J. and Pannocchia, G. (2010). Cooperative distributed model predictive control. *Systems and Control Letters*, 59, pp. 460-469.

[2] Stewart, B. T., Venkat, A. N., Rawlings, J. B. and Wright, S. J. (2010). Hierarchical cooperative distributed model predictive control. *Proceedings of American Control Conference*, Marriott Waterfront, Baltimore, MD, USA, pp. 3963-3968.

[3] Christofides, P. D., Scattolini, R., Munoz de la Pena D. and Liu, J. (2013). Distributed model predictive control: a tutorial review and future research directions. Elsevier Computers and Chemical Engineering, vol. 51, pp. 21-41.

[4] Liu, J., Munoz de la Pena, D. and Christofides, P. D. (2010). Distributed model predictive control of nonlinear systems subject to asynchronous and delayed measurements. *Automatica*, vol. 46, pp. 52-61.

[5] Liu, J., Chen, X., Munoz de la Pena, D. and Christofides, P. D. (2012). Iterative distributed model predictive control of nonlinear systems: handling asynchronous, delayed measurements. *IEEE Transactions On Automatic Control*, 52(2), pp. 528-534.

[6] Giselsson, P. and Rantzer, A. (2014). On feasibility, stability and performance in distributed model predictive control. *IEEE Transactions on Automatic Control*, 59(4), pp. 1031-1036.

[7] Maestre, J. M., Munoz de la Pena D., Camacho E. F. and Alamo T. (2011). Distributed model predictive control based on agent negotiation. *Journal of Process Control*, 21, pp. 685-697.

[8] Negenborn, R. R. (2007) *Multi-agent model predictive control with applications to power networks*. PhD Thesis, Delft University of Technology.

[9] Scattolini, R. (2009). Architecture for distributed and hierarchical model predictive control - a review. *Journal of Process Control*, 19(5), pp. 723-731.

[10] Doan, M. D., Giselsson, P., and Keviczky, T. (2013). A distributed accelerated gradient algorithm for distributed model predictive control of hydro power valley. *Control Engineering Practice*, 21(11), pp. 1594-1605.

[11] Negenborn, R. R., Van Overloop, P. J., Keviczky, T., De Schutter, B. (2009). Distributed model predictive control of irrigation canals. *Networks and Heterogeneous Media*, 4(2), pp. 359-380.

[12] Igreja, J., Cadete, F., Lemos, J. (2011). Application of distributed model predictive control to a water delivery canal. in *Proceedings of the 19th Mediterranean Conference On Control and Automation*. Corfu, Greece, pp. 682-687.

[13] Anand, A., Joshua, G., Sundaramoorthy, S., Samavedham, L. (2011). Coordinating multiple model predictive controllers for multi-reservoir management. in *Proceedings of the 2011 IEEE International Conference On Networking, Sensing and Control*, Delft, The Netherlands, pp. 1-6.

[14] Goldsmith, A. (2005). *Wireless Communications*. Cambridge University Press.

[15] Farhadi, A. Dower, P. M. and Cantoni, M. (2013). Computation time analysis of centralized and distributed optimization algorithms applied to automated irrigation networks. *Proceedings of 2013 Australian Control Conference*, Perth, Australia, pp. 263-269.

[16] Billingsley, P. (1986). *Probability and measure*. John Wiley and Sons.

[17] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

[18] Kearney, M. and Cantoni, M. (2012). MPC-based reference management for automated irrigation channels. *Proceedings of 2012 Australian Control Conference*, Sydney, Australia, pp. 349-354.

[19] Chaudhry, M. H. (1993). *Open-channel flow*. Englewood Cliffs, NJ: Prentice-Hall.

[20] Weyer, E. (2001). System identification of an open water channel. *Elsevier Control Engineering Practice*, vol. 9, 1289-1299.

[21] Ooi, S. K., Krutzen, M. P. M. and Weyer, E. (2005). On physical and data driven modelling of irrigation channels. *Elsevier Control Engineering Practice*, vol. 13, pp. 461-471.

[22] Cantoni, M., Weyer, E., Li, Y., Ooi, S. K., Mareels, I. and Ryan, M. (2007). Control of large-scale irrigation networks. *Proceedings of the IEEE*, 95(1), pp. 75-91.

[23] Zhou, K., Doyle, J. C. and Glover, K. (1996). *Robust and optimal control*. Prentice Hall.

[24] Farhadi, A., Cantoni, M. and Dower, P. M. (2013). Computation time analysis of a distributed optimization algorithm applied to automated irrigation networks. *Proceedings of the 52nd IEEE Conference on Decision and Control*, Florence, Italy, pp. 2193-2199.

[25] Lofberg, J. (2004). Yalmip: a toolbox for modeling and optimization in MATLAB. in *Proceedings of the CACSD* Conference, Taipei, Taiwan. [Online]. Available: http://users.isy.liu.se/johanl/yalmip