2021 IEEE Conference on Control Technology and Applications (CCTA).
August 8-11, 2021. San Diego, California

2021 IEEE Conference on Control Technology and Applications (CCTA).
August 8-11, 2021. San

2021 IEEE Conference on Control Technology and Applications (CCTA) | 978-1-6654-3643-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCTA48906.2021.9658733

A sensitivity-based distributed model predictive control algorithm for nonlinear continuous-time systems*

Hartwig Huber[1] and Knut Graichen[1]

***Abstract*— Model predictive control (MPC) is a frequently used control technique. An extension of MPC is distributed MPC (DMPC) that can be used to meet restrictions in computation time and make flexible system reconfiguration possible. This contribution presents a DMPC algorithm, which uses sensitivities that contain information about the influence of a control action on neighboring agents. Three different ways of calculation are presented. The algorithm itself performs local optimization and exchanges sensitivities on agent level until a convergence criterion is met. The method is applied to several examples to demonstrate its performance, including trajectories and time analysis. In particular, it is shown that the computation time on agent level is almost constant for an increasing system size.**

I. INTRODUCTION

Model predictive control (MPC) is a modern control technique that solves an optimal control problem (OCP) on a finite horizon. The obtained control trajectory is then applied to the plant for one sampling step before the horizon is shifted and the OCP is solved again based on the new measurement (see e.g. [1], [2], [3], [4]). A big advantage of this method is that input and state constraints can be considered directly in the optimization problem. A broad spectrum of theoretical and application-oriented contributions to this method can be found in the literature. Regardless of the approach, for general problems, numerical methods are needed to solve the optimization problem. However, despite recent advances in the field of processing units, for computationally demanding systems the calculation time still exceeds the sampling time limit. Distributed model predictive control (DMPC) seeks to tackle this problem by splitting up the OCP, solving the remaining (smaller) problems in parallel and exchanging necessary information between the subsystems. Furthermore, it enables the possibility of a flexible reconfiguration of the network [6], [7]. The main point of such algorithms is to force the solution of the local problems to converge to the central solution. To this end, a variety of algorithms have been proposed so far, some of which are mentioned hereafter.

The alternating direction method of multipliers (ADMM) [8] became popular in recent years and can be used to solve distributed static optimization problems. The method itself is based on dual decomposition (see e.g. [9]) as well as the augmented Lagrangian method and assumes strong duality of the optimization problem (see e.g. [10]). Numerous contributions make use of this scheme, such as those mentioned in the short overview in [11]. DMPC applications of ADMM can be found e.g. in [12], [13], [14]. A variety of alternative approaches to ADMM have been proposed so far, such as [15] utilizing an alternating direction inexact Newton method, which is able to handle non-convex OCPs. Moreover, the algorithms presented in [16] aim to approximate the exact Newton direction in a distributed manner. The contribution [17] comprises a Jacobi algorithm to solve distributed convex optimization problems.

A different DMPC scheme uses interaction operators, also called *sensitivities*, see [18]. In contrast to ADMM, this approach belongs to the family of primal decomposition methods and thus no dual problem must be solved. An advantage of primal methods is, that strong duality is not required to hold. Concerning sensitivity-based DMPC, in [19] the continuous-time linear case, in [20] the linear discrete-time case and in [21] an introductory nonlinear discrete-time case are investigated. All these contributions follow the same idea that can be traced back to [18]. However, none of them provides an efficient technique to evaluate the sensitivities in a general manner. Furthermore, the approach in [21] heavily depends on the discretization of the optimal control problem and therefore also loses insight into the the nonlinear system structure.

The main contribution of this paper is to propose a solution to these issues and formulate a distributed sensitivity-based algorithm for nonlinear continuous-time model predictive control. In fact, it is possible to take advantage of the classical optimal control theory to compute the sensitivities in an adjoint-based straightforward manner. Besides, three different versions can be formulated, depending on how the couplings of the subsystems are handled. To the authors' knowledge, no formulation of a sensitivity-based DMPC algorithm for the continuous-time nonlinear case has been considered so far. The paper has the following structure: In the next section, the problem under consideration and the notation are discussed. Then in Section III the distributed algorithm is presented, in Section IV alternate formulations and extensions are described, while finally in Section V simulation results of several examples are given.

II. PROBLEM FORMULATION

In the field of distributed control, systems are usually described by means of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$. Each vertex, also called *agent* in the DMPC sense, represents a subsystem, usually called *local* system. The edges, embody the couplings between the agents. In

*This work was supported by German Federal Ministry of Economic Affairs and Energy under grant number 03ET1495B

[1]The authors are with the Chair of Automatic Control, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

general, the local dynamics of agent $i \in \mathcal{V}$ depend on states and inputs of its neighbors $j \in \mathcal{N}^i$. The neighborhood $\mathcal{N}^i$ is the set of vertices $j \in \mathcal{V}$ directly connected to the subsystem $i \in \mathcal{V}$. Using this convention, we consider the *central* OCP

$$\min_{\boldsymbol{u}} \quad \sum_{i\in\mathcal{V}} \left(V_i(\boldsymbol{x}_i(T)) + \int_0^T l_i(\boldsymbol{x}_i, \boldsymbol{u}_i)\,\mathrm{d}t \right) \tag{1a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 = \tilde{\boldsymbol{x}}(t_l) \tag{1b}$$

$$\boldsymbol{u}_i \in [\boldsymbol{u}_i^-, \boldsymbol{u}_i^+], \quad i \in \mathcal{V}, \tag{1c}$$

with state $\boldsymbol{x}^\top(t) = [\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_{|\mathcal{V}|}^\top] \in \mathbb{R}^n$, input $\boldsymbol{u}^\top(t) = [\boldsymbol{u}_1^\top, \dots, \boldsymbol{u}_{|\mathcal{V}|}^\top] \in \mathbb{R}^m$ and initial state $\boldsymbol{x}_0 \in \mathbb{R}^n$. Furthermore, $V_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is the (local) terminal cost function and $l_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \times \mathbb{R} \to \mathbb{R}$ the (local) integral cost function, where $n_i \in \mathbb{N}$ and $m_i \in \mathbb{N}$ are the dimensions of the local state $\boldsymbol{x}_i$ and local input $\boldsymbol{u}_i$ respectively. The central dynamics are given by (1b), while $\boldsymbol{f}(\cdot)$ is assumed to be differentiable w.r.t. its arguments. Local input constraints (1c) are considered in the form of box constraints. It is furthermore assumed, that a solution to the OCP (1) exists. The local dynamics of agent i can be written as

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}), \;\; \boldsymbol{x}_i(0) = \tilde{\boldsymbol{x}}_i(t_l), \tag{2}$$

where $\boldsymbol{x}_{\text{-}i}$ and $\boldsymbol{u}_{\text{-}i}$ denote all remaining trajectories of states and inputs respectively, i.e.

$$\boldsymbol{x}_{\text{-}i} = \left[\boldsymbol{x}_j\right]_{j\in\mathcal{N}^i}, \quad \boldsymbol{u}_{\text{-}i} = \left[\boldsymbol{u}_j\right]_{j\in\mathcal{N}^i}. \tag{3}$$

Note that $(\boldsymbol{x}_i, \boldsymbol{x}_{-i})$ involves a re-ordering and omission of some elements of $\boldsymbol{x}$. Using the same function representation as in (2) therefore represents a slight abuse of notation, which is tolerated for the sake of readability.

In terms of MPC, (1) is solved repeatedly in each sampling step l over a finite horizon T starting from the current measurement $\tilde{\boldsymbol{x}}(t_l)$. The resulting optimal control trajectory $\boldsymbol{u}^*(t)$ is then applied to the system for one sampling step. In the next sampling step, a new measurement is obtained, the horizon is shifted and (1) is solved again (see e.g. [1]). Note that the cost functional in (1a) is a sum over local cost functionals.

Throughout the paper, a partial derivative w.r.t. $\boldsymbol{x}$ of a function $\boldsymbol{f}(\boldsymbol{x})$ is denoted by $\partial_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})$ and the Gâteaux directional derivative of a functional $F(\boldsymbol{h})$ in direction $\delta\boldsymbol{h}$ is expressed by $\delta F(\boldsymbol{h}; \delta\boldsymbol{h})$. The superscript $[k]$ represents the iteration index. Furthermore, implicit time dependency of a function will be denoted by $[t]$, whenever it is convenient. The explicit time dependency of trajectories, e.g. $\boldsymbol{x}(t)$, is omitted in most cases for the sake of compactness.

III. Sensitivity-based distributed optimization algorithm

The purpose of a DMPC algorithm is to solve (1) in a distributed fashion. The main idea of the proposed algorithm is to use *sensitivities* that contain information about the expected change in costs of a specific subsystem w.r.t. an external trajectory. Hence, the sensitivity information is a measure in which direction an agent should alter its trajectories in order to improve the performance of a neighbor. Thus, by altering the structure of the subsystems, consequently the calculation of the sensitivities changes. In the case of static optimization, the sensitivity is equivalent to the gradient of the cost function w.r.t. an external trajectory (see e.g. [20]). However, in the continuous-time case, the respective calculation is more involved.

A. Local OCP with central dynamics - version (i)

One choice for the local problems to be solved on agent level is a straightforward extension of [20], where each agent accounts for the central dynamics but only optimizes its local inputs. The local problem of agent $i \in \mathcal{V}$ therefore reads

$$\min_{\boldsymbol{u}_i} \; V_i(\boldsymbol{x}_i(T)) + \int_0^T l_i(\boldsymbol{x}_i, \boldsymbol{u}_i)\,\mathrm{d}t + \sum_{j\in\mathcal{N}^i} \delta J_j(\boldsymbol{u}_i^{[k]}; \boldsymbol{u}_i - \boldsymbol{u}_i^{[k]}) \tag{4a}$$

$$\text{s.t.} \;\; \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}^{[k]}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{4b}$$

$$\boldsymbol{u}_i \in [\boldsymbol{u}_i^-, \boldsymbol{u}_i^+] \tag{4c}$$

with

$$J_j(\boldsymbol{u}_i) := \Big\{ V_j(\boldsymbol{x}_j(T)) + \int_0^T l_j(\boldsymbol{x}_j, \boldsymbol{u}_j)\,\mathrm{d}t \;\; \text{s.t.} \;\; \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}^{[k]}), \boldsymbol{x}(0) = \boldsymbol{x}_0 \Big\}. \tag{5}$$

The Gâteaux directional derivatives $\delta J_j(\boldsymbol{u}_i^{[k]}; \boldsymbol{u}_i - \boldsymbol{u}_i^{[k]})$ in the local cost (4a) represent the sensitivities of the neighbors $j \in \mathcal{N}^i$ w.r.t. the control $\boldsymbol{u}_i$ of agent i. Furthermore, $\boldsymbol{u}_j^{[k]}$ is the control trajectory of neighboring agents in the previous iteration $[k]$. The expression $\boldsymbol{u}_i - \boldsymbol{u}_i^{[k]}$ describes the change of the control trajectory compared to the trajectory of the preceding iteration. Hence, by means of (4a), the objectives of the neighboring agents are accounted for in the local optimization.

B. Sensitivity calculation

Let's recall that a functional $F : \mathbb{X} \to \mathbb{R}$ is called Gâteaux differentiable at $\boldsymbol{h} \in \mathbb{X}$ if $\delta F(\boldsymbol{h}; \delta\boldsymbol{h})$ exists for all continuous directions $\delta\boldsymbol{h} \in \mathbb{X}$ in a Banach space $\mathbb{X}$. Then

$$\delta F(\boldsymbol{h}; \delta\boldsymbol{h}) := \lim_{\varepsilon\to 0} \frac{F(\boldsymbol{h} + \varepsilon\delta\boldsymbol{h}) - F(\boldsymbol{h})}{\varepsilon}, \tag{6}$$

is called the Gâteaux directional derivative of $F(\boldsymbol{h})$ at $\boldsymbol{h}$ in direction $\delta\boldsymbol{h}$ (see e.g. [22]).

The Gâteaux directional derivative in (4a) can be calculated conveniently using optimal control theory. By defining the Hamiltonian

$$H_j[t] := l_j(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{\lambda}_j^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_j, \boldsymbol{u}_{\text{-}j}^{[k]}), \tag{7}$$

the sensitivity $\delta J_j(\boldsymbol{u}_i; \delta\boldsymbol{u}_i)$ can be expressed as

$$\delta J_j(\boldsymbol{u}_i; \delta\boldsymbol{u}_i) = \int_0^T \left(\partial_{\boldsymbol{u}_i} H_j[t]\right)^\top \delta\boldsymbol{u}_i(t)\,\mathrm{d}t, \tag{8}$$

where $\boldsymbol{\lambda}_j \in \mathbb{R}^n$ is the solution of the adjoint system

$$\begin{aligned}\dot{\boldsymbol{\lambda}}_j = & -\partial_{\boldsymbol{x}} l_j(\boldsymbol{x}_j, \boldsymbol{u}_j) - \partial_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_j, \boldsymbol{u}_{\text{-}j}^{[k]})^\top \boldsymbol{\lambda}_j, \\ \boldsymbol{\lambda}_j(T) = & \, \partial_{\boldsymbol{x}} V_j(\boldsymbol{x}(T)),\end{aligned} \tag{9}$$

associated to OCP (4) of agent j. The derivation is given in Appendix A. Note that in the local problem (4) the direction $\delta \boldsymbol{u}_i$ is defined by the deviation $\boldsymbol{u}_i - \boldsymbol{u}_i^{[k]}$. An essential discovery at this point is that the sensitivity can be deduced from the Hamiltonian (7) of the respective local problem, which is evaluated locally by agent j together with the partial derivative of the Hamiltonian $\partial_{\boldsymbol{u}_i} H_j[t]$. In order to do this, the external trajectories $\boldsymbol{u}_{\text{-}j}^{[k]}$ need to be exchanged between the agents in every iteration. Since the local problems (4) consider central dynamics, the neighborhood $\mathcal{N}^i$ involves every other agent and therefore the trajectories are broadcast over the network. The fact that every agent considers the central dynamics (4b) presumably benefits the convergence behavior, but also increases the computational burden per agent for larger systems. Obviously, this circumstance is not intended for a distributed algorithm and therefore adaptations are introduced in Section IV. Note that the OCP formulation (4) furthermore assumes every agent to incorporate an input in its local dynamics, i.e. $\partial_{\boldsymbol{u}_i} \boldsymbol{f}(\cdot) \neq \boldsymbol{0}$, $\forall\, i \in \mathcal{V}$.

C. Algorithm

In view of the preceding declarations, Algorithm 1 summarizes the principle steps that consist of the two computation steps 4 and 6, where the sensitivities (8) are evaluated and local problems (4) are solved, respectively. These steps are carried out in parallel. The communication steps 5 and 7 are used to send and receive the locally computed sensitivities and trajectories. Note, that only coupled agents need to communicate with each other and therefore the graph mentioned in Section II also describes the communication topology. The algorithm is executed until a convergence criterion is satisfied. In the simplest case, it is based on the change in costs of two consecutive iterations, i.e. $|J_i^{[k+1]} - J_i^{[k]}| \leq \varepsilon$, $\forall\, i \in \mathcal{V}$, but other criteria may be applied as well.

Algorithm 1 represents a generalization of the linear discrete-time case [20] to nonlinear, continuous-time systems. Although [20] contains convergence results under certain conditions, the nonlinear, continuous-time case is more involved and therefore a fundamental proof of convergence is subject of future research.

Algorithm 1 Sensitivity-based distributed optimal control

1: Initialize agents $i \in \mathcal{V}$
2: Set $k = 0$
3: **do**
4: Every $i \in \mathcal{V}$: Calculate sensitivities (8) in parallel
5: Every $i \in \mathcal{V}$: Send sensitivities to all $j \in \mathcal{N}^i$
6: Every $i \in \mathcal{V}$: Compute new optimal trajectories $(\boldsymbol{u}_i^{[k+1]}, \boldsymbol{x}_i^{[k+1]})$ by solving the local OCPs (4)
7: Every $i \in \mathcal{V}$: Receive trajectories from agents $j \in \mathcal{N}^i$
8: Set $k \leftarrow k + 1$
9: **while** not converged or $k \leq k_{\max}$

IV. Algorithmic Adaptations

This section presents several adaptations of Algorithm 1. In particular, two alternative formulations of the local problems (4) are considered that avoid the problem of accounting for the global dynamics (4b) on agent level. Furthermore, measures to enhance the convergence behavior are presented.

A. Local OCP with local dynamics - version (ii)

An apparent solution to overcome the drawback of increased computation time is to use local dynamics in every agent instead of the centralized dynamics (4b). To this end, the local OCP formulation (4) of the algorithm is modified according to

$$\min_{\boldsymbol{u}_i} \; V_i(\boldsymbol{x}_i(T)) + \int_0^T l_i(\boldsymbol{x}_i, \boldsymbol{u}_i)\,\mathrm{d}t + \sum_{j \in \mathcal{N}^i} \delta J_j(\boldsymbol{u}_i^{[k]}; \boldsymbol{u}_i - \boldsymbol{u}_i^{[k]}) \tag{10a}$$

$$\text{s.t. } \dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i}^{[k]}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}^{[k]}),\; \boldsymbol{x}_i(0) = \boldsymbol{x}_{i,0} \tag{10b}$$

$$\boldsymbol{u}_i \in [\boldsymbol{u}_i^-, \boldsymbol{u}_i^+], \tag{10c}$$

with

$$J_j(\boldsymbol{u}_i) := \Big\{ V_j(\boldsymbol{x}_j(T)) + \int_0^T l_j(\boldsymbol{x}_j, \boldsymbol{u}_j)\,\mathrm{d}t \quad \text{s.t. } \dot{\boldsymbol{x}}_j = \boldsymbol{f}_j(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i}^{[k]}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}^{[k]}), \boldsymbol{x}_j(0) = \boldsymbol{x}_{j,0} \Big\}. \tag{11}$$

In contrast to the OCP (4), the dynamics (10b) in the OCP (10) regard only the local states $\boldsymbol{x}_i$ instead of the overall state $\boldsymbol{x}$. Thus, the cost functional (11) of the neighbors is depicted w.r.t. the control $\boldsymbol{u}_i$ and also considers the respective local dynamics. The remaining trajectories $(\boldsymbol{x}_j^{[k]}, \boldsymbol{u}_j^{[k]})$ of the neighborhood $j \in \mathcal{N}^i$ from the previous iteration are communicated to every agent i.

In this sense, by defining the Hamiltonian

$$H_j[t] := l_j(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{\lambda}_j^\top \boldsymbol{f}_j(\boldsymbol{x}_j, \boldsymbol{x}_{\text{-}j}^{[k]}, \boldsymbol{u}_j, \boldsymbol{u}_{\text{-}j}^{[k]}), \tag{12}$$

the sensitivity evaluates to

$$\delta J_j(\boldsymbol{u}_i; \delta \boldsymbol{u}_i) = \int_0^T \left(\partial_{\boldsymbol{u}_i} H_j[t]\right)^\top \delta \boldsymbol{u}_i(t)\,\mathrm{d}t, \tag{13}$$

where $\boldsymbol{\lambda}_j \in \mathbb{R}^{n_j}$ is the solution of the adjoint system

$$\begin{aligned}\dot{\boldsymbol{\lambda}}_j = & -\partial_{\boldsymbol{x}_j} l_j(\boldsymbol{x}_j, \boldsymbol{u}_j) - \partial_{\boldsymbol{x}_j} \boldsymbol{f}_j(\boldsymbol{x}_j, \boldsymbol{x}_{\text{-}j}^{[k]}, \boldsymbol{u}_j, \boldsymbol{u}_{\text{-}j}^{[k]})^\top \boldsymbol{\lambda}_j, \\ \boldsymbol{\lambda}_j(T) = & \, \partial_{\boldsymbol{x}_j} V_j(\boldsymbol{x}_j(T))\end{aligned} \tag{14}$$

associated to OCP (10) of agent j. An advantage of the OCP formulation (10) is that the computational effort of the individual agents per iteration k is independent of the overall system size. In addition, the amount of communication is reduced because the trajectories do not need to be broadcast

over the network. On the other hand, it requires not only that each subsystem must possess an individual control, i.e. $\partial_{\boldsymbol{u}_i} \boldsymbol{f}_i \neq \mathbf{0}$, but also that the neighbor dynamics can be affected as well, i.e. $\partial_{\boldsymbol{u}_i} \boldsymbol{f}_j \neq \mathbf{0}, \forall\, j \in \mathcal{N}^i$. If this condition is not fulfilled, the respective sensitivity (13) with (12) would always evaluate to zero, i.e. $\delta J_j(\boldsymbol{u}_i; \delta \boldsymbol{u}_i) \equiv 0$ and thus no coordination between the agents takes place in the algorithm.

B. Local OCP with interaction variables - version (iii)

In order to overcome the aforementioned restriction, the interaction variables

$$\boldsymbol{z}_i = \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{x}_i \end{bmatrix} \tag{15}$$

are introduced, that represent the influence of agent i on its respective neighborhood $\mathcal{N}^i$, i.e. the sensitivity is evaluated w.r.t. both states and controls. This idea was proposed in [19] for the linear case and is adopted in the following for the nonlinear case. Hence the local problem (10) becomes

$$\min_{\boldsymbol{u}_i} V_i(\boldsymbol{x}_i(T)) + \int_0^T l_i(\boldsymbol{x}_i, \boldsymbol{u}_i)\, \mathrm{d}t + \sum_{j \in \mathcal{N}^i} \delta J_j(\boldsymbol{z}_i^{[k]}; \boldsymbol{z}_i - \boldsymbol{z}_i^{[k]}) \tag{16a}$$

$$\text{s.t. } \dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i}^{[k]}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}^{[k]}), \quad \boldsymbol{x}_i(0) = \boldsymbol{x}_{i,0} \tag{16b}$$

$$\boldsymbol{u}_i \in [\boldsymbol{u}_i^-, \boldsymbol{u}_i^+], \tag{16c}$$

with

$$J_j(\boldsymbol{z}_i) := \Big\{ V_j(\boldsymbol{x}_j(T)) + \int_0^T l_j(\boldsymbol{x}_j, \boldsymbol{u}_j)\, \mathrm{d}t \quad \text{s.t. } \dot{\boldsymbol{x}}_j = \boldsymbol{f}_j(\boldsymbol{x}_i, \boldsymbol{x}_{\text{-}i}^{[k]}, \boldsymbol{u}_i, \boldsymbol{u}_{\text{-}i}^{[k]}), \boldsymbol{x}_j(0) = \boldsymbol{x}_{j,0} \Big\}. \tag{17}$$

In this case, (17) depends on the interaction variables $\boldsymbol{z}_i$ and therefore the sensitivity includes not only the inputs but also the states. Thus the calculation of δJ_j is slightly different than in the previous formulations, i.e.

$$\delta J_j(\boldsymbol{x}_i, \boldsymbol{u}_i; \delta \boldsymbol{x}_i, \delta \boldsymbol{u}_i) = \int_0^T \left(\partial_{\boldsymbol{x}_i} H_j[t]\right)^\top \delta \boldsymbol{x}_i(t) + \left(\partial_{\boldsymbol{u}_i} H_j[t]\right)^\top \delta \boldsymbol{u}_i(t)\, \mathrm{d}t, \tag{18}$$

with the Hamiltonian

$$H_j[t] := l_j(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{\lambda}_j^\top \boldsymbol{f}_j(\boldsymbol{x}_j, \boldsymbol{x}_{\text{-}j}^{[k]}, \boldsymbol{u}_j, \boldsymbol{u}_{\text{-}j}^{[k]}) \tag{19}$$

where $\boldsymbol{\lambda}_j \in \mathbb{R}^{n_j}$ is the solution of the adjoint system

$$\begin{aligned} \dot{\boldsymbol{\lambda}}_j = & -\partial_{\boldsymbol{x}_j} l_j(\boldsymbol{x}_j, \boldsymbol{u}_j) - \partial_{\boldsymbol{x}_j} \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{x}_{\text{-}j}^{[k]}, \boldsymbol{u}_j, \boldsymbol{u}_{\text{-}j}^{[k]})^\top \boldsymbol{\lambda}_j \\ & - \sum_{l \in \mathcal{N}^j} \partial_{\boldsymbol{x}_j} \delta J_l(\boldsymbol{z}_j^{[k]}; \boldsymbol{z}_j - \boldsymbol{z}_j^{[k]}), \\ \boldsymbol{\lambda}_j(T) = & \, \partial_{\boldsymbol{x}_j} V_j(\boldsymbol{x}_j(T)) \end{aligned} \tag{20}$$

associated to OCP (16) of agent j. The relation (18) is derived analogously to the previous version (see Appendix A). Note that the interaction variables (15) include the states, which results in an additional term in both the sensitivity (18) and the adjoint system (20) of the respective agent. The state trajectories $\boldsymbol{x}_{\text{-}i}^{[k]}$ of the neighbors are received in step 7 of the algorithm. While the OCPs (10) and (16) share the advantage of local dynamics, (16) additionally has the advantage of being applied to systems with arbitrary couplings, which allows to cover a wide range of problems.

C. Enhancing convergence behavior

In order to enhance the convergence behavior and robustness of Algorithm 1, two additional measures are presented. The first one is a *convexification term* which is a common method for improving convergence (see [23]). To this end, the terms

$$\int_0^T ||\boldsymbol{x}_i - \boldsymbol{x}_i^{[k]}||^2_{\boldsymbol{\Omega}_x} + \int_0^T ||\boldsymbol{u}_i - \boldsymbol{u}_i^{[k]}||^2_{\boldsymbol{\Omega}_u} \tag{21}$$

are added to the cost functional of OCP (4), (10) and (16) respectively, where $\boldsymbol{\Omega}_x \in \mathbb{R}^{n_i \times n_i}$ and $\boldsymbol{\Omega}_u \in \mathbb{R}^{m_i \times m_i}$ are appropriate positive semidefinite weighting matrices.

The second measure is motivated by [24] and seeks to prevent drastic changes in the control variable by an additional damping step after step 7 in Algorithm 1,

$$\boldsymbol{u}_i^{[k+1]} \leftarrow \alpha \boldsymbol{u}_i^{[k+1]} + (1 - \alpha) \boldsymbol{u}_i^{[k]}, \tag{22}$$

where $\boldsymbol{u}_i^{[k]}$ is the solution of the previous iteration and $\alpha \in (0, 1]$ a suitable parameter.

Both measures seek to prevent the algorithm from making drastic changes in a single iteration and are beneficial given the fact that the sensitivity can be interpreted as a first order approximation of the real change in costs.

V. Results

The DMPC algorithm based on the different OCP formulations is evaluated on several numerical examples which are briefly reviewed in the sequel. As indicated in the section titles, Algorithm 1 based on OCP (4), (10) and (16) is referred to as version (i), (ii) and (iii), respectively.

A. Van der Pol oscillator

The Van der Pol (VDP) oscillator is often used in control theory to demonstrate control algorithms, see e.g. [26]. The local dynamics of this system are given by

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} x_{i,2} \\ a(1 - b x_{i,1}^2) x_{i,2} - c x_{i,1} + u_i \end{bmatrix} + \sum_{j \in \mathcal{N}^i} \begin{bmatrix} 0 \\ d x_{j,1} x_{j,2} \end{bmatrix}, \tag{23}$$

with state $\boldsymbol{x}_i = \begin{bmatrix} x_{i,1} & x_{i,2} \end{bmatrix}^\top$ and parameters a, b, c and d. The oscillators are coupled in a circular manner, thus every agent is coupled to two other subsystems. Note that this renders a fully actuated scaleable system and thus the versions (i) & (iii) of the algorithm are applied.

B. Water tank

A second example is a water tank (WT) model with the dynamics

$$\dot{x}_i = a u_i - b + \sum_{j \in \mathcal{N}^i} c \cdot \operatorname{sgn}(x_j - x_i) \cdot \sqrt{|x_j - x_i|} \quad (24)$$

with parameters a, b, c and the water level x_i (see e.g. [14]). Since the right hand side of (24) is not continuous for $x_j - x_i = 0$, a suitable polynomial fit is used for $|x_j - x_i| < 0.025$. The tanks form a cascade and thus every agent has two neighbors (except the first and last tank). Furthermore, the first tank has an input ($a_1 \neq 0$), while the last tank has a drain ($b_N \neq 0$). For all other tanks $a_i = b_i = 0$ holds, which renders the system underactuated. Hence, only version (iii) is applied. Additionally, local state constraints limiting the water level and input constraints are used, which are considered in the local optimization step. The objective is to control the input such that a desired water level is maintained in the last tank.

C. Building automation

Another example is posed by a system used in building control applications. The (normalized) model representing temperature dynamics of a room ventilation system (RVS) is given by

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} a u_i (b - x_{i,1}) + \frac{(x_{i,2} - x_{i,1})}{c} \\ \frac{(x_{i,1} - x_{i,2})}{d} + \sum_{j \in \mathcal{N}^i} \frac{(x_{j,1} - x_{i,2})}{d} \end{bmatrix}, \quad (25)$$

with parameters a, b, c and d, see e.g. [27]. It models the behavior of the room temperature $x_{i,1}$ and wall temperature $x_{i,2}$ of room i. The room temperature is influenced by the inflow of fresh air u_i, while the wall temperature is effected by all rooms connected to it, due to the heat exchange between neighboring rooms. Both DMPC versions (i) & (iii) are applied to this problem.

D. Random network

In order to present an example which features input couplings, the linear random network (RN)

$$\dot{x}_i = a x_i + u_i + \sum_{j \in \mathcal{N}^i} b x_j + u_j \quad (26)$$

with parameters a and b is used, for which the coupling between the agents is determined randomly. The network is created by adding edges between vertices with a probability $p \in (0, 1)$, which corresponds to a Gilbert graph. The process is repeated until a connected graph is obtained. In the following simulations, a probability of $p = 0.1$ is used. All DMPC versions (i), (ii) and (iii) of the algorithm are applied to this system.

E. Simulations

For all examples, quadratic weights for the cost function terms are chosen for both integral and terminal costs, i.e. $V_i(\boldsymbol{x}_i(T)) = \boldsymbol{x}_i^\top(T) \boldsymbol{P}_i \boldsymbol{x}_i(T)$ and $l_i(\boldsymbol{x_i}, \boldsymbol{u}_i) = \boldsymbol{x}_i^\top \boldsymbol{Q}_i \boldsymbol{x}_i + \boldsymbol{u}_i^\top \boldsymbol{R}_i \boldsymbol{u}_i$. Table I lists the respective numerical values used in the simulation of the examined examples. The convergence criterion of Algorithm 1 considers the change in costs between two consecutive iterations (as suggested in Section III-C). The algorithm was implemented in C++ and the local OCPs are solved using the toolbox GRAMPC [25]. In order to evaluate the algorithm, a central MPC was simulated (using GRAMPC) with the same parameters, which is used as a benchmark for the distributed solution. Figure 1 shows the water levels of the tank system. It can be seen that the third tank reaches its desired water level.

Figure 2 shows the convergence behavior and cost trajectories of the examples with 5 (VDP), 10 (RVS), 5 (RN) and 3 (WT) agents. It can be seen, that the algorithm converges after several iterations in the first MPC sampling step, starting from an unbiased initial input trajectory, i.e. $\boldsymbol{u}_i^{[0]}(t) \equiv \boldsymbol{0}$. Furthermore, the cost trajectories show that the distributed algorithm converges to the same solution as the central MPC. The oscillations in the convergence behavior of the water tank stems from the fact that the sensitivities must be propagated from the last to the first tank over the iterations and vice versa. Thus, the frequency of the oscillation depends on the number of tanks.

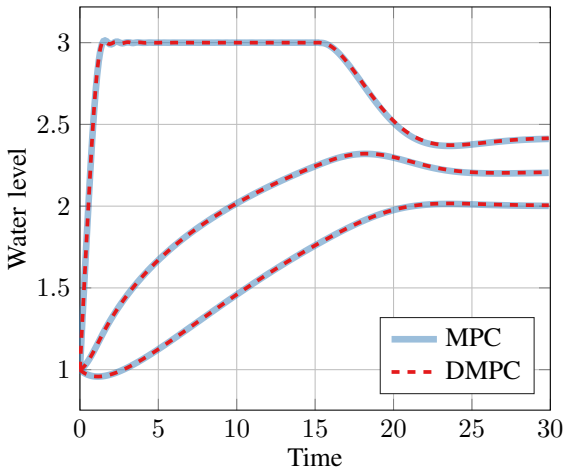

Fig. 1. Trajectories of the water tank level with three agents.

In addition, Fig. 3 shows the computational burden of the overall central solution of the VDP example as well as for a single DMPC agent of version (iii). The number of coupled oscillators (and thus the number of agents) was increased

TABLE I
COST FUNCTIONAL WEIGHTS

	$\boldsymbol{P}_i$	$\boldsymbol{Q}_i$	$\boldsymbol{R}_i$
VDP	$\begin{bmatrix} 33.3 & 3.4 \\ 3.4 & 3.8 \end{bmatrix}$	$\begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}$	0.02
RVS	$\begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}$	0.01
RN	1	1	1
WT	$\begin{cases} 1 & \text{for last tank} \\ 0 & \text{else} \end{cases}$	$\begin{cases} 1 & \text{for last tank} \\ 0 & \text{else} \end{cases}$	7.5

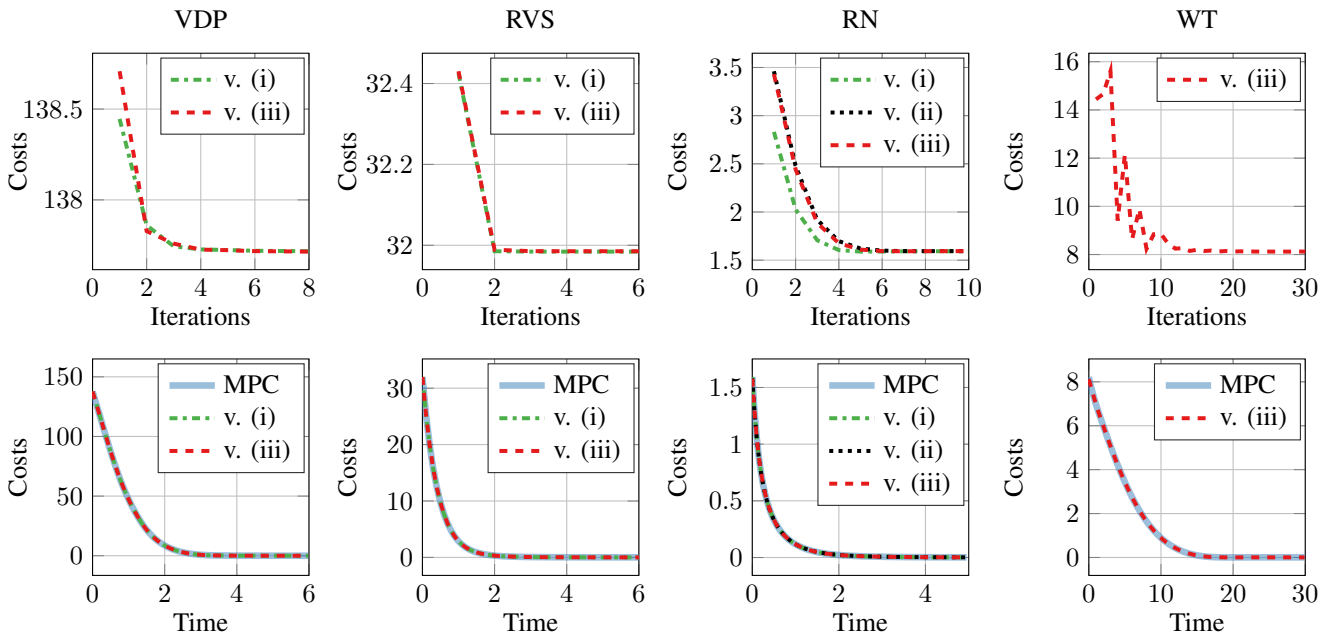

Fig. 2. Convergence behavior in the first MPC sampling step (top row) and cost trajectories (bottom row) of the whole simulation of the different algorithm versions in reference to the central MPC solution.

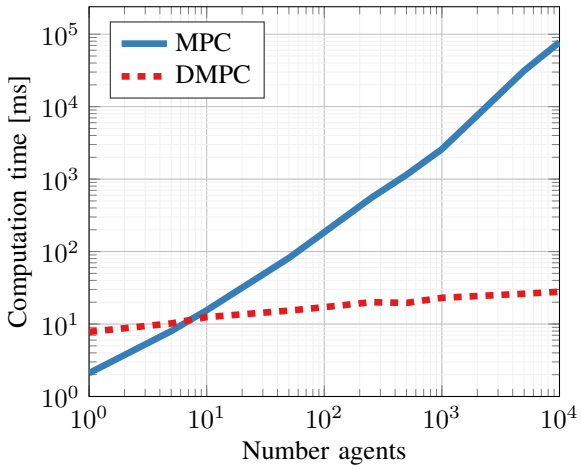

Fig. 3. Computation time for central and distributed MPC (version (iii)) for the VDP problem over number of agents.

from 1 to 10^4 and the average computation time over the first 25 sample steps was recorded. It can be seen, that for the central MPC the required time increases with growing system size, while the computational burden per agent is almost constant for the DMPC, yielding approximately 28 ms for the case of 10^4 agents.

VI. CONCLUSIONS

An algorithm for distributed nonlinear continuous-time MPC was presented, which uses local sensitivities in order to find the optimal solution. Three different formulations of the local problems are presented that differ in the way the systems dynamics are handled. The sensitivities of the neighboring cost w.r.t. the local control can be calculated in a straightforward manner using the Hamiltonian formulation. The individual versions are implemented and evaluated for a set of examples. The algorithm succeeds to converge towards the central optimal solution. Moreover, if only the local dynamics are considered on agent level, the computation time remains approximately constant for an increasing system size. In future research, theoretical statements in terms of convergence behavior are of interest. Moreover, experimental setups in order to investigate the influence of real communication could be examined.

APPENDIX

In the following, the fundamental calculations to derive the sensitivity (8) are given. The sensitivities of all three versions can be derived by applying the same ideas and calculations and therefore a general case is depicted thereafter. Thus instead of using local trajectories $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$, general states $\boldsymbol{x}$ and inputs $\boldsymbol{u}$ are used for the sake of readability. The following calculations are based on results in the field of dynamic optimization, see e.g. [4], [28]. Lets consider the general OCP

$$\min_{\boldsymbol{u}} \; V(\boldsymbol{x}(T)) + \int_0^T l(\boldsymbol{x}, \boldsymbol{u}) \, \mathrm{d}t \tag{27a}$$

$$\text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{27b}$$

$$\boldsymbol{u} \in [\boldsymbol{u}^-, \boldsymbol{u}^+]. \tag{27c}$$

In view of the sensitivity, the main question is, how to evaluate the derivative of (27) w.r.t. $\boldsymbol{u}$. In order to elaborate this, let

$$\boldsymbol{w}^\varepsilon = \boldsymbol{u} + \varepsilon \delta \boldsymbol{u} \tag{28}$$

with $\varepsilon \in \mathbb{R}$ be a variation of the control vector and $\boldsymbol{y}^\varepsilon$ the respective response of the system. Applying a variation with $\varepsilon = 0$ would result in $\boldsymbol{w}^0 = \boldsymbol{u}$ and consequently $\boldsymbol{y}^0 = \boldsymbol{x}$. Furthermore assume $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ to be differentiable. Then the augmented cost functional

$$J(\boldsymbol{w}^\varepsilon) := V(\boldsymbol{y}^\varepsilon(T)) + \int_0^T l(\boldsymbol{y}^\varepsilon, \boldsymbol{w}^\varepsilon) + \boldsymbol{\lambda}^\top [\boldsymbol{f}(\boldsymbol{y}^\varepsilon, \boldsymbol{w}^\varepsilon) - \dot{\boldsymbol{y}}^\varepsilon] \,\mathrm{d}t \quad (29)$$

can be defined, with a so far undetermined $\boldsymbol{\lambda}^\top(t) \in \mathbb{R}^n$. Integration by parts and total differentiation w.r.t. ε yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\varepsilon} J(\boldsymbol{w}^\varepsilon) &= \partial_{\boldsymbol{x}} V^\top(\boldsymbol{y}^\varepsilon(T)) \partial_\varepsilon \boldsymbol{y}^\varepsilon - \left[\boldsymbol{\lambda}^\top \partial_\varepsilon \boldsymbol{y}^\varepsilon\right]_0^T \\ &+ \int_0^T \left[\partial_{\boldsymbol{u}} l(\boldsymbol{y}^\varepsilon, \boldsymbol{w}^\varepsilon) + \partial_{\boldsymbol{u}} \boldsymbol{f}(\boldsymbol{y}^\varepsilon, \boldsymbol{w}^\varepsilon)^\top \boldsymbol{\lambda}\right]^\top \delta \boldsymbol{u} \\ &+ \left[\partial_{\boldsymbol{x}} l(\boldsymbol{y}^\varepsilon, \boldsymbol{w}^\varepsilon) + \partial_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{y}^\varepsilon, \boldsymbol{w}^\varepsilon)^\top \boldsymbol{\lambda} + \dot{\boldsymbol{\lambda}}\right]^\top \partial_\varepsilon \boldsymbol{y}^\varepsilon \,\mathrm{d}t. \quad (30)\end{aligned}$$

Then, by considering the limit $\varepsilon \to 0$ and regarding consistent initial conditions $\partial_\varepsilon \boldsymbol{y}^0(0) = \boldsymbol{0}$, the Gâteaux directional derivative

$$\begin{aligned}\delta J(\boldsymbol{u}; \delta \boldsymbol{u}) &= \left[\partial_{\boldsymbol{x}} V^\top(\boldsymbol{x}(T)) - \boldsymbol{\lambda}^\top(T)\right] \partial_\varepsilon \boldsymbol{y}^0(T) \\ &+ \int_0^T \left[\partial_{\boldsymbol{u}} l(\boldsymbol{x}, \boldsymbol{u}) + \partial_{\boldsymbol{u}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})^\top \boldsymbol{\lambda}\right]^\top \delta \boldsymbol{u} \\ &+ \left[\partial_{\boldsymbol{x}} l(\boldsymbol{x}, \boldsymbol{u}) + \partial_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})^\top \boldsymbol{\lambda} + \dot{\boldsymbol{\lambda}}\right]^\top \partial_\varepsilon \boldsymbol{y}^0 \,\mathrm{d}t \quad (31)\end{aligned}$$

is obtained. Then, considering the adjoint system

$$\dot{\boldsymbol{\lambda}} = -\partial_{\boldsymbol{x}} l(\boldsymbol{x}, \boldsymbol{u}) - \partial_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})^\top \boldsymbol{\lambda}, \quad (32a)$$

$$\boldsymbol{\lambda}(T) = \partial_{\boldsymbol{x}} V(\boldsymbol{x}(T)), \quad (32b)$$

eventually leads to

$$\delta J(\boldsymbol{u}; \delta \boldsymbol{u}) = \int_0^T (\partial_{\boldsymbol{u}} H)^\top \delta \boldsymbol{u} \,\mathrm{d}t, \quad (33)$$

with the Hamiltonian

$$H[t] := l(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}). \quad (34)$$

Note that in the case of sensitivity based DMPC, the adjoint system solved in the local optimization step 6 is equivalent to the one obtained by deriving the sensitivity. Furthermore note that by choosing $\boldsymbol{w}^\varepsilon$ according to the local OCPs (4), (10) and (16), the sensitivities of the various versions can be derived.

REFERENCES

[1] E. F. Camacho, and C. Bordons Alba, Model Predictive Control. 2nd ed. London: Springer, 2004.

[2] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, Model Predictive Control: Theory, Computation, and Design. 2nd ed. Madison: Nob Hill Publishing, 2017.

[3] L. Grüne, and J. Pannek, Nonlinear Model Predictive Control. London: Springer, 2011.

[4] A. E. Bryson, and Y.-C. Ho, Applied Optimal Control. New York: John Wiley & Sons, 1975.

[5] J. Nocedal, and S. J. Wright, Numerical Optimization. New York: Springer, 2006.

[6] R. Scattolini, Architectures for distributed and hierarchical Model Predictive Control – A review, *Journal of Process Control*, vol. 19, no. 5, pp. 723-731, 2009.

[7] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, Distributed model predictive control, *IEEE Control Systems*, vol. 22, no. 1, pp. 44-52, 2002.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the Alternating Direction Method of Multipliers, *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1-122, 2011.

[9] D. P. Palomar, and M. Chiang, A tutorial on decomposition methods for network utility maximization, *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439-1451, 2006.

[10] D. P. Bertsekas, Nonlinear Programming. 2nd ed. Belmont: Athena Scientific, 1999.

[11] D. K. Mohlzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, A survey of distributed optimization and control algorithms for electric power systems *IEEE Trans. on Smart Grid*, vol. 8, no. 6, pp. 2941-2962, 2017.

[12] R. Rostami, G. Costantini, and D. Görges, ADMM-based distributed model predictive control: primal and dual approaches, *in Proc. of IEEE Conference on Decision and Control*, Melbourne, 2017, pp. 6598-6603.

[13] Z. Wang, and C. J. Ong, Distributed model predictive control of linear discrete-time systems with local and global constraints, *Automatica*, vol. 81, pp. 184-195, 2017.

[14] S. Hentzelt, A. Klinger, and K. Graichen, Experimental results for distributed model predictive control applied to a water distribution system, *in Proc. of IEEE International Symposium on Intelligent Control*, Antibes, 2014, pp. 1100-1106.

[15] B. Houska, J. Frasch, and M. Diehl, An augmented lagrangian based algorithm for distributed non convex optimization, *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1101-1127, 2016.

[16] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie, Distributed newton method for large-scale consensus optimization, *IEEE Transactions of Automatic Control*, vol. 64, no. 10, pp. 3983-3994, 2019.

[17] M. Doan, M. Diehl, T. Keviczky, and B. De Schutter, A jacobi decomposition algorithm for distributed convex optimization in distributed model predictive control, *in Proc. 20th IFAC World Congress*, Toulouse, 2017, pp. 4905-4911.

[18] M. D. Mesarovic, D. Macko, and Y. Takahara, Theory of Hierarchical, Multilevel, Systems. Academic Press, New York, 1970.

[19] G. Cohen, On an algorithm of decentralized optimal control, *Journal of Math. Analysis and Applications*, vol. 59, pp. 242-259, 1977.

[20] H. Scheu, and W. Marquardt, Sensitivity-based coordination in distributed model predictive control, *Journal of Process Control*, vol. 21, no. 5, pp. 715-728, 2011.

[21] H. Scheu, J. Busch, and W. Marquardt, Nonlinear distributed dynamic optimization based on first order sensitivities, *in Proc. of the American Control Conference*, Baltimore, 2010, pp. 1574-1579.

[22] D. C. Kravvaritis, and A. N. Yannacopoulos, Variational Methods in Nonlinear Analysis. Berlin: Walter de Gruyter, 2020.

[23] D.P. Bertsekas, Convexification procedures and decomposition methods for nonconvex optimization problems, *Journal of Optimization Theory and Applications*, vol. 29, pp. 169-197, 1979.

[24] M.D. Doan, *Distributed Model Predictive Controller Design based on Distributed Optimization*, Ph.D. dissertation, Center for Systems and Control, Delft Univ. of Tech., Delft, 2012.

[25] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC), *Optimization and Engineering*, vol. 20, pp. 769-809, 2019.

[26] M. A. Barrón, and M. Sen, Synchronization of four coupled van der Pol oscillators, *Nonlinear Dynamics*, vol. 56, pp. 357-367, 2009.

[27] K. B. Jemaa, P. Kotman, and K. Graichen, Model-based potential analysis of demand-controlled ventilation in buildings, *in Proc. of 9th Vienna International Conference on Mathematical Modelling*, Vienna, 2018, pp. 85-90.

[28] G. Allaire, Numerical Analysis and Optimization. Oxford: Oxford University Press, 2007.