Author/s:
Guo, M;Lang, A;Cantoni, M

Title:
Moving Horizon Estimation for Linear Cascade Systems

Date:
2018-01-01

Citation:
Guo, M., Lang, A. & Cantoni, M. (2018). Moving Horizon Estimation for Linear Cascade Systems. 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), 00, pp.2026-2031. IEEE. https://doi.org/10.1109/ICARCV.2018.8581272.

Persistent Link:
https://hdl.handle.net/11343/307665

Moving Horizon Estimation for Linear Cascade Systems

Meichen Guo, Adair Lang, Michael Cantoni

***Abstract*— A structured approach to the problem of state estimation for cascaded linear sub-systems is studied in terms of minimizing a measure of the error relative to a model over a moving horizon of past system input and output observations. A quadratic programming formulation of this optimization problem is considered and two approaches are explored. One approach involves solving the Karush-Kuhn-Tucker conditions directly, and the other is based on the alternating direction method of multipliers. In both cases, the problem structure can be exploited to yield distributed computations in the following sense: Construction of the estimate for each sub-system component of the state involves information pertaining to the two immediate neighbours only. Numerical simulations based on model data from an automated irrigation channel are used to investigate and compare the computational burden of the two approaches.**

I. Introduction

Moving horizon estimation (MHE) involves the use of observed data over a fixed moving time horizon to determine the state of a system by solving an optimization problem. In the literature, the MHE method has been applied to linear stochastic systems and deterministic systems; e.g., see [1]–[5]. Specifically, [3] focused on MHE for a class of deterministic unconstrained linear systems with knowledge of system inputs and outputs. The authors included a prediction term in the estimation cost function and gave an upper bound on the estimate error. In [4], sufficient conditions for the stability of MHE for linear constrained systems were derived.

When applied directly to networks of sub-systems, the aforementioned MHE strategies lead to centralized observers that lack the structure to enable distributed implementation. More recent work considers the design of decentralized or distributed MHE strategies. In [6], decentralized partition-based MHE (PMHE) algorithms are developed for linear large-scale systems. However, all-to-all data exchange is required for the sub-computations associated with the component of the state for each sub-system. The authors of [7] improved the performance of PMHE but still required all-to-all data exchange. In [8], a distributed MHE method is proposed for unconstrained large-scale systems described by sparse banded or sparse multibanded system matrices, which can arise when linear dynamical systems are cascaded in series. The approach is based on a Chebyshev approximation of the centralized solution of the MHE problem. The hop proximity of information required to compute the estimate of each sub-system state grows with the time horizon.

Funded in part by the Australian Research Council (LP160100666).

The authors are with Electrical and Electronic Engineering Department, The University of Melbourne, Parkville, VIC 3010, Australia cantoni@unimelb.edu.au

In this paper, MHE is considered for unconstrained linear cascade systems; i.e., a string of sub-systems with spatially directed flow of information, like an irrigation channel, vehicle platoon, or supply chain. The problem is reformulated as structured quadratic program, in which the initial prediction error is treated as a decision variable, and the dynamics are encoded as equality constraints. Instead of resolving the dynamics, as done in previous work [8], structure in the equality constraints is exploited in solving the quadratic program. While this approach is similar to [9] and [10] where the context is moving horizon control, there are distinctive aspects of the MHE context here, as noted in subsequent developments.

Two algorithms are developed to solve the aforementioned structured quadratic program for MHE. One algorithm involves direct solution of the Karush-Kuhn-Tucker (KKT) conditions for optimality [11], and the other one is based on the alternating direction method of multipliers (ADMM) [12]. The resulting algorithms are both highly structured. In particular, distributed computation can be achieved in the sense that the estimate of a particular sub-system component of the state can be determined from data that is associated with the immediate neighbours only. The two algorithms are applied in a simulation example and the results are analysed and compared. The approach based on direct solution of the KKT conditions involves sequential computation of sub-system components of the state, which leads to an overall computational time that scales linearly with the number of subsystems, and cubically in the time horizon. The information exchange between each step of the sequential computation of the sub-system states is limited to the neighbouring sub-systems in the cascade. Thus a distributed implementation would also incur low communication overhead. The ADMM approach, by contrast, can takes many iterations to converge. On the other hand, the iterations for computing the sub-system components of the states are amenable to parallelization, which leads to per iteration computation time that remains constant with the length of the cascade. However, as the number of iterations required can be large, the overall computation time and exchange of information may be excessive in a distributed implementation.

The rest of the paper is organised as follows. In Section II, the MHE problem is formulated into a quadratic program. Section III develops two structured solvers. In Section IV, an automated irrigation channel example is considered and the simulation results are analyzed. Finally, some concluding remarks are given in Section V.

II. Problem Formulation

Consider a linear cascade system consisting of $N \in \mathbb{N}$ sub-systems. Each sub-system i for $i = 1, \dots, N$ has discrete time dynamics

$$\begin{aligned} x_i(t+1) &= A_i x_i(t) + B_i u_i(t) + E_i x_{i-1}(t), \\ y_i(t) &= C_i x_i(t), \end{aligned} \tag{1}$$

where the system state $x_i(t) \in \mathbb{R}^{n_i}$, input $u_i(t) \in \mathbb{R}^{m_i}$, output $y_i(t) \in \mathbb{R}^{v_i}$ for time instant $t \in \mathbb{N}$, and $E_1 = 0$. Using the MHE method, at each time an estimate of the state is constructed over a given finite past time horizon based on input and output observations over that window of time; i.e., observations of the system output $y_{ti} = [y_i^\intercal(t-T), \dots, y_i^\intercal(t)]^\intercal$ and input $u_{ti} = [u_i^\intercal(t-T), \dots, u_i^\intercal(t-1)]^\intercal$, are known for given time horizon $T \in \mathbb{N}$. That is to say, at each $t = T, T+1, \dots$, the objective is to find the best (in some sense) estimate $\hat{x}_{ti} = [\hat{x}_{ti}^\intercal(t-T), \dots, \hat{x}_{ti}^\intercal(t)]^\intercal$ for $x_{ti} = [x_i^\intercal(t-T), \dots, x_i^\intercal(t)]^\intercal$. As problem data, in addition to y_{ti} and u_{ti}, a prediction $\bar{x}_{ti}(t-T)$, defined as

$$\begin{aligned} \bar{x}_{ti}(t-T) = A_i \hat{x}_{(t-1)i}(t-1-T) + B_i u_i(t-1-T) \\ + E_i \hat{x}_{(t-1)(i-1)}(t-1-T), \end{aligned}$$

is also used. This prediction is the estimate $\hat{x}_{(t-1)i}(t-T)$ at stage $t-1$.

More specifically, the MHE method involves solving the following optimisation problem, in a receding horizon fashion for $t = T, T+1, \dots$.

Problem 2.1: Given $\bar{x}_{ti}(t-T)$, y_{ti}, and u_{ti}, find $\hat{x}_{ti}$ and δ_{ti} for $i = 1, \dots, N$ that minimize the cost function

$$J_t = \frac{1}{2} \sum_{i=1}^{N} \Big[\mu \|\delta_{ti}\|^2 + \sum_{k=t-T}^{t} \|y_i(k) - C_i \hat{x}_{ti}(k)\|^2 \Big], \tag{2}$$

with $\mu \geq 0$, subject to the constraints

$$\begin{aligned} \hat{x}_{ti}(k+1) &= A_i \hat{x}_{ti}(k) + B_i u_i(k) + E_i \hat{x}_{t(i-1)}(k), \\ &\qquad k = t-T, \dots, t-1, \\ \hat{x}_{ti}(t-T) &= \bar{x}_{ti}(t-T) + \delta_{ti}, \end{aligned} \tag{3}$$

for $i = 1, \dots, N$.

Remark 2.1: The system dynamics is encoded explicitly as the equality constraint (3). In principle, the estimates $\hat{x}_{ti}(t-T+1), \dots, \hat{x}_{ti}(t)$ in the cost function can be resolved using the system dynamics, and $\hat{x}_{ti}(t-T)$ becomes the only decision variable in a problem with little structure; see [3]. Instead, in this work the structure of the equality constraints in Problem 2.1 is exploited to devise a solution method that is amenable to distributed computation. ■

Remark 2.2: The cost function includes δ_{ti} as a decision variable, which is constrained to be the difference between the estimate of $x_i(t-T)$ at current stage and its prediction from the previous stage. The parameter μ represents our confidence in the prediction from the previous stage. ■

Assumption 2.1: The pair $(\tilde{A}, \tilde{C})$ is observable over time horizon T where $\tilde{C} = \operatorname{diag}(C_1, \dots, C_N)$ and

$$\tilde{A} = \begin{bmatrix} A_1 & 0 & \cdots & \cdots & 0 \\ E_2 & A_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & E_N & A_N \end{bmatrix}.$$

As shown in [3, Proposition 1], if either $\mu > 0$ or Assumption 2.1 is satisfied, Problem 2.1 has a unique solution. Moreover, as noise is not considered in this work, the solution will asymptotically converge to the true states.

To facilitate exploitation of the cascade structure in (1), the cost function and constraints can be reformulated as follows

$$J_{ti} = \frac{1}{2} \mu \delta_{ti}^\intercal \delta_{ti} + \frac{1}{2} (y_{ti} - \bar{C}_i \hat{x}_{ti})^\intercal (y_{ti} - \bar{C}_i \hat{x}_{ti}), \tag{4}$$

$$-\bar{A}_i \hat{x}_{ti} + \bar{E}_i \hat{x}_{t(i-1)} + K_i \delta_{ti} + c_{ti} = 0 \tag{5}$$

where $\bar{E}_1 = 0$, $c_{ti} = \bar{B}_i u_{ti} + \tilde{x}_{ti}$, $\tilde{x}_{ti} = [\bar{x}_{ti}^\intercal(t-T), 0, \dots, 0]^\intercal$, $\bar{B}_i = [0, \dots, 0; \operatorname{diag}(B_i, \dots, B_i)]$, $\bar{C}_i = \operatorname{diag}(C_i, \dots, C_i)$,

$$\bar{A}_i = \begin{bmatrix} I_{n_i} & 0 & \cdots & \cdots & 0 \\ -A_i & I_{n_i} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A_i & I_{n_i} \end{bmatrix}, \quad K_i = \begin{bmatrix} I_{n_i} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$\bar{E}_i = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ E_i & \ddots & & & \vdots \\ 0 & E_i & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & E_i & 0 \end{bmatrix}.$$

III. Exploiting the Cascade Structure

In this section, Problem 2.1 is solved using two methods, and the cascade structure is exploited to obtain distributed algorithms.

A. Directly solving the KKT conditions

In this subsection, we obtain and directly solve the KKT conditions for Problem 2.1. Since the problem is a quadratic program with no inequality constraints, the KKT conditions are a system of linear equations characterizing optimality, and there is no need for Newton iterations to solve them.

Using (4) and (5) the KKT condition for Problem 2.1 can be written as

$$\begin{aligned} \bar{C}_i^\intercal \bar{C}_i \hat{x}_{ti} - \bar{A}_i^\intercal p_{ti} + \bar{E}_{i+1}^\intercal p_{t(i+1)} - \bar{C}_i^\intercal y_{ti} = 0, \\ \mu \delta_{ti} + K_i^\intercal p_{ti} = 0, \\ -\bar{A}_i \hat{x}_{ti} + \bar{E}_i \hat{x}_{t(i-1)} + K_i \delta_{ti} + c_{ti} = 0, \end{aligned} \tag{6}$$

for $i = 1, \dots, N$, where $p_{ti} \in \mathbb{R}^{n_i}$ is the dual variable, $\hat{x}_{Ti} = 0$, and $\bar{E}_{N+1} = 0$. Assuming that $\mu > 0$, we can use

$\delta_{ti} = -\mu^{-1}K_i^\intercal p_{ti}$ to eliminate δ_{ti}. Then, for each sub-system $i = 1, \ldots, N$, (6) becomes

$$D_i\zeta_{ti} + \Upsilon_i\zeta_{t(i-1)} + \Upsilon_{i+1}^\intercal\zeta_{t(i+1)} = \psi_{ti} \tag{7}$$

where $\zeta_{ti} = [\hat{x}_{ti}^\intercal, p_{ti}^\intercal]^\intercal$,

$$D_i = \begin{bmatrix} \bar{C}_i^\intercal\bar{C}_i & -\bar{A}_i^\intercal \\ -\bar{A}_i & -\mu^{-1}K_iK_i^\intercal \end{bmatrix}, \Upsilon_i = \begin{bmatrix} 0 & 0 \\ \bar{E}_i & 0 \end{bmatrix}, \psi_{ti} = \begin{bmatrix} \bar{C}_i^\intercal y_{ti} \\ -c_{ti} \end{bmatrix}.$$

Stacking equations (7) for all $i = 1, \ldots, N$ gives the overall KKT conditions as

$$\begin{bmatrix} D_1 & \Upsilon_2^\intercal & 0 & \cdots & 0 \\ \Upsilon_2 & D_2 & \Upsilon_3^\intercal & \ddots & \vdots \\ 0 & \Upsilon_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & D_{N-1} & \Upsilon_N^\intercal \\ 0 & \cdots & 0 & \Upsilon_N & D_N \end{bmatrix} \begin{bmatrix} \zeta_{t1} \\ \zeta_{t2} \\ \vdots \\ \vdots \\ \zeta_{tN} \end{bmatrix} = \begin{bmatrix} \psi_{t1} \\ \psi_{t2} \\ \vdots \\ \vdots \\ \psi_{tN} \end{bmatrix}. \tag{8}$$

The block tridiagonal structure of (8) facilitates computation of the solution.

Lemma 3.1: The unique solution to (8) is given by

$$\tilde{\psi}_{ti} = \begin{cases} \psi_{tN} & \text{for } i = N \\ \psi_{ti} - \Upsilon_{i+1}^\intercal\Sigma_{i+1}^{-1}\tilde{\psi}_{t(i+1)} & \text{for } i = N-1, \ldots, 1 \end{cases}$$
$$\zeta_{ti} = \begin{cases} \Sigma_1^{-1}\tilde{\psi}_{t1} & \text{for } i = 1 \\ \Sigma_i^{-1}(\tilde{\psi}_i - \Upsilon_i\zeta_{t(i-1)}) & \text{for } i = 2, \ldots, N, \end{cases} \tag{9}$$

where

$$\Sigma_i = \begin{cases} D_N & \text{for } i = N \\ D_i - \Upsilon_{i+1}^\intercal\Sigma_{i+1}^{-1}\Upsilon_{i+1} & \text{for } i = N-1, \ldots, 1. \end{cases} \tag{10}$$

In particular, the inverse of Σ_i can be expressed as $\Sigma_i^{-1} = S_i^\intercal\Omega_iS_i$ where

$$S_i = \begin{bmatrix} I_{\bar{n}_i} & 0 \\ 0 & -\bar{A}_i^{-1} \end{bmatrix},$$
$$\Omega_i = \begin{bmatrix} R_i(I_{\bar{n}_i} + Q_iR_i)^{-1} & I_{\bar{n}_i} - R_i(I_{\bar{n}_i} + Q_iR_i)^{-1}Q_i \\ (I_{\bar{n}_i} + Q_iR_i)^{-1} & -(I_{\bar{n}_i} + Q_iR_i)^{-1}Q_i \end{bmatrix},$$
$$Q_i = \begin{cases} \bar{C}_N^\intercal\bar{C}_N & \text{for } i = N \\ \bar{C}_i^\intercal\bar{C}_i - \bar{E}_{i+1}^\intercal(\Sigma_{i+1})_{22}^{-1}\bar{E}_{i+1} & \text{for } i = 1, \ldots, N-1 \end{cases},$$

$\bar{n}_i = n_i(T+1)$, and $R_i = \mu^{-1}\bar{A}_i^{-1}K_iK_i^\intercal\bar{A}_i^{-\intercal}$ for $i = 1, \ldots, N$.

Proof: The backward-forward substitution is used to obtain the solution (9) [13]. First, using $\Sigma_N = D_n$ and $\tilde{\psi}_{tN} = \psi_{tN}$ to solve the last block of (8) gives $\zeta_{tN} = \Sigma_N^{-1}(\tilde{\psi}_N + \Upsilon_N\zeta_{t(N-1)})$. Next, this solution is used with Σ_{N-1} and $\tilde{\psi}_{N-1}$ to solve $\zeta_{t(N-1)}$. This process is repeated until the first block is reached, to yield the solution described by (9) and (10).

Note that matrix Σ_i can be expressed as

$$\Sigma_i = \begin{bmatrix} Q_i & -\bar{A}_i^\intercal \\ -\bar{A}_i & -\mu^{-1}K_iK_i^\intercal \end{bmatrix}. \tag{11}$$

Moreover,

$$\bar{\Sigma}_i = S_i\Sigma_iS_i^\intercal = \begin{bmatrix} Q_i & I_{\bar{n}_i} \\ I_{\bar{n}_i} & -R_i \end{bmatrix} \tag{12}$$

is congruent to Σ_i. Denoting $\Omega_i = \bar{\Sigma}_i^{-1}$ gives that $\Sigma_i^{-1} = S_i^\intercal\bar{\Sigma}_i^{-1}S_i = S_i^\intercal\Omega_iS_i$. ■

Remark 3.1: The block tridiagonal system (8) can also be solved by using iterative consensus methods, such as the ones developed in [14]–[16]. In this case, every computing agent, one for each sub-system, maintains iterates of all variables, and inter-iterate exchange of information is limited to neighbouring agents. However, the number of iterations required for these methods cannot be bounded *a priori*. Moreover, the rate of convergence decreases, and therefore the number of iterations and amount of information exchange to converge increase, as the cascade length increases. In particular, it is known that the rate of convergence of the consensus methods is related to the second smallest eigenvalue $\lambda_2(L)$ of the graph Laplacian, which tends to 0 as the length of a cascade increases. ■

B. An ADMM approach

Following the approach in [10], define the splitting variable $\hat{z}_{ti} = \hat{x}_{t(i-1)}$ for $i = 2, \ldots, N$. Then then constraints in Problem 2.1 can be re-written as

$$\begin{aligned} -\bar{A}_i\hat{x}_{ti} + \bar{E}_i\hat{z}_{ti} + K_i\delta_{ti} + c_{ti} = 0, \\ \hat{z}_{ti} - \hat{x}_{t(i-1)} = 0. \end{aligned} \tag{13}$$

Denoting $\hat{x}_t = [\hat{x}_{t1}^\intercal, \ldots, \hat{x}_{tN}^\intercal]^\intercal$, $\hat{z}_t = [\hat{z}_{t2}^\intercal, \ldots, \hat{z}_{tN}^\intercal]^\intercal$, $\delta_t = [\delta_{t1}^\intercal, \ldots, \delta_{tN}^\intercal]^\intercal$, $\xi_t = [\hat{z}_t^\intercal, \delta_t^\intercal]^\intercal$, and $c_t = [c_{t1}^\intercal, \ldots, c_{tN}^\intercal]^\intercal$ gives the constraints as

$$F\hat{x}_t + G\xi_t - \hat{c}_t = 0 \tag{14}$$

where

$$F = \begin{bmatrix} -\hat{A} \\ \Psi \end{bmatrix}, \quad G = \begin{bmatrix} \hat{E} & \hat{K} \\ I_{\tilde{n}} & 0_{\tilde{n}\times\hat{n}} \end{bmatrix}, \quad \hat{c}_t = \begin{bmatrix} -c_t \\ 0_{\tilde{n}\times 1} \end{bmatrix},$$

$\Psi = [-I_{\tilde{n}}, 0]$, $\hat{A} = \text{diag}(\bar{A}_1, \ldots, \bar{A}_N)$, $\hat{E} = \text{diag}(\bar{E}_1, \ldots, \bar{E}_N)[0_{[(T+1)\sum_2^N n_i]\times n_1(T+1)}, I_{(T+1)\sum_2^N n_i}]^\intercal$, $\tilde{n} = (T+1)\sum_{i=1}^{N-1} n_i$, $\hat{n} = \sum_{i=1}^N n_i$, and $\hat{K} = \text{diag}(K_1, \ldots, K_N)$. Then Problem 2.1 can be reformulated as follows.

Problem 3.1: With the notation defined above, for a cascade system composed of sub-systems (1),

$$\begin{aligned} \min_{\hat{x}_t, \xi_t} \quad & f(\hat{x}_t) + g(\xi_t) \\ \text{subject to} \quad & F\hat{x}_t + G\xi_t - \hat{c}_t = 0 \end{aligned} \tag{15}$$

where $f(\hat{x}_t) = \frac{1}{2}(y_t - \hat{C}\hat{x}_t)^\intercal(y_t - \hat{C}\hat{x}_t)$, $g(\xi) = \frac{1}{2}\mu(H\xi_t)^\intercal(H\xi_t)$, $y_t = [y_{t1}^\intercal, \ldots, y_{tN}^\intercal]^\intercal$, $\hat{C} = \text{diag}(\bar{C}_1, \ldots, \bar{C}_N)$, and $H = [0, I_{\tilde{n}}]$.

To solve this problem, define the augmented Lagrangian

$$\begin{aligned} L_{\rho t}(\hat{x}_t, \xi_t, \lambda_t) = f(\hat{x}_t) + g(\xi_t) \\ + \frac{\rho_t}{2}\|F\hat{x}_t + G\xi_t - \hat{c}_t + \lambda_t\|_2^2 \end{aligned} \tag{16}$$

where ρ_t is a penalty parameter and λ_t is the scaled dual variable. The ADMM algorithm involves the following iterations [12]:

$$\begin{aligned}\hat{x}_t^{j+1} &= \underset{\hat{x}_t}{\text{argmin}}\ L_{\rho t}(\hat{x}_t, \xi_t^j, \lambda_t^j),\\ \xi_t^{j+1} &= \underset{\xi_t}{\text{argmin}}\ L_{\rho t}(\hat{x}_t^{j+1}, \xi_t, \lambda_t^j),\\ \lambda_t^{j+1} &= \lambda_t^j + \alpha_t(F\hat{x}_t^{j+1} + G\xi_t^{j+1} - \hat{c}_t) \end{aligned} \tag{17}$$

for $j \in \mathbb{N}$ where $\alpha_t > 0$ is the step size for the dual update. Since, $\hat{x}_t^{j+1}$ and ξ_t^{j+1} minimize $L_{\rho t}(\hat{x}_t, \xi_t^j, \lambda_t^j)$ and $L_{\rho t}(\hat{x}_t^{j+1}, \xi_t, \lambda_t^j)$, respectively, the iterations are given by

$$\begin{aligned}(\hat{C}^\intercal\hat{C} + \rho_t F^\intercal F)\hat{x}_t^{j+1} &= -\rho_t F^\intercal(G\xi_t^j - \hat{c}_t + \lambda_t^j) + \hat{C}^\intercal y_t,\\ (\rho_t G^\intercal G + \mu H^\intercal H)\xi_t^{j+1} &= -\rho_t G^\intercal(F\hat{x}_t^{j+1} - \hat{c}_t + \lambda_t^j),\\ \lambda_t^{j+1} &= \lambda_t^j + \alpha_t(F\hat{x}_t^{j+1} + G\xi_t^{j+1} - \hat{c}_t).\end{aligned}$$

Note that,

$$\begin{aligned}\hat{C}^\intercal\hat{C} + \rho_t F^\intercal F &= \text{diag}(\Phi_1, \ldots, \Phi_N),\\ \rho_t G^\intercal G + \mu H^\intercal H &= \begin{bmatrix}\rho_t\hat{E}^\intercal\hat{E} + \rho_t I_{\tilde{n}} & \rho_t\hat{E}^\intercal\hat{K}\\ \rho_t\hat{K}^\intercal\hat{E} & \rho_t\hat{K}^\intercal\hat{K} + \mu I_{\hat{n}}\end{bmatrix}\end{aligned}$$

where $\Phi_i = \bar{C}_i^\intercal\bar{C}_i + \rho_t\bar{A}_i^\intercal\bar{A}_i + \rho_t I$ for $i = 1, \ldots, N-1$ and $\Phi_N = \bar{C}_N^\intercal\bar{C}_N + \rho_t\bar{A}_N^\intercal\bar{A}_N$. It can be verified that $\hat{E}^\intercal\hat{K} = \hat{K}^\intercal\hat{E} = 0$. As such, $\rho_t G^\intercal G + \mu H^\intercal H = \text{diag}(\rho_t\hat{E}^\intercal\hat{E} + \rho_t I_{\tilde{n}}, (\rho_t+\mu)I_{\hat{n}})$. Therefore, both matrices $\hat{C}^\intercal\hat{C}+\rho_t F^\intercal F$ and $\rho_t G^\intercal G+\mu H^\intercal H$ are block diagonal, symmetric, and positive definite, yielding the following lemma.

Lemma 3.2: The iterations (17) are uniquely defined.

Remark 3.2: Typically the function $f(\hat{x}_t)$ is convex but not strongly convex since $\hat{C}^T\hat{C}$ can be singular. However, both matrices F and G in the constraints have full column rank. So according to [17, Theorem 3.1], the subproblems for $\hat{x}_t$ and ξ_t are strongly convex, and the algorithm (17) converges linearly to the optimal solution if the step size α_t is sufficiently small. This matches the simulation results in Section IV, where $\alpha = 1$. It is of note that different split reformulations of Problem 2.1 do not as readily lead to these important properties and the well-posedness result of Lemma 3.2. ■

C. Computational Overheads

In this subsection, the computational overhead of the two algorithms, including exchanges between the computational stages associated with the estimate of each sub-system, is discussed.

When the KKT conditions are solved directly, for each sub-system $i = 1, \ldots, N-1$, the calculation of $\tilde{\psi}_{ti}$ requires the vector $\tilde{\psi}_{t(i+1)}$ pertaining to sub-system $i+1$, and for each sub-system $i = 2, \ldots, N$, the calculation of ζ_{ti} requires the vector $\zeta_{t(i-1)}$ pertaining to sub-system $i-1$. Therefore, there is a computational dependence up the cascade and then down the cascade, respectively. Overall, to construct the estimate for each sub-system $i = 2, \ldots, N-1$, there is a computational dependence of the component of the estimate for sub-system i and its two immediate neighbours. Thus, the calculations are sequential and the number of exchanges is bounded by $2N$. The computational time is N times the computational time for one sub-system, which scale cubically with T.

For ADMM, the block diagonal structure enables parallel computation of the iterations for each sub-system. After updating $\hat{x}_{ti}^j$ for all $i = 1, \ldots, N$, the estimate $\hat{x}_{ti}^j$ is used to compute $\hat{z}_{ti}^j$ for $i+1$, each $i = 1, \ldots, N-1$. After updating $\hat{z}_{ti}^j$ for $i = 2, \ldots, N$, the estimate $\hat{z}_{ti}^j$ is used to compute the estimate $\hat{x}_{ti}^{j+1}$ for $i-1$. Note that an infinity norm stopping criterion can be tested in a decentralized way. Thus, relative to the direct KKT method, given N processors, the per-iterate computation time of ADMM is independent of the length of the cascade. However, since it is difficult to bound the the number of iterations required for ADMM to converge *a priori*, the total time and number of exchanges cannot be bounded as in the case of the direct KKT method.

When the KKT conditions are solved directly, it is noted that matrices Σ_i for $i = 1, \ldots, N$ are independent of the input and output data at each time stage. Therefore, the calculation of Σ_i^{-1} can be done at stage $t = T$ then stored for future use. This will save some computational time in the subsequent stages; when this is done only the first stage incurs computation that scales cubically in the time horizon. Similar action can be taken for ADMM as Φ_i for $i = 1, \ldots, N$ are independent of the measurements.

IV. Numerical Example

In this section, we apply the algorithms developed in the previous section to an automated irrigation channel. Simulation results are illustrated and analyzed to evaluate the performance of the developed algorithms. A comparison between the two algorithms is given in Subsection IV-B.

A. Simulation results

In the simulation, the irrigation channel consists of a string of pools linked by two adjacent flow regulators. As shown in [18], for channels that operate under decentralised distant-downstream control, each pool can be described by a sub-system in the form of (1). For each sub-system, the control input $u_i(t)$ is the water level reference and the output $y_i(t)$ is the water level. Each sub-system has 4 states including 2 for the water level dynamics and 2 for the PI-type feedback controller that sets the upstream inflow on the basis of the measured downstream water level error. The MHE problem involves using the most recent measurements of the water level reference and water level over a certain time horizon to estimate the full state of each pool.

In the simulation, we use the model data from [19] for the pools, the distributed controllers, and the discretisation sample-period. The many pool channels used in the simulation are constructed by concatenating sections of the channel considered in [19]. For each sub-system $i = 1, \ldots, N$ in the form of (1), $n_i = 4$, $m_i = 1$, and $v_i = 1$. The other model parameters are non-uniform along the channel.

The ADMM algorithm is applied by choosing penalty parameter $\rho_t = 0.5$ and dual update step size $\alpha_t = 1$. The

primal residual r_P and dual residual r_D are defined as

$$r_P^j = F\hat{x}_t^j + G\xi_t^j - \hat{c}_t, \quad r_D^j = \rho_t[F^\intercal G(\xi_t^{j+1} - \xi_t^j)]. \quad (18)$$

To get comparable estimation results between directly solving KKT and using ADMM, we choose the stopping criteria for the iterations in ADMM as $\|r_P\|_\infty < 10^{-5}$ and $\|r_D\|_\infty < 10^{-5}$. However, this tolerance makes the number of iterations required for convergence of ADMM extremely large. To speed up the convergence, we choose an equivalent state-space representation of the channel model obtained by pre-scaling the system. It is noted that, this change also reduces the estimate error when the KKT conditions are directly solved, but does not have much impact on its computational time. This highlights the sensitivity of ADMM to system setup.

Figure 1 illustrates the normalised computational time of solving the KKT conditions directly and the normalised computational time required for the iterations to converge when ADMM is implemented without parallelism; i.e., the iterates associated with each sub-system are computed sequentially on a single processor. In the simulation, the cascade length N and time horizon length T vary from 5 to 100, and the figure is in log-log scale.

It can be observed from Figure 1 that, when the KKT conditions are directly solved, the computational time scales linearly in N when T is fixed and exhibits cubic scalability in T when N is fixed. In the case that $N = T$, the computational time scales as NT^3. When ADMM is used, the simulation results show that the computational time needed for the iterations to converge scales linearly in N. With an increasing T, the computational time has a scalability between linear and quadratic. The normalised computational time per iteration with growing N and T is illustrated in Figure 2. It shows that within one iteration, the computational time scales linearly in both N and T. For further analysis, Figure 3 illustrates the number of iterations needed for convergence with different initial conditions. It can be seen that, the number of iterations remains constant in the length of N and scales sub-linearly in T. Under different initial conditions, the number of iterations also shows sub-linear scalability in T, but is considerably less than the other case.

Since only the water level of each pool is measured in the simulation, the function $f(\hat{x}_t)$ in Problem 3.1 is convex but not strongly convex. However, as mentioned in the previous section, the subproblems for each $\hat{x}_t$ and ξ_t are strongly convex and linear convergence rate can be guaranteed. The convergence rates of the primary residual r_P with different penalty parameter ρ_t are illustrated in Figure 4 and compared with the cases where $f(\hat{x}_t)$ is strongly convex, i.e., all the four states of each pool are measured. It can be observed that, whether $f(\hat{x}_t)$ is strongly convex or not, the asymptotic convergence rates are linear. The residuals converge faster in the beginning of the iterations when all states are measured. However, when it comes to the asymptotic phase of the iterations, measuring all the states does not always give higher convergence rates. For instance, in Figure 4, when

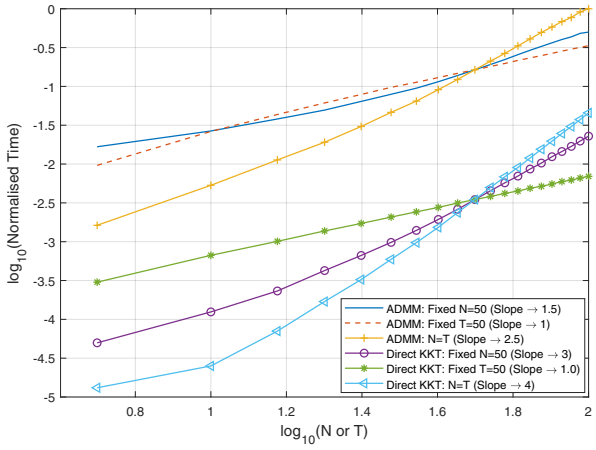

Fig. 1. Normalised computational time with respect to growing N and T.

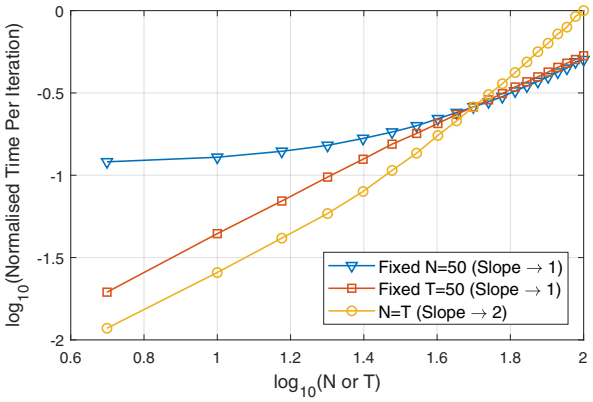

Fig. 2. Normalised computational time per iteration using ADMM.

$\rho_t = 0.5$, the residual converges faster when only the water level is measured.

B. Discussion

In this subsection, we discuss the performance of the two algorithms and some comparisons between them.

In theory, directly solving the KKT conditions in the way shown in (10) and (9) leads to an overall complexity of NT^3, which matches the simulation result in Figure 1. For ADMM, an interesting observation is that the number of iterations needed for convergence scales in T but not in N. The reason behind this is not clear for now and is part of the ongoing work. However, it can be concluded from Figure 3 that the system dynamics have a great effect on the number of iterations.

Comparing the simulation results of the two algorithms, we can see that in the setting of this work, directly solving the KKT conditions is computational advantageous. Since this work considers linear cascade systems without inequality constraints, the KKT conditions are linear and can be solved in 1 iteration while the ADMM algorithm needs a number of iterations to approach the solution. However, as the simulation is conducted using only one core in Matlab, the result cannot show the parallel computation of ADMM. If

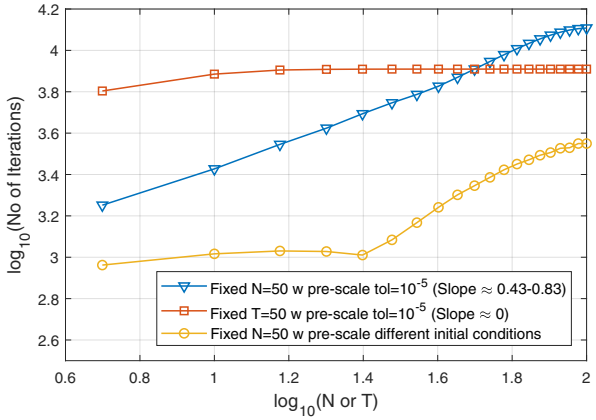

Fig. 3. Number of iterations need for ADMM to converge.

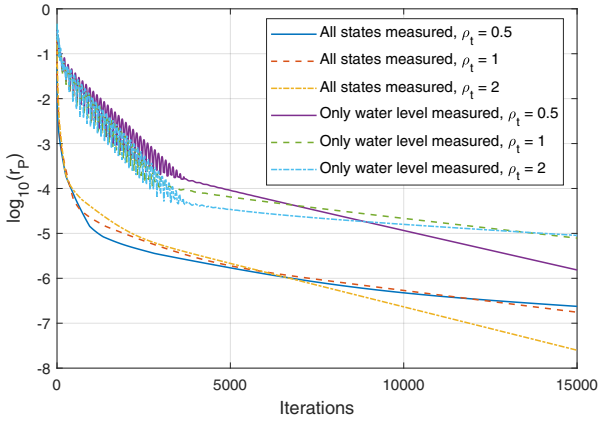

Fig. 4. Convergence rates of primary residual in ADMM.

each subsystem is equipped with a processor to compute in parallel, the overall computation time of ADMM can be reduced by a factor of N. Nonetheless, it can be seen from Fig. 1, the point of intersection where ADMM computation time would be for large N, and the overall inter-iteration information exchange overhead would be very high. Meanwhile, as mentioned in the previous subsection, ADMM is more sensitive to system setup, and the number of iterations needed for convergence is not bounded. In other words, when the problem is not well conditioned, directly solving KKT can render satisfying results while ADMM cannot or takes an extremely long time to reach convergence.

V. Conclusion

This work concerns the MHE of a class of unconstrained linear cascade systems. By exploiting the structure in the equality constraints that encode the system dynamics, two algorithms are developed for computing the solution. The resulting algorithms are amenable to distributed computation. Comparisons in terms of computational overheads, including data exchange between computational stages associated with each sub-system, are made. A simulation example based on the application of these algorithms to the dynamics of an automated irrigation channel is presented. Future work includes the consideration of strings in which there is bi-directional flow of information in the dynamics, and the inclusion of constraints.

References

[1] W. H. Kwon, P. S. Kim, and P. Park, "A receding horizon kalman FIR filter for linear continuous-time systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 11, pp. 2115–2120, Nov. 1999.

[2] K. V. Ling and K. W. Lim, "Receding horizon recursive state estimation," *IEEE Transactions on Automatic Control*, vol. 44, no. 9, pp. 1750–1753, Sep. 1999.

[3] A. Alessandri, M. Baglietto, and G. Battistelli, "Receding-horizon estimation for discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 3, pp. 473–478, Mar. 2003.

[4] C. V. Rao, J. B. Rawlings, and J. H. Lee, "Constrained linear state estimation - a moving horizon approach," *Automatica*, vol. 37, no. 10, pp. 1619–1628, Oct. 2001.

[5] D. Sui, T. A. Johansen, and L. Feng, "Linear moving horizon estimation with pre-estimating observer," *IEEE Transactions on Automatic Control*, vol. 55, no. 10, pp. 2363–2368, Oct. 2010.

[6] M. Farina, G. Ferrari-Trecate, and R. Scattolini, "Moving-horizon partition-based state estimation of large-scale systems," *Automatica*, vol. 46, no. 5, pp. 910–918, May 2010.

[7] R. Schneider and W. Marquardt, "Convergence and stability of a constrained partition-based moving horizon estimator," *IEEE Transactions on Automatic Control*, vol. 61, no. 5, pp. 1316–1321, May 2016.

[8] A. Haber and M. Verhaegen, "Moving horizon estimation for large-scale interconnected systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2834–2847, Nov. 2013.

[9] M. Cantoni, F. Farokhi, E. Kerrigan, and I. Shames, "Structured computation of optimal controls for constrained cascade systems," *International Journal of Control*, 2017, https://doi.org/10.1080/00207179.2017.1366668.

[10] M. Cantoni, A. Zafar, and F. Farokhi, "Scalable iterations for solving constrained LQ control problems with cascade dynamics," in *Proc. 23rd International Symposium on Mathematical Theory of Networks and Systems*, Hong Kong, 2018.

[11] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia: SIAM, 1997.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating eirection method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–100, 2011.

[13] G. Meurant, "A review on the inverse of symmetric tridiagonal and block tridiagonal matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 3, pp. 707–728, Jul. 1992.

[14] J. Liu, A. S. Morse, A. Nedić, and T. Başar, "Exponential convergence of a distributed algorithm for solving linear algebraic equations," *Automatica*, vol. 83, pp. 37–46, Sep. 2017.

[15] B. D. O. Anderson, S. Mou, A. S. Morse, and U. Helmke, "Decentralized gradient algorithm for solution of a linear equation," *Numerical Algebra, Control & Optimization*, vol. 6, no. 3, pp. 319–328, Sep. 2016.

[16] G. Shi, B. D. O. Anderson, and U. Helmke, "Network flows that solve linear equations," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2659–2674, Jun. 2017.

[17] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Mathematical Programming*, vol. 162, no. 1-2, pp. 165–199, Mar. 2017.

[18] L. Soltanian and M. Cantoni, "Decentralized string-stability analysis for heterogeneous cascades subject to load-matching requirements," *Multidimensional Systems and Signal Processing*, vol. 26, no. 4, pp. 985–999, Oct. 2015.

[19] A. R. Neshastehriz, M. Cantoni, and I. Shames, "Water-level reference planning for automated irrigation channels via robust MPC," in *European Control Conference (ECC)*, Strasbourg, France, Jun. 2014, pp. 1331–1336.