Delft University of Technology

Delft Center for Systems and Control

Technical report 10-063

An iterative scheme for distributed model predictive control using Fenchel's duality*

M.D. Doan, T. Keviczky, and B. De Schutter

If you want to cite this report, please use the following reference instead:
M.D. Doan, T. Keviczky, and B. De Schutter, "An iterative scheme for distributed model predictive control using Fenchel's duality," *Journal of Process Control*, vol. 21, no. 5, pp. 746–755, June 2011. doi:10.1016/j.jprocont.2010.12.009

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/10_063.html

An iterative scheme for distributed model predictive control using Fenchel's duality

Minh Dang Doan[a], Tamás Keviczky[a], Bart De Schutter[a]

[a]*Delft University of Technology, Delft, The Netherlands*
(e-mail: {m.d.doan,t.keviczky,b.deschutter}@tudelft.nl)

Abstract

We present an iterative distributed version of Han's parallel method for convex optimization that can be used for distributed model predictive control (DMPC) of industrial processes described by dynamically coupled linear systems. The underlying decomposition technique relies on Fenchel's duality and allows subproblems to be solved using local communications only. We investigate two techniques aimed at improving the convergence rate of the iterative approach and illustrate the results using a numerical example. We conclude by discussing open issues of the proposed method and by providing an outlook on research in the field.

Keywords: distributed optimization, dual decomposition methods, decentralized and cooperative control, distributed model predictive control

1. Introduction

Nowadays, Model Predictive Control (MPC) is widely used for controlling industrial processes [1], and it also has been studied thoroughly by the scientific community [2, 3, 4]. MPC can naturally handle operational constraints and, moreover, it is designed for multi-input multi-output systems, both of which contributed to the popularity of MPC. Another advantage of MPC is that it relies on optimization techniques to solve the control problem. Hence, improvements in optimization techniques can help to broaden the applications of MPC for more complex problems.

When considering a control problem for a large-scale networked system (such as complex manufacturing or infrastructure processes), using MPC in a centralized fashion may be considered impractical and unsuitable due to the computational burden and the requirement of global communications across the network. It is also inflexible against changes of network structure and the limitation of information exchange between different authorities who might be in control of a local subsystem. In order to deal with these limitations, Distributed MPC (DMPC) has been proposed for control of such large-scale systems, by decomposing the overall system into small subsystems [5, 6]. The subsystems then employ distinct MPC controllers that only solve local control problems, use local information from neighboring subsystems, and collaborate to achieve globally attractive solutions.

DMPC is an emerging topic for scientific research. The open issues of DMPC have recently been discussed in [7, 8]. Several DMPC methods were proposed for different problem setups. For systems with decoupled dynamics, a DMPC scheme for multiple vehicles with coupled cost functions was proposed in [9], utilizing predicted trajectories of the neighbors in each subsystem's optimization. A DMPC scheme with a suffi-

cient stability test for dynamically decoupled systems was presented in [10], in which each subsystem optimizes also over the behaviors of its neighbors. In [11], Richards and How proposed a robust DMPC method for decoupled systems with coupled constraints, based on constraint tightening and a serial solution approach. For systems with coupled dynamics and decoupled constraints, a DMPC scheme has been developed in [12] based on a Jacobi algorithm that deals with the primal problem, using a convex combination of new and old solutions. In [13], the neighboring subsystem states are treated as bounded contracting disturbances, and each subsystem solves a min-max problem. A partitioning-based algorithm was proposed in [14, 15], with sufficient conditions for the a posteriori stability analysis. In [16], Li et al. proposed an algorithm with stability conditions in which subproblems are solved in parallel in order to get a Nash equilibrium. Several DMPC algorithms based on decomposing of the global optimization problems were proposed in [17, 18, 19]. Other recent work on applications of DMPC is reported in [20, 21, 22].

In this paper, we present a decomposition scheme based on Han's parallel method [23, 24], aiming to solve the centralized optimization problem of MPC in a distributed way. This approach results in two distributed algorithms that are applicable to DMPC of large-scale industrial processes. The main ideas of our algorithms are to find a distributed update method that is equivalent to Han's method (which relies on global communications), and to improve the convergence speed of the algorithm [25]. We will demonstrate the application of our methods in a simulated water network control problem. The open issues of the proposed scheme will be discussed to formulate future research directions.

The paper is organized as follows. The MPC problem is formulated and the underlying optimization problem is stated in Section 2. In Section 3, we summarize Han's parallel method for convex programs [24] as the starting point for our approach. In Section 4, we present two distributed MPC schemes that exploit the structure of the optimization problem for local communications. The first DMPC scheme uses a distributed iterative algorithm that we prove to be equivalent to Han's algorithm. As a consequence of this equivalence, the proposed DMPC scheme achieves the global optimum upon convergence and thus inherits feasibility and stability properties from its centralized MPC counterpart. The second DMPC scheme is an improved algorithm that aims to speed up the convergence of the distributed approach. In Section 5, we illustrate the application of the new DMPC schemes in an example system involving irrigation canals. In Section 6, we discuss the open issues of Han's method and other dual decomposition techniques for DMPC that motivate directions for future research. Section 7 concludes the paper.

2. MPC problem formulation

2.1. Subsystems and their neighborhood

Consider a plant consisting of M dynamically coupled subsystems. The dynamics of each subsystem are assumed linear and to be influenced directly by only a *small* number of other subsystems. Moreover, each subsystem i is assumed to have local linear coupled constraints involving only variables from a small number of other subsystems.

Based on the couplings, we define the 'neighborhood' of subsystem i, denoted as $\mathcal{N}^i$, as the set including i and the indices of subsystems that have either a direct dynamical coupling or a constraint coupling with subsystem i.

2.2. Coupled subsystem model

We assume that each subsystem can be represented by a discrete-time, linear time-invariant model of the form[1]:

$$x_{k+1}^i = \sum_{j \in \mathcal{N}^i} A^{ij} x_k^j + B^{ij} u_k^j, \tag{1}$$

where $x_k^i \in \mathbb{R}^{n^i}$ and $u_k^i \in \mathbb{R}^{m^i}$ are the states and control inputs of the i-th subsystem at time step k, respectively.

2.3. Linear coupled constraints

Each subsystem i is assumed to have local linear coupled constraints involving only variables within its neighborhood $\mathcal{N}^i$. Within one prediction period, all constraints that subsystem i is involved in can be written in the following form

$$\sum_{j \in \mathcal{N}^i} \sum_{k=0}^{N-1} D_k^{ij} x_k^j + E_k^{ij} u_k^j = c_{\text{eq}} \tag{2}$$

$$\sum_{j \in \mathcal{N}^i} \sum_{k=0}^{N-1} \bar{D}_k^{ij} x_k^j + \bar{E}_k^{ij} u_k^j \leq \bar{c}_{\text{ineq}} \tag{3}$$

in which N is the prediction horizon, c_{eq} and $\bar{c}_{\text{ineq}}$ are column vectors, and D_k^{ij}, E_k^{ij}, $\bar{D}_k^{ij}$, and $\bar{E}_k^{ij}$ are matrices with appropriate dimensions.

2.4. First MPC problem

We will formulate the centralized MPC problem for systems of the form (1) using a terminal point constraint approach that imposes constraints to zero out all terminal states. Under the conditions that a feasible solution of the centralized MPC problem exists, and that the point with zero states and inputs is in the relative interior of the constraint set, this MPC scheme ensures feasibility and stability, as shown in [3] and [26]. However, the algorithm proposed in this paper will also work with any other centralized MPC approach that does not require a terminal point constraint, provided that the subsystems have local stabilizing terminal controllers. We will further assume without loss of generality that the initial time is zero.

The optimization variable of the centralized MPC problem is constructed as a stacked vector of predicted subsystem control inputs *and* states over the prediction horizon:

$$\begin{aligned} \mathbf{x} = \Big[&\left(u_0^1\right)^T, \ldots, \left(u_0^M\right)^T, \ldots, \left(u_{N-1}^1\right)^T, \ldots, \left(u_{N-1}^M\right)^T, \\ &\left(x_1^1\right)^T, \ldots, \left(x_1^M\right)^T, \ldots, \left(x_N^1\right)^T, \ldots, \left(x_N^M\right)^T \Big]^T \end{aligned} \tag{4}$$

Recall that n^i and m^i denote the numbers of states and inputs of subsystem i. The number of optimization variables for the centralized problem is thus:

$$n_{\mathbf{x}} = N \sum_{i=1}^{M} m^i + N \sum_{i=1}^{M} n^i \tag{5}$$

[1] This system description is chosen for simplicity of exposition and our framework can be easily extended to consider output signals with appropriate observability assumptions.

The cost function of the centralized MPC problem is assumed to be decoupled and convex quadratic:

$$J = \sum_{i=1}^{M} \sum_{k=0}^{N-1} \left(\left(u_k^i\right)^T R_i u_k^i + \left(x_{k+1}^i\right)^T Q_i x_{k+1}^i \right) \tag{6}$$

with positive definite weights R_i, Q_i. This cost function can be rewritten using the decision variable $\mathbf{x}$ as

$$J = \mathbf{x}^T H \mathbf{x} \tag{7}$$

in which the Hessian H is an appropriate block-diagonal, positive definite matrix.

Remark 2.1. *The positive definiteness assumption on Q_i and R_i and the choice of the centralized variable as described in* (4) *without eliminating state variables will help to compute the inverse of the Hessian easily, by allowing simple inversion of each block on the diagonal of the Hessian.*

The centralized MPC problem, named ($\mathcal{P}$), is defined as: the minimization of (6), subject to (1) for $i = 1, \ldots, M, k = 0, \ldots, N-1$, (2) and (3) for $i = 1, \ldots, M$, as well as $x_N^i = 0$ for $i = 1, \ldots, M$.

We can rewrite problem ($\mathcal{P}$) in a compact form as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{x}^T H \mathbf{x} \\ \text{s.t.} \quad & a_l^T \mathbf{x} = b_l, \quad l = 1, \ldots, n_{\text{eq}} \\ & a_l^T \mathbf{x} \leq b_l, \quad l = n_{\text{eq}} + 1, \ldots, s \end{aligned} \tag{8}$$

with $s = n_{\text{eq}} + n_{\text{ineq}}$. The algorithms to be described in the next sections will focus on how to solve this optimization problem.

3. Han's parallel method for convex programs

Han's algorithm [24] is a method to decompose the Fenchel's dual problem [27]. Fenchel's duality theorem aims at minimizing a difference $f(\mathbf{x}) - g(\mathbf{x})$, where f is a convex function and g is a concave function. A special case of this problem is minimizing f over a constraint set C, where g is a penalty function for violating the constraint. In Han's problem, the set C is the intersection of local constraint sets, and the dual variables are iteratively projected onto the local constraint sets. As a consequence, the sum of the dual variables converges to the minimizer of the Fenchel's dual problem [24]. In this section, we summarize the main elements of Han's parallel method, followed by a simplified version for the case of definite quadratic programming.

3.1. Han's algorithm for general convex problems

The class of optimization problems tackled by Han's algorithm is the following:

$$\begin{aligned} \min_{\mathbf{x}} \quad & q(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in C \triangleq C_1 \cap \cdots \cap C_s \end{aligned} \tag{9}$$

where $C_1, \cdots, C_s$ are closed convex sets and $C \neq \varnothing$, and where $q(\mathbf{x})$ is uniformly convex[2] and differentiable on $\mathbb{R}^{n_\mathbf{x}}$.

A problem of type (9) can be solved by Han's algorithm. In the following algorithm we will describe Han's method, which is an iterative procedure. We use p as iteration counter of the algorithm, and the superscript (p) for variables that are computed at iteration p.

Algorithm 3.1. ***Han's algorithm for convex programs***

Let α be a sufficiently large number[3] and define $\boldsymbol{y}^{(0)} = \boldsymbol{y}_1^{(0)} = \cdots = \boldsymbol{y}_s^{(0)} = 0$, with $\boldsymbol{y}^{(0)}, \boldsymbol{y}_l^{(0)} \in \mathbb{R}^{n_x}, l = 1, \ldots, s$, and $\boldsymbol{x}^{(0)} = \nabla q^* \left(\boldsymbol{y}^{(0)}\right)$ with q^* being the conjugate function[4] of q. For $p = 1, 2, \ldots$, we perform the following computations:

1) For $l = 1, \ldots, s$, find $\boldsymbol{z}_l^{(p)}$ that solves

$$\begin{aligned} \min_{z} \quad & \frac{1}{2}\left\| \boldsymbol{z} + \alpha \boldsymbol{y}_l^{(p-1)} - \boldsymbol{x}^{(p-1)} \right\|_2^2 \\ s.t. \quad & \boldsymbol{z} \in C_l \end{aligned} \tag{10}$$

2) Assign

$$\boldsymbol{y}_l^{(p)} = \boldsymbol{y}_l^{(p-1)} + (1/\alpha)\left(\boldsymbol{z}_l^{(p)} - \boldsymbol{x}^{(p-1)}\right) \tag{11}$$

3) Set $\boldsymbol{y}^{(p)} = \boldsymbol{y}_1^{(p)} + \cdots + \boldsymbol{y}_s^{(p)}$

4) Compute

$$\boldsymbol{x}^{(p)} = \nabla q^* \left(\boldsymbol{y}^{(p)}\right) \tag{12}$$

In [24], Han and Lou also showed that Algorithm 3.1 converges to the global optimum if the conditions on q and C mentioned after (9) are satisfied.

Remark 3.2. *Han's method essentially solves the dual problem of* (9)*, so that $\boldsymbol{y}^{(p)}$ converges to the solution of the Fenchel's dual problem:*

$$\min_{\boldsymbol{y}} \left(q^*(\mathbf{y}) - \delta^*(\mathbf{y}|C)\right) \tag{13}$$

in which $\delta(\boldsymbol{x}|C)$ is the indicator function, which is 0 if $\boldsymbol{x} \in C$ and ∞ otherwise. The conjugate function of $\delta(\boldsymbol{x}|C)$ is $\delta^*(\mathbf{y}|C) = \sup_{\boldsymbol{x}\in C} \boldsymbol{y}^T \boldsymbol{x}$. According to Fenchel's duality theorem [27], the minimum of the convex problem $f(\boldsymbol{x}) - g(\boldsymbol{x})$, where f is a convex function on $\mathbb{R}^{n_x}$ and g is a concave function on $\mathbb{R}^{n_x}$, equals the maximum of the concave problem $g^*(\mathbf{y}) - f^*(\mathbf{y})$, or equivalently the minimum of $f^*(\mathbf{y}) - g^*(\mathbf{y})$. In this situation $f \equiv q$ and $g \equiv \delta$. A value $\boldsymbol{y}^{(\bar{p})}$ achieved when Algorithm 3.1 converges is an optimizer of (13)*, hence $\boldsymbol{x}^{(\bar{p})} = \nabla q^* \left(\boldsymbol{y}^{(\bar{p})}\right)$ is the solution of* (9)*.*

---

[2] A function $q(\mathbf{x})$ is uniformly convex (or strongly convex) on a set S if there is a constant $\rho > 0$ such that for any $\mathbf{x}_1,\ \mathbf{x}_2 \in S$ and for any $\lambda \in (0, 1)$:

$$q(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda q(\mathbf{x}_1) + (1-\lambda) q(\mathbf{x}_2) - \rho\lambda(1-\lambda)\|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

[3] α is a design parameter that has to be sufficiently large. With $\alpha \geq s/\rho$ Han's method will converge [24]. For positive definite QPs we can choose ρ as one half of the smallest eigenvalue of the Hessian matrix.

[4] The conjugate function of a function $q(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ is defined by: $q^*(\mathbf{y}) = \sup_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}} \left(\mathbf{y}^T\mathbf{x} - q(\mathbf{x})\right)$. The conjugate function q^* is always convex [28].

3.2. Han's algorithm for definite quadratic programs

In case the optimization problem has a positive definite cost function and linear constraints as in (9), the optimization problem (10) and the derivative of conjugate function (12) have analytical solutions, and then Han's method becomes simpler. In the following we show how the analytical solutions of (10) and (12) can be obtained when applying Algorithm 3.1 to the problem (8).

Remark 3.3. *The result of simplifying Han's method in this section is slightly different from the original one described in [24], so as to correct the minor mistakes we found in that paper.*

As in (9), each constraint $\mathbf{x} \in C_l$ is implicitly expressed by a scalar linear equality or inequality constraint. So (10) takes one of the following two forms:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\|\mathbf{z} + \alpha \mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \\ \text{s.t.} \quad & a_l^T \mathbf{z} = b_l \end{aligned} \tag{14}$$

or

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\|\mathbf{z} + \alpha \mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \\ \text{s.t.} \quad & a_l^T \mathbf{z} \leq b_l \end{aligned} \tag{15}$$

Let us first consider (15):

- If $a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) \leq b_l$, then $\mathbf{z}_l^{(p)} = \mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}$ is the solution of (15). Substituting this $\mathbf{z}_l^{(p)}$ into (11), leads to the following update of $\mathbf{y}_l^{(p)}$:

$$\begin{aligned} & \mathbf{y}_l^{(p)} = \mathbf{y}_l^{(p-1)} + (1/\alpha)\left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\right) \\ \Rightarrow \; & \mathbf{y}_l^{(p)} = 0 \end{aligned} \tag{16}$$

- If $a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) > b_l$, then the constraint is active. The optimization problem (15) aims to find the point in the half-space $a_l^T \mathbf{z} \leq b_l$ that minimizes its distance to the point $\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}$ (which is outside that half-space). The solution is the projection of the point $\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}$ on the hyperplane $a_l^T \mathbf{z} = b_l$, which is given by the following formula:

$$\mathbf{z}_l^{(p)} = \mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)} - \frac{a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l\right) - b_l}{a_l^T a_l} a_l \tag{17}$$

Substituting this $\mathbf{z}_l^{(p)}$ into (11), leads to:

$$\begin{aligned} \mathbf{y}_l^{(p)} &= \mathbf{y}_l^{(p-1)} + \frac{1}{\alpha}\left(-\alpha \mathbf{y}_l^{(p-1)} - \frac{a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l}{a_l^T a_l} a_l\right) \\ &= -\frac{a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l}{\alpha a_l^T a_l} a_l \end{aligned} \tag{18}$$

Then defining $\gamma_l^{(p)} = a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l$ yields

$$\mathbf{y}_l^{(p)} = -\frac{\gamma_l^{(p)}}{\alpha a_l^T a_l} a_l \tag{19}$$

If we define

$$\gamma_l^{(p)} = \max\{a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l, 0\} \tag{20}$$

then we can use the update formula (19) for both cases.

Similarly, for the minimization under equality constraint (14), we define

$$\gamma_l^{(p)} = a_l^T \left(\mathbf{x}^{(p-1)} - \alpha \mathbf{y}_l^{(p-1)}\right) - b_l \tag{21}$$

and the update formula (19) gives the result of (11).

Now we consider step 4) of Algorithm 3.1. As shown in [28], the function $q(\mathbf{x}) = \mathbf{x}^T H \mathbf{x}$ with H being a positive definite matrix, is uniformly convex on $\mathbb{R}^{n_\mathbf{x}}$ and has the conjugate function:

$$q^*(\mathbf{y}) = \frac{1}{2} \mathbf{y}^T H^{-1} \mathbf{y} \tag{22}$$

$$\Rightarrow \nabla q^*(\mathbf{y}) = H^{-1} \mathbf{y} \tag{23}$$

Consequently, in Han's algorithm for the definite quadratic program (8), it is not necessary to compute $\mathbf{z}^{(p)}$, and $\mathbf{y}^{(p)}$ can be eliminated using (19). We are now ready to describe the simplified Han's algorithm for problem (8), with the choice $\alpha = s/\rho$ (cf. footnote 3).

Algorithm 3.4. ***Han's algorithm for definite quadratic programs***

For each $l = 1, \ldots, s$*, compute*

$$c_l = \frac{-1}{\alpha a_l^T a_l} H^{-1} a_l \tag{24}$$

Initialize $\gamma_1^{(0)} = \cdots = \gamma_s^{(0)} = 0$ *and* $\boldsymbol{x}^{(0)} = 0$*. For* $p = 1, 2, \ldots$*, perform the following computations:*

1) *For each* l *corresponding to an equality constraint* ($l = 1, \ldots, n_{\text{eq}}$)*, compute* $\gamma_l^{(p)} = a_l^T \boldsymbol{x}^{(p-1)} + \gamma_l^{(p-1)} - b_l$.
   For each l *corresponding to an inequality constraint* ($l = n_{\text{eq}} + 1, \ldots, s$)*, compute* $\gamma_l^{(p)} = \max\{a_l^T \boldsymbol{x}^{(p-1)} + \gamma_l^{(p-1)} - b_l, 0\}$*;*

2) *Set*

$$\boldsymbol{x}^{(p)} = \sum_{l=1}^{s} \gamma_l^{(p)} c_l \tag{25}$$

Note that Han's method splits up the computation into s parallel subproblems, where s is the number of constraints. However, although Algorithm 3.4 is simpler than the original form in Algorithm 3.1, it still requires a *global update scheme* and the parallel problems still operate with the full-sized decision vector. Implementing the scheme in a DMPC system, where the goal is to reduce the size of local computations and to rely on local communication between subsystems only, is not straightforward. In the following section, we will exploit the structure of the problem (8), resulting in a distributed algorithm that does not require global communications.

4. Distributed version of Han's method for the MPC problem

4.1. Distributed version of Han's method with common step size

The main idea behind the distributed version of Han's method is illustrated in Figures 1(a) and 1(b), with a simple system consisting of 4 subsystems and the coupling matrix that shows how subsystems are coupled via their variables (boxes on the same row indicate the variables that are coupled in one constraint). In Han's method using global variables, a subsystem has to communicate with all other subsystems in order to compute the updates of the global variables. For the distributed version of Han's method, each subsystem i only communicates with the other subsystems of which the variables are necessary for computing the updates of its local variables, i.e., the subsystems in its neighborhood $\mathcal{N}^i$.

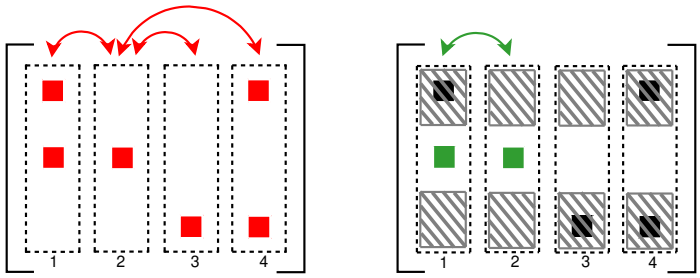

Figure 1: Illustration of communication links with (a) the centralized coordination version and (b) the distributed coordination version of Han's algorithm for an example 4-subsystem problem. In (a), an update for a global variable requires the 2nd subsystem to communicate with all the others. In (b), the 2nd subsystem only cares about its local variable, therefore it should only communicate with the 1st subsystem.

For the algorithm presented in this section, we use M local controllers attached to M subsystems. Each controller i then computes $\gamma_l^{(p)}$ with regards to a small set of constraints indexed by $l \in L_i$, where L_i is a set of indices[5] of several constraints that involve subsystem i. Subsequently, it performs a local update for its own variables, such that the parallel local update scheme will be equivalent to the global update scheme in Algorithm 3.4.

4.1.1. Initialization of the algorithm

Store invariant parameters

The parameter α is chosen as in Algorithm 3.4 and stored in the memory of all local controllers.

We also compute s invariant values c_l as in (24), in which each c_l corresponds to one constraint of (8). Note that H is block-diagonal, H^{-1} can be computed easily by inverting each block of H and has the same block structure as H. Hence c_l is as sparse as the corresponding a_l. We can see that c_l can be computed locally by a local controller with a priori knowledge of the parameter a_l and the weighting blocks on the diagonal of H that correspond to the non-zero elements of a_l.

[5]The choice of L_i will be described in the next section.

We assume that each local controller i knows its local dynamics, and the input and state weights of its neighbors in the cost function. Then each local controller i can compute the c_l values associated with its dynamic equality constraints.

Assign responsibility of each local controller

Each local controller is in charge of updating the variables of its subsystem. Moreover, we also assign to each local controller the responsibility of updating *some* intermediate variables that relate to several equality or inequality constraints in which its subsystem's states or inputs appear. The control designer has to assign each of the s scalar constraints to one of the M local controllers[6] such that the following requirements are satisfied:

- Each constraint is taken care of by one and only one local controller (even for a coupled constraint, there will be only one controller that is responsible).
- A local controller can only be in charge of constraints that involve its own variables.

Let L_i denote the set of indices l that local controller i is in charge of[7]. We also define $L_{\mathcal{N}^i}$ as the set of indices l corresponding to the constraints that are taken care of by subsystem i or by any neighbor of i:

$$L_{\mathcal{N}^i} = \bigcup_{j \in \mathcal{N}^i} L_j \tag{26}$$

If a local controller i is in charge of the constraints indexed by $l \in L_i$, then it computes locally c_l using (24) and exchanges these values with its neighbors. Then each local controller i stores $\{c_l\}_{l \in L_{\mathcal{N}^i}}$ in its memory throughout the optimization process.

4.1.2. Iterative procedure

The distributed algorithm consists of an iterative procedure running within each sampling interval. At each iteration, four steps are executed: two steps are communications between each local controller and its direct neighbors, and two are computation steps that are performed locally by the controllers in parallel. Since feasibility is only guaranteed upon convergence of Han's algorithm, we assume that the sampling time used is large enough such that the algorithm can converge within one sampling interval. This assumption will be used in Proposition 4.7, and its restrictiveness will be discussed in Section 6.

In this algorithm description, p is used to denote the iteration step. Values of variables obtained at iteration p are denoted with superscript (p).

Definition 4.1 (Index matrix of subsystems). *In order to present the algorithm compactly, we introduce the index matrix of subsystems: each subsystem i is assigned a square diagonal matrix $\mathfrak{I}^i \in \mathbb{R}^{n_x \times n_x}$, with an entry on the diagonal being 1 if it corresponds to the position of a variable of subsystem i in the vector $\boldsymbol{x}$, and 0 otherwise. In short, $\mathfrak{I}^i$ is a selection matrix such that the multiplication $\mathfrak{I}^i \boldsymbol{x}$ only retains the variables $u_0^i, \ldots, u_{N-1}^i, x_1^i, \ldots, x_N^i$ of subsystem i in its nonzero entries.*

[6] Note that s, the total number of constraints, is often much larger than M.

[7] Note that this partitioning is not unique and has to be created according to a procedure that is performed in the initialization phase.

From Definition 4.1 it follows that:

$$\sum_{i=1}^{M} \Im^i = I \tag{27}$$

Definition 4.2 (Self-image). *We denote with $\boldsymbol{x}^{(p)|i} \in \mathbb{R}^{n_x}$ the vector that has the same size as $\boldsymbol{x}$, containing $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$ (i.e. the values of i's variables computed at iteration p) at the right positions, and zeros for the other entries. This vector is called the self-image of $\boldsymbol{x}^{(p)}$ made by subsystem i.*

Using the index matrix notation, the relation between $\mathbf{x}^{(p)|i}$ and $\mathbf{x}^{(p)}$ is:

$$\mathbf{x}^{(p)|i} = \Im^i \mathbf{x}^{(p)} \tag{28}$$

Definition 4.3 (Neighborhood image). *Extending the concept of self-image, we denote with $\boldsymbol{x}^{(p)|\mathcal{N}^i}$ the neighborhood image of subsystem i made from $\boldsymbol{x}$. At step p of the iteration, subsystem i constructs $\boldsymbol{x}^{(p)|\mathcal{N}^i}$ by putting the values of its neighbors' variables and its own variables into the right positions, and filling in zeros for the remaining slots of $\boldsymbol{x}$. The neighborhood image $\boldsymbol{x}^{(p)|\mathcal{N}^i}$ satisfies the following relations:*

$$\boldsymbol{x}^{(p)|\mathcal{N}^i} = \sum_{j \in \mathcal{N}^i} \boldsymbol{x}^{(p)|j} \tag{29}$$

$$\boldsymbol{x}^{(p)|\mathcal{N}^i} = \left(\sum_{j \in \mathcal{N}^i} \Im^j \right) \boldsymbol{x}^{(p)} \tag{30}$$

By definition, we also have the following relation between the self-image and the neighborhood image made by the same subsystem:

$$\mathbf{x}^{(p)|i} = \Im^i \mathbf{x}^{(p)|\mathcal{N}^i} \tag{31}$$

Using the notation described above, we now describe the subtasks that each controller will use in the distributed algorithm.

- **Communications with the neighbors**

  Each controller i communicates only with its neighbors $j \in \mathcal{N}^i$ to get updated values of their variables and sends its updated variables to them. The data that each subsystem i transmits to its neighbor $j \in \mathcal{N}^i$ consists of the self-image $\mathbf{x}^{(p)|i}$ and the intermediate variables $\gamma_l^{(p)}, l \in L_i$, which are maintained locally by subsystem i.

- **Update intermediate variables γ_l**

  When the local controller i updates γ_l corresponding to each constraint $l \in L_i$ under its responsibility, it computes in the following manner:

  – If constraint l is an equality constraint ($l \in \{1, \ldots, n_{\text{eq}}\}$), then

  $$\gamma_l^{(p)} = a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l \tag{32}$$

  – If constraint l is an inequality constraint ($l \in \{n_{\text{eq}} + 1, \ldots, s\}$), then

  $$\gamma_l^{(p)} = \max\{a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l, 0\} \tag{33}$$

- **Update main variables**

  Local controller i uses all $\gamma_l^{(p)}$ values that it has (by communications and those computed by itself) to compute an 'assumed neighborhood image' $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$. Note that $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ has the same structure as the neighborhood image $\mathbf{x}^{(p-1)|\mathcal{N}^i}$. However, it is not the exact update of the neighborhood image. Indeed, $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ is used only for constructing the new self-image by selecting the variables of subsystem i in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$:

$$\mathbf{x}^{(p)|i} = \Im^i \mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} \tag{34}$$

  which contains $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$.

- **Check the local termination criteria**

  For each local controller, there are local termination criteria. The local termination criteria also aim to keep a subsystem informed when other subsystems terminate. Hence when one set of local termination criteria is satisfied, the termination criteria for all subsystems are also satisfied. Each controller checks the local termination criteria using local communications only[8]. When all local controllers have converged, the algorithm stops and the local control actions are implemented.

In the following, we will describe the new method using the distributed algorithm.

Algorithm 4.4. ***Distributed algorithm for definite quadratic programs***

Initialize with $p = 0$, $u_k^{i,(0)} = 0, x_{k+1}^{i,(0)} = 0, \forall i, k = 0, \ldots, N-1$ *(this means* $\boldsymbol{x}^{(0)|i} = 0, \forall i$*, implying* $\boldsymbol{x}^{(0)} = 0$*), and* $\gamma_l^{(0)} = 0, l = 1, \ldots, s$

Next, for $p = 1, 2, \ldots$*, the following steps are executed:*

1) ***Communications to get the updated main variables***
   Each controller i gets updated values of $\boldsymbol{x}^{(p-1)|j}$ *from its neighbors* $j \in \mathcal{N}^i$*, where only non-zero elements need to be transmitted*[9].
   Then controller i constructs the neighborhood image $\boldsymbol{x}^{(p-1)|\mathcal{N}^i}$ *using formula* (29).
2) ***Update intermediate variables*** γ_l ***in parallel***
   Each local controller i updates γ_l *for each* $l \in L_i$*, using* (32) *or* (33).
3) ***Communications to get the updated intermediate variables***
   Each local controller i gets $\gamma_l^{(p)}, l \in L_{\mathcal{N}^i}$ *that are updated by controllers in the neighborhood of i.*
4) ***Update main variables in parallel***
   Each local controller i computes an assumed neighborhood image *of* $\boldsymbol{x}$*:*

$$\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i} = \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \tag{35}$$

   Then controller i constructs the new self-image*, using* (34).

---

[8] Checking the termination criteria in a distributed fashion requires a dedicated logic scheme, several schemes were described in [29, Chapter 8].

[9] Since $\mathbf{x}^{(p-1)|i}$ only has a few non-zero elements, which are $u_0^{i,(p-1)}, \ldots, u_{N-1}^{i,(p-1)}, x_1^{i,(p-1)}, \ldots, x_N^{i,(p-1)}$, only these values need to be transmitted by controller i to reduce communications.

5) ***Check the local termination criteria in parallel***
Each local controller checks the local termination criteria. If local termination criteria are satisfied, the algorithm stops, otherwise go to step 1) to start a new iteration.

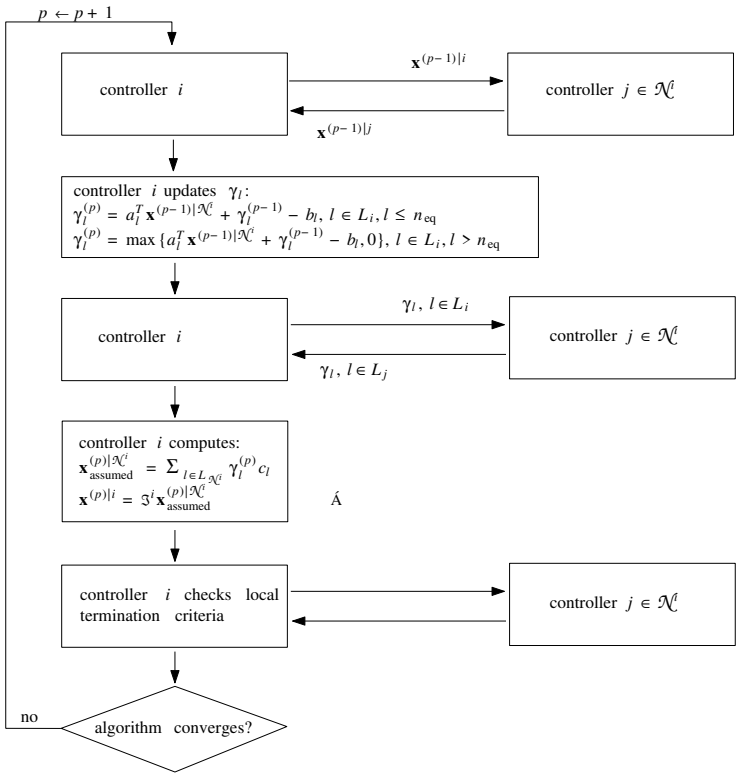

Figure 2: Computation and communication flow-chart of controller i in each iteration of Algorithm 4.4. Controller i only needs to communicate with its neighbor $j \in \mathcal{N}^i$.

In Algorithm 4.4, the activities of one local controller can be demonstrated by the diagram in Figure 2. The diagram clearly shows that in the distributed algorithm, each local controller i only communicates with its neighbors $j \in \mathcal{N}^i$, enabling implementation of the method in a distributed setting. The properties of the distributed algorithm will be discussed in the following subsections.

4.1.3. Proof of equivalence to Han's algorithm using a global update scheme

In Algorithm 3.4, at step 2), the centralized variable $\mathbf{x}^{(p)}$ is updated via a global update scheme. In Algorithm 4.4, by the local update scheme we obtain $\mathbf{x}^{(p)|i}$ for $i = 1, \ldots, M$. The equivalence of these two algorithms is stated in the following proposition:

Proposition 4.5. *Applying Algorithms 3.4 and 4.4 to the same problem* (8) *with the same parameter α, at any iteration p, the following statements hold:*

a) $\gamma_l^{(p)}$ are the same in Algorithms 3.4 and 4.4, for all $l \in \{1, \ldots, s\}$.

b) $\boldsymbol{x}^{(p)} = \sum_{i=1}^{M} \boldsymbol{x}^{(p)|i}$*, in which* $\boldsymbol{x}^{(p)}$ *is calculated in Algorithm 3.4 while* $\boldsymbol{x}^{(p)|i}, i = 1, \ldots, M$ *are calculated in Algorithm 4.4.*

Hence, Algorithm 3.4 and Algorithm 4.4 are equivalent.

Proof: The proposition will be proved by induction.

It is clear that properties a) and b) hold for $p = 0$.

Now consider iteration p, and assume that the properties a) and b) hold for all iterations before iteration p.

First, we prove property a). For any l and i such that $l \in L_i$, we have:

$$\begin{aligned} a_l^T \mathbf{x}^{(p-1)} &= a_l^T \sum_{j=1}^{M} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} \\ &= a_l^T \left(\sum_{j \in \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} + \sum_{j \notin \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} \right) \end{aligned} \tag{36}$$

Due to the definition of *neighborhood*, a subsystem outside $\mathcal{N}^i$ does not have any coupled constraints with subsystem i. Therefore, $a_l^T \sum_{j \notin \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} = 0$, which - in combination with (29) - leads to:

$$a_l^T \mathbf{x}^{(p-1)} = a_l^T \sum_{j \in \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} = a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} \tag{37}$$

Equation (37) then guarantees that $\gamma_l^{(p)}$ computed at step 1) of Algorithm 3.4 and at step 2) of Algorithm 4.4 are the same.

Now consider property b), where the main argument is the following: The same set of $\gamma_l^{(p)}$ and c_l are used for updating i's variables in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ (at step 4 of Algorithm 4.4) and in $\mathbf{x}^{(p)}$ (at step 2 of Algorithm 3.4). Thus each vector of local update $\mathbf{x}^{(p)|i}$, which contains values of i's variables selected from $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$, is a part of the centralized update $\mathbf{x}^{(p)}$.

More specifically, we can express the formula of $\mathbf{x}^{(p)|i}$ computed in Algorithm 4.4 as

$$\begin{aligned} \mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} &= \mathfrak{I}^i \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \\ \Rightarrow \sum_{i=1}^{M} \mathbf{x}^{(p)|i} &= \sum_{i=1}^{M} \mathfrak{I}^i \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \end{aligned} \tag{38}$$

Note that in the following equations, $\mathbf{x}^{(p)}$ refers to the update of the decision variable computed by (25) in Algorithm 3.4, which we can express as

$$\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathfrak{I}^i \mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathfrak{I}^i \sum_{l=1}^{s} \gamma_l^{(p)} c_l \tag{39}$$

in which the first equality is due to the relation (27), the second equality is from (25).

Recall that c_l has the same structure as a_l, and if $l \notin L_{\mathcal{N}^i}$ then a_l and c_l do not have any non-zero values at the positions associated with variables of subsystem i. Therefore

$$\mathfrak{I}^i \sum_{l=1}^{s} \gamma_l^{(p)} c_l = \mathfrak{I}^i \left(\sum_{l \notin L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l + \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \right) = \mathfrak{I}^i \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \tag{40}$$

This equality shows that (39) and (38) are equivalent, thus proving the equality in property b): $\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathbf{x}^{(p)|i}$. □

The equivalence of Algorithms 3.4 and 4.4 implies that problem (8) can be solved using Algorithm 4.4. This allows us to implement a DMPC scheme using Algorithm 4.4 that does not need global communications.

4.1.4. Properties of the distributed MPC controller

Convergence, feasibility, and stability properties of the DMPC scheme using Algorithm 4.4 are established by the following propositions:

Proposition 4.6. *Assume that* ($\mathcal{P}$) *has a feasible solution. Then Algorithm 4.4 asymptotically converges to the centralized solution of* ($\mathcal{P}$) *at each sampling step.*

Proof: In [24] it is shown that Han's method is guaranteed to converge to the centralized solution of the convex quadratic program (8) under the conditions that $q(\mathbf{x})$ is uniformly convex and differentiable on $\mathbb{R}^{n_\mathbf{x}}$ and (8) has a feasible solution. Due to the positive definiteness of Q_i and R_i, and the assumption that ($\mathcal{P}$) has a feasible solution, such conditions hold for the quadratic problem (8). Moreover, Algorithm 4.4 is equivalent to Han's method for the problem (8). Hence, the distributed scheme in Algorithm 4.4 converges to the centralized solution of (8), which is the same as ($\mathcal{P}$). □

Proposition 4.7. *Assume that at every sampling step, Algorithm 4.4 asymptotically converges. Then the DMPC scheme is recursively feasible and stable.*

Proof: By letting Algorithm 4.4 converge at every sampling step, the centralized solution of ($\mathcal{P}$) is obtained. Recursive feasibility and stability is guaranteed as a consequence of centralized MPC with a terminal point constraint, as shown in [3] and [26]. □

It is also worth to address the conservativeness of the MPC formulation using the terminal point constraint $x_N = 0$, which would reduce the domain of attraction of MPC. However, this issue is not related to Han's method. In fact, the distributed Han's method is able to handle optimization problems of other MPC formulations, given that the cost function has a sparse coupling structure. Note that finding other MPC formulations with a sparse coupling structure is not straightforward, we will discuss this problem in Section 6.

4.2. Distributed version of Han's method with scaled step size

A disadvantage of Han's method (and its distributed version) is the slow convergence rate, due to the fact that it is essentially a projection method to solve the dual problem of (8). Moreover, Han's (distributed) method uses zeros as the initial guess, which prevents warm starting of the algorithm by choosing an initial guess that is close to the optimizer. Therefore, we need to modify the method to achieve a better convergence rate.

In this section, we present two modifications of the distributed version of Han's method:

- Scaling of the step sizes related to dual variables by using heterogeneous α_l for the update of each l-th dual variable instead of the same α for all dual variables.
- Use of nonzero initial guesses, which allows taking the current MPC solution as the start for the next sample step.

Note that the modified distributed algorithm is then not equivalent to the centralized algorithm anymore. There is no convergence proof for the modified distributed algorithm yet; this will be discussed in Section 6.

In order to implement the above modifications, the improved distributed version of Han's method is initialized similarly to the distributed algorithm in Section 4.1.1, except for the following procedures:

1. **Pre-computed invariant parameters**
   Each subsystem i computes and stores the following parameters throughout the control scheme:
   - For each $l \in L_i$: $\alpha_l = (k_\alpha)_l \alpha_0$, where k_α is the scaling vector. α_l acts as local step size regarding the l-th dual variable, and therefore k_α should be chosen such that the convergence rates of all s dual variables are improved. The method to choose k_α will be discussed in Remark 4.9.
   - For each $l \in L_i$: $\bar{c}_l = \frac{-1}{a_l^T a_l} H^{-1} a_l$. We can see that $\bar{c}_l$ can be computed locally by a local controller with a priori knowledge of the parameter a_l and the weighting blocks on the diagonal of H that correspond to the non-zero elements of a_l.
2. **MPC step**
   At the beginning of the MPC step, the current states of all subsystems are measured. The sequences of predicted states and inputs generated in the previous MPC step are shifted forward one step, then we add zero states and zero inputs to the end of the shifted sequences. The new sequences are then used as the initial guess for solving the optimization problem in the current MPC step[10]. The initial guess for each subsystem can be defined locally. For subsystem i, denote the initial guess as $\mathbf{x}^{(0)|i}$. At the first MPC step, we have $\mathbf{x}^{(0)|i} = 0, \forall i$.
   The current state is plugged into the MPC problem, then we get an optimization problem of the form (8). This problem will be solved by the following modified distributed algorithm of Han's method.

Algorithm 4.8. ***Improved distributed algorithm for the MPC optimization problem***

Initialize with $p = 0$. Each subsystem i uses the initial guess as $\boldsymbol{x}^{(0)|i}$.
Next, for $p = 1, 2, \ldots$, the following steps are executed:

1) See step 1 of Algorithm 4.4.

2) See step 2 of Algorithm 4.4, except that for $p = 1$, each subsystem i computes the initial intermediate variables by[11]*:*

$$\gamma_l^{(1)} = a_l^T \left(\boldsymbol{x}^{(0)|\mathcal{N}^i} - \frac{\alpha_l}{s} H \boldsymbol{x}^{(0)|\mathcal{N}^i} \right) - b_l, \quad l \in L_i, l \leq n_{\text{eq}} \tag{41}$$

$$\gamma_l^{(1)} = \max \left\{ a_l^T \left(\boldsymbol{x}^{(0)|\mathcal{N}^i} - \frac{\alpha_l}{s} H \boldsymbol{x}^{(0)|\mathcal{N}^i} \right) - b_l, 0 \right\}, \quad l \in L_i, l > n_{\text{eq}} \tag{42}$$

[10] The idea of using previously predicted states and inputs for initialization is a popular technique in MPC [4]. Especially with Han's method, whose convergence rate is slow, an initial guess that is close to the optimal solution will be very helpful to reduce the number of iterations.

[11] The intermediate variables are constructed following the formulas (20)–(21) with replacing the common α by α_l for each $l \in \{1, \ldots, s\}$, where we implicitly use $\mathbf{y}_l^{(0)} = \frac{1}{s}\mathbf{y}^{(0)}, \forall l \in \{1, \ldots, s\}$ and $\mathbf{y}^{(0)} = H\mathbf{x}^{(0)}$. Also note that since a_l only involves neighboring subsystems and H is block-diagonal, the computation $a_l^T \left(\mathbf{x}^{(0)} - \frac{\alpha_l}{s} H \mathbf{x}^{(0)} \right)$ only uses values from neighboring subsystems, similarly to the argument for (37).

3) See step 3 of Algorithm 4.4.

4) See step 4 of Algorithm 4.4 but with a different formula to update the assumed neighborhood image for each i:

$$\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i} = \sum_{l \in L_{\mathcal{N}^i}} \frac{1}{\alpha_l} \gamma_l^{(p)} \bar{c}_l \tag{43}$$

5) See step 5 of Algorithm 4.4.

When the iterative procedure finishes, each subsystem applies the first input $u_0^{i,(p)}$, then waits for the next state measurement to start a new MPC step.

Remark 4.9. *The main improvement of Algorithm 4.8 over Algorithm 4.4 is the improved convergence speed, which heavily depends on a good choice of the scaling vector k_α. We have observed that the convergence rate of some dual variables under the responsibility of a subsystem i will affect the convergence rate of dual variables under the responsibility of its neighbors in $\mathcal{N}^i$. Therefore the choice of scaling vector should focus on improving the convergence rate of dual variables that appear to converge more slowly. In our case, we rely on the Hessian to find the scaling vector. Specifically, for each subsystem i, let $\bar{h}_i$ denote the average weight of its variables (i.e. average of entries related to i's states and inputs in the diagonal of the Hessian). We then choose the scaling factor $(k_\alpha)_l = 1/\bar{h}_i$, for all $l \in L_i$. We also multiply the scaling vector k_α with a factor $\theta \in (0, 1)$ for enlarging the step sizes of all dual variables. In the first MPC step, we start tuning with $\theta \simeq 1$ and gradually reduce θ until it causes the algorithm to diverge, then we stop and choose the smallest θ such that the algorithm still converges.*

The choice of the scaling vector depends on the structure of the centralized optimization problem, thus we only need to choose it once in the first MPC step. Then for the next MPC steps, we can re-use the same scaling vector.

The efficiency of Algorithm 4.8 will be demonstrated in the example of irrigation canal control, which is presented in the next section.

5. Application of Han's method for distributed MPC in canal systems

5.1. The example canal system

The novel DMPC approach is applicable to a wide range of large-scale systems which could be modeled in the LTI form as described in Section 2. In this section, we demonstrate its application in an example control problem, where the objective is to regulate the water flows in a system of irrigation canals. Irrigation canals are large-scale systems, consisting of many interacting components, and spanning vast geographical areas. For the most efficient and safe operation of these canals, maintaining the levels of the water flows close to pre-specified reference values is crucial, both under normal operating conditions as well as in extreme situations. Manipulation of the water flows in irrigation canals is typically done using devices such as pumps and gates.

The example irrigation canal to be considered is a 4-reach canal system as illustrated in Figure 3. In this system, water flows from an upstream reservoir through the reaches, under the control of 4 gates and a pump at the end of the canal system that discharges water.

The control design is based on the master-slave control paradigm, in which the master controllers compute the flows through the gates, while each slave controller uses the local control actuators to guarantee the flow set by the master controller [30]. We will use the new DMPC method to design the master controllers.

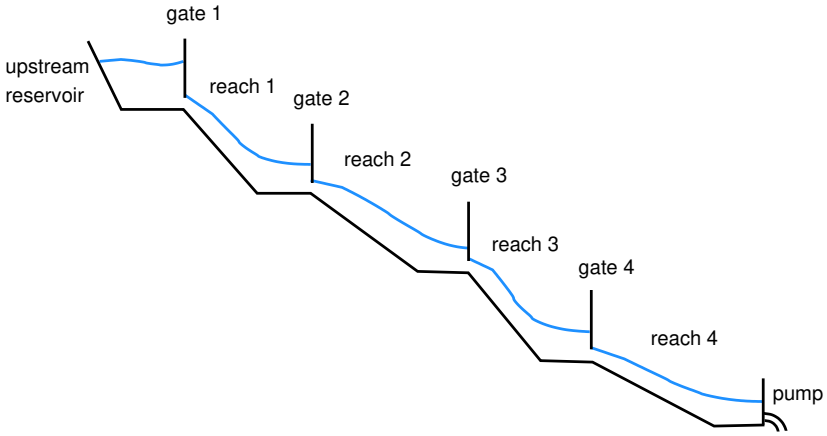

Figure 3: The example canal system

5.2. *Modeling the canal*

The canal system is divided into 4 subsystems, each of which corresponds to a reach and also includes the local controller at the upstream gate of the reach. The 4th subsystem has one more controller, corresponding to the pump at its downstream end.

We use a simplified model for each subsystem as illustrated in Figure 4, and then obtain the overall model by connecting the subsystem models. A subsystem is approximately modeled by a reservoir with upstream in-flow and downstream out-flow.

The discrete-time model of reach i is represented by:

$$h^i_{k+1} - h^i_k = \frac{T_s}{A^i_s}\left[\left(Q^i_{\text{in}}\right)_k - \left(Q^i_{\text{out}}\right)_k\right] \tag{44}$$

where superscript i represents the subsystem index, subscript k is for the time index, T_s is the sampling time, h is the downstream water level of the reach, A_s is the water surface (i.e. the volume of reservoir $= h \cdot A_s$), Q_{in} and Q_{out} are the in-flow and the out-flow of the canal which are measured at the upstream and downstream ends, respectively. Denote the flow passing i-th gate by q^i, and the flow passing the pump by p^4. Due to the mass conservation law, we have $Q^i_{\text{out}} = Q^{i+1}_{\text{in}} = q^{i+1}$, for $i = 1, 2, 3$, and $Q^4_{\text{out}} = p^4$.

In order to derive local dynamics, we choose input and state vectors of subsystem i as

$$x^i_k = h^i_k$$
$$u^i_k = \begin{cases} q^i_k & , \quad i = 1, 2, 3 \\ \begin{bmatrix} q^i_k \\ p^i_k \end{bmatrix} & , \quad i = 4 \end{cases}$$

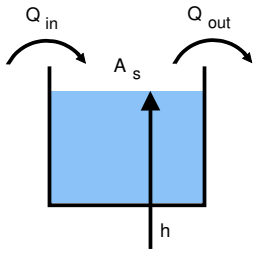

Figure 4: Model of a reach

The dynamics of each subsystem can be represented by a discrete-time, linear time-invariant model of the form (1) with the state-space matrices:

$$
\begin{aligned}
&A^{ii} = \ 1 \ , \quad i = 1, \ldots, 4; \quad A^{ij} = \ 0 \ , \quad i \neq j \\
&B^{ii} = \ T_\mathrm{s}/A_\mathrm{s}^i \ , \quad i = 1, 2, 3; \quad B^{44} = \begin{bmatrix} T_\mathrm{s}/A_\mathrm{s}^4 & -T_\mathrm{s}/A_\mathrm{s}^4 \end{bmatrix} \\
&B^{i(i+1)} = \ -T_\mathrm{s}/A_\mathrm{s}^i \ , \quad i = 1, 2; \quad B^{34} = \begin{bmatrix} -T_\mathrm{s}/A_\mathrm{s}^4 & 0 \end{bmatrix} \\
&B^{ij} = 0 \ , \quad i = 1, 2, 3, \quad j \notin \{i, i+1\}.
\end{aligned}
$$

5.3. Simulation results

DMPC methods are applied to the regulation problem of the simulated canal system described in previous subsections using sampling time $T_s = 240s$, with a perturbed initial state. We use the distributed Han's method with and without the modifications described in Section 4, and compare the results. Figure 5 shows the convergence of the distributed solutions to the centralized solution for the problem. Starting from the same initial guess in the first MPC step, i.e. all variables are initialized with zeros, the distributed algorithm with modifications achieves a better convergence rate, allowing the distributed optimization to converge within an acceptable number of iterations. Similar results were also achieved for the next MPC steps, when we simulate the closed-loop MPC and let the distributed solutions converge to the centralized solution at every step, with maximally 100 iterations per step.

6. Discussion and outlook on future research

Two distributed versions of Han's method have been described in Section 4, followed by a short demonstration of their usage in Section 5. Although these algorithms help to implement Han's method in a distributed setting for MPC, there are still some theoretical issues that need to be addressed.

Firstly, the proposed distributed algorithms deal with quadratic programs only. Although many MPC problems for linear time-invariant systems are formulated as quadratic programs, there are other variants that use different objective functions, and nonlinear MPC would also yield more complicated optimization problems than quadratic programs. With such problems, we might not be able to implement Han's parallel method in a distributed fashion. This issue motivates the research on other decomposition methods that can handle more general problems, e.g. convex problems with linear or decoupled nonlinear constraints.

As noted in Section 4.1, the MPC formulation in this paper employs the terminal constraint $x_N = 0$, which is a conservative approach. In case we want to use less

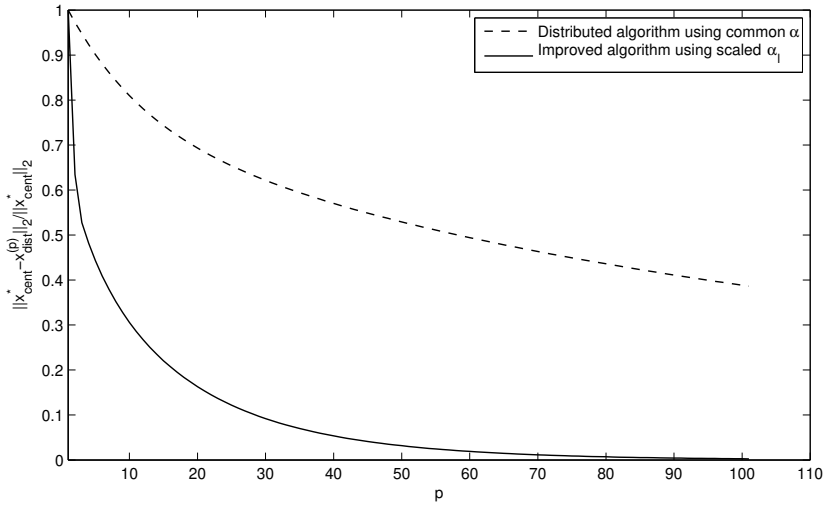

Figure 5: Comparison of convergence rates of two distributed versions of Han's method for the first sampling time step (k=1)

conservative MPC, e.g. MPC with a terminal constraint set and a terminal controller, we need to find a separable terminal penalty function and local terminal constraint sets. However, to the authors' best knowledge, there is still no distributed scheme available to construct local terminal constraint sets and local terminal controllers (and also the terminal penalty matrix that is solution of the Riccati equation), other than assuming them to be completely decoupled. Therefore, although distributed Han's method can also be applied to any uniformly convex QP problem with sparse coupling structure, it requires further research on MPC formulations that have such optimization problems.

In general, Han's method has a slow convergence rate due to its iterative projection nature, which is inherited by Algorithm 4.4. Since the feasibility and stability properties are derived upon convergence of the algorithm within each sampling step, we need to speed up the convergence of this method. The distributed version of Han's method with scaling can improve the convergence rate significantly, as illustrated in Section 5. However, its proof of convergence is still lacking. We observe that in setups that are more complicated, the proposed method to choose the scaling vector does not always work well (sometimes after several sample steps, the algorithm does not converge anymore). Due to the requirement not to have global communications, it is difficult to adjust the scaling vector during the iteration to reach convergence. Therefore speeding up Han's method while providing a proof for convergence is still an open issue, and we may use a coordinator at a higher level of hierarchy that has global communication capabilities to tackle this issue.

Another issue is due to the formulation of the optimization problem for MPC, where we keep both inputs and states as variables of the centralized optimization problem and do not eliminate the states using the dynamic model equations. This formulation is advantageous in distributed MPC because the Hessian will then keep a block diagonal structure, and the neighborhood of each subsystem will only contains its direct neighbors (the neighborhood would be greatly extended if we eliminate the states in the optimization problem). However, using states as variables requires considering the dynamical equations as equality constraints of the optimization problem, and the existence of equality constraints typically requires an exact solution in order to guarantee

feasibility. Since Han's method converges asymptotically, we may not be able to get the exact optimal multipliers in real-time, and then the corresponding primal iterates would not be guaranteed to be feasible. In general, most dual decomposition methods do not provide primal feasible solutions before reaching the dual optimal solutions, so this feasibility issue also applies to other dual decomposition methods.

In future research, we will also study dual decomposition methods that can provide primal feasible solutions in a finite number of iterations. In order to tackle the convex problem, we intend to make use of the subgradient schemes proposed in [31] and [32], which extend the traditional primal recovery schemes for linear programs [33, 34]. With this approach, the standard proof for MPC stability, which is based on optimality, will not be obtained. Therefore, we need to prove stability of suboptimal MPC, which can be based on the theorems proposed in [35], i.e. showing the reduction of the cost function (acting as a Lyapunov function) associated with the feasible solution. We intend to use the bounds of suboptimality of the subgradient iterations to show the decreasing property of the cost function. Finding such bounds that are suitable for proving suboptimal MPC stability is still an open question.

7. Conclusions

A decomposition approach based on Fenchel's duality and Han's parallel method has been developed in this paper, resulting in two distributed algorithms that are applicable to DMPC. The first distributed algorithm generates updates that are equivalent with those computed globally by Han's method for definite quadratic problems, and therefore it has the same convergence property as Han's method. Moreover, feasibility and stability of DMPC are achieved upon convergence of the iterations. The second distributed algorithm aims to improve the convergence speed by using scaled step sizes and nonzero initial guess. The new methods have been applied to an example irrigation canal network, demonstrating their applicability for water network and other large-scale networked systems. We have also summarized open issues of using Han's method and other dual decomposition methods for MPC, including the topics of distributed formulation, convergence rate, primal feasibility, and stability of MPC. These issues were used to recommend future research directions.

Acknowledgment. Research supported by the European 7th framework STREP project "Hierarchical and distributed model predictive control", contract number INFSO-ICT-223854.

References

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, July 2003.

[2] J. M. Maciejowski, *Predictive Control with Constraints*. Harlow, England: Prentice Hall, 2002.

[3] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, June 2000.

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

[5] D. Jia and B. Krogh, "Distributed model predictive control," in *American Control Conference*, vol. 4, pp. 2767–2772, 2001.

[6] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, pp. 44–52, Feb. 2002.

[7] J. B. Rawlings and B. T. Stewart, "Coordinating multiple optimization-based controllers: New opportunities and challenges," *Journal of Process Control*, vol. 18, pp. 839–845, Oct. 2008.

[8] R. Scattolini, "Architectures for distributed and hierarchical model predictive control - A review," *Journal of Process Control*, vol. 19, pp. 723–731, May 2009.

[9] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, pp. 549–558, Apr. 2006.

[10] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, pp. 2105–2115, Dec. 2006.

[11] A. Richards and J. How, "Robust distributed model predictive control," *International Journal of Control*, vol. 80, pp. 1517–1531, Sept. 2007.

[12] A. Venkat, I. Hiskens, J. Rawlings, and S. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Transactions on Control Systems Technology*, vol. 16, pp. 1192–1206, Nov. 2008.

[13] D. Jia and B. Krogh, "Min-max feedback model predictive control for distributed control with communication," in *American Control Conference*, vol. 6, pp. 4507–4512, 2002.

[14] A. Alessio and A. Bemporad, "Decentralized model predictive control of constrained linear systems," in *European Control Conference*, pp. 2813–2818, 2007.

[15] A. Alessio and A. Bemporad, "Stability conditions for decentralized model predictive control under packet drop communication," in *American Control Conference*, pp. 3577–3582, 2008.

[16] S. Li, Y. Zhang, and Q. Zhu, "Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem," *Information Sciences*, vol. 170, pp. 329–349, Feb. 2005.

[17] E. Camponogara and S. Talukdar, "Distributed model predictive control: Synchronous and asynchronous computation," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 37, pp. 732–745, Sept. 2007.

[18] I. Necoara, D. Doan, and J. Suykens, "Application of the proximal center decomposition method to distributed model predictive control," in *IEEE Conference on Decision and Control*, pp. 2900–2905, Dec. 2008.

[19] I. Necoara and J. Suykens, "Application of a smoothing technique to decomposition in convex optimization," *IEEE Transactions on Automatic Control*, vol. 53, pp. 2674–2679, Dec. 2008.

[20] M. Mercangoz and F. J. Doyle III, "Distributed model predictive control of an experimental four-tank system," *Journal of Process Control*, vol. 17, pp. 297–308, Mar. 2007.

[21] R. R. Negenborn, P. J. van Overloop, T. Keviczky, and B. De Schutter, "Distributed model predictive control for irrigation canals," *Networks and Heterogeneous Media*, vol. 4, pp. 359–380, June 2009.

[22] M. Arnold, R. R. Negenborn, G. Andersson, and B. De Schutter, "Distributed predictive control for energy hub coordination in coupled electricity and gas networks," in *Intelligent Infrastructures* (R. R. Negenborn, Z. Lukszo, and H. Hellendoorn, eds.), pp. 235–273, Dordrecht, The Netherlands: Springer, 2010.

[23] D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. De Schutter, "A distributed version of Han's method for DMPC using local communications only," *Control Engineering and Applied Informatics*, vol. 11, pp. 6–15, Sept. 2009.

[24] S.-P. Han and G. Lou, "A parallel algorithm for a class of convex programs," *SIAM Journal on Control and Optimization*, vol. 26, pp. 345–355, Mar. 1988.

[25] M. D. Doan, T. Keviczky, and B. De Schutter, "An improved distributed version of Han's method for distributed MPC of canal systems," in *12th symposium on Large Scale Systems: Theory and Applications*, July 2010.

[26] S. S. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback control laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations," *Journal of Optimization Theory and Applications*, vol. 57, pp. 265–293, May 1988.

[27] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton University Press, 1970.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, MA: Cambridge University Press, 2004.

[29] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[30] J. Schuurmans, A. Hof, S. Dijkstra, O. H. Bosgra, and R. Brouwer, "Simple water level controller for irrigation and drainage canals," *Journal of Irrigation and Drainage Engineering*, vol. 125, pp. 189–195, July 1999.

[31] A. Nedic and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, pp. 1757–1780, Nov. 2009.

[32] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical Programming*, vol. 120, pp. 221–259, Aug. 2009.

[33] T. Larsson and Z. Liu, "A Lagrangean relaxation scheme for structured linear programs with application to multicommodity network flows," *Optimization*, vol. 40, pp. 247–284, 1997.

[34] H. Sherali and G. Choi, "Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs," *Operations Research Letters*, vol. 19, pp. 105–113, Sept. 1996.

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