A Time Splitting Based Real-Time Iteration Scheme for Nonlinear MPC

Yuning Jiang*, Colin N. Jones, Boris Houska

Abstract— **This paper proposes a parallelizable real-time algorithm for model predictive control (MPC). In contrast to existing distributed and parallel optimization algorithms for linear MPC such as dual decomposition or the alternating direction method of multipliers (ADMM), the proposed algorithm can deal with nonlinear dynamic systems as well as non-convex stage costs. Existing real-time algorithms for MPC simulate and compute sensitivities of the predicted state trajectories on the whole prediction horizon. Different from this, the proposed method uses a reversed real-time scheme, where small-scale nonlinear MPC problems are solved on much shorter horizons and in parallel during the feedback phase, while a large equality constrained coupled QP is solved during the preparation step. This makes the proposed algorithm particularly suited for nonlinear MPC problems with long prediction horizons. The performance and advantages of the proposed method compared to existing real-time nonlinear MPC algorithms are illustrated by applying the method to a benchmark case study.**

I. Introduction

Model predictive control (MPC) is an advanced control technique, widely used in industry [23], [24]. The main idea of MPC is to solve an optimization problem at every sampling time minimizing a given performance criterion on a finite prediction horizon subject to a potentially nonlinear system dynamic as well as state and control constraints. There now exists a large variety of software [17], [10], which implement tailored algorithms [5], [28] for solving small to medium scale optimal control problems in real-time. However, for distributed systems, systems that comprise a large number of differential states, or systems for which long prediction horizons are desired, the computation time of state-of-the-art optimal control algorithms can become a limiting factor.

As many optimal control algorithms are based on direct methods [1], the question how to implement fast distributed MPC algorithms reduces to the question how to solve large-scale structured or distributed optimization problems. For the class of convex optimization problems there exists a large number of distributed optimization algorithms, which are often based on dual decomposition [19]. Here, the dual ascent problem can, for example, be solved by a semi-smooth Newton method [6], [16] or other smoothing techniques based on self-concordant barrier functions [27]. Linear MPC controllers based on dual decomposition methods have been developed and implemented by many authors as reviewed in [25], [7].

Another class of distributed optimization methods for convex optimization problems are based on the alternating direction method of multipliers [3]. In [21] an optimal control algorithm based on ADMM has been developed. Here, the authors apply ADMM in order to distribute the optimal control problem in time, i.e., the corresponding method is applicable in order to solve linear MPC problems with long prediction horizons. Other authors have analyzed applications of ADMM to cooperative control [4], [14], [18]. Moreover, an inexact fast alternating minimization algorithm for distributed MPC has been proposed in [22].

Concerning nonlinear MPC there exists algorithms, which can be used for large-scale applications [5] such as the sequential convex programming based method for which most of the function evaluations, sensitivity computations as well as convex solvers can, in principle, be parallelized. For example, the nonlinear programming software library `GALAHAD` [8] implements a number of augmented Lagrangian based methods, which are suited to large-scale applications. However, distributed nonlinear MPC methods, which solve smaller scale nonlinear optimization problems as part of their iterations, remain scarce. This is mostly due to the fact that non-convex distributed optimization is a very recent field of research for which only a few methods exist [9]. Dual decomposition methods are not applicable to nonconvex optimization problems, because in general there is a duality gap. Similarly, ADMM methods are in general divergent when applied to non-convex optimization problems, as discussed in [11]. Nevertheless, a notable exception is the alternating direction augmented Lagrangian based inexact Newton (ALADIN) method, which has been proposed recently [11]. ALADIN is a distributed nonconvex optimization problem solver. Initial attempts to apply ALADIN in the context of nonlinear optimal control can be found in [15].

The paper is organized as follows. Section II presents the nonlinear MPC problems as well as a review of the main idea for splitting optimal control problems along the time horizon. In Section III, one of the main contributions of this paper is outlined, namely a parallelizable scheme for nonlinear MPC based on a real-time variant of ALADIN. Similar to existing real-time methods [5], the scheme is divided into two phases, a feedback phase and a preparation phase. However, in contrast to [5], the proposed scheme solves a QP in the preparation phase, while small-scale NLPs are solved in parallel during the feedback phase. This has the advantage that the run-time of the feedback phase is independent of

*Corresponding author.

Y. Jiang and B. Houska are with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China {`jiangyn, borish`}`@shanghaitech.edu.cn`

C. N. Jones is with the Automatic Control Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, and is a Visiting Professor at ShanghaiTech University, Shanghai, China {`colin.jones`}`@epfl.ch`

the prediction horizon of the NMPC controller. Section IV provides a local stability proof, which is non-trivial and different from [5], as the QP is solved in the preparation phase. In Section VII, a nonlinear benchmark problem with 4 states and 2 controls is used to illustrate the performance of the proposed real time scheme. For this example, we observe a run-time speed-ups of a factor 16 compared to the traditional real-iteration scheme for NMPC, as implemented in [10]. Section VIII concludes the paper.

Notation. The indicator function is denoted by

$$I(x,y) = \begin{cases} 0 & \text{if } x = y \\ \infty & \text{otherwise.} \end{cases}$$

We use the symbols $\mathbb{S}^n_{++}(\mathbb{S}^n_+)$ to denote the set of symmetric, positive (semi-) definite matrices in $\mathbb{R}^{n\times n}$ and $\|x\|^2_\Sigma = x^\intercal \Sigma x$ to denote Euclidean norms with scaling $\Sigma \in S^n_{++}$. The n-dimensional open unit disk with center 0 and radius $r > 0$ is denoted by $\mathcal{B}^n_r$. A minimizer of an equality constrained optimization problem is called a regular KKT point, if the linear independence constraint qualification (LICQ) and second order sufficient condition (SOSC) are satisfied [20].

II. Nonlinear Model Predictive Control

A. Problem Formulation

This paper concerns nonlinear discrete-time optimal control problems of the form

$$\begin{aligned} V_N(\hat{x}) \;=\; \min_{x,u} \quad & \sum_{k=0}^{N-1} l(x_k, u_k) + M(x_N) \\ \text{s.t.} \quad & \begin{cases} \forall\, k \in \{0, ..., N-1\} \\ x_{k+1} = f(x_k, u_k)\ , \\ x_0 = \hat{x}\ . \end{cases} \end{aligned} \tag{1}$$

Here, x and u denote the state trajectory and the control inputs, respectively, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ the dynamic system equation, $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R} \cup \{\infty\}$ the stage cost, and $M : \mathbb{R}^{n_x} \to \mathbb{R} \cup \{\infty\}$ the terminal cost. The function $V_N : \mathbb{R}^{n_x} \to \mathbb{R} \cup \{\infty\}$ is called the optimal value function. In the context of nonlinear MPC, $\hat{x}$ denotes the current state estimate and N the prediction horizon.

Assumption 1 *The functions f and l are twice Lipschitz-continuously differentiable on $\mathcal{B}^{n_x}_r \times \mathcal{B}^{n_u}_r$ for a given $r > 0$, we have $f(0,0) = 0$ as well as $l(0,0) = 0$, and there exists a constant $c > 0$ such that*

$$\forall x \in \mathcal{B}^{n_x}_r, \qquad l(x,u) \geq c\left(\|x\|_2^2 + \|u\|_2^2\right).$$

Notice that this assumption ensures that the matrices

$$\begin{aligned} &A = \frac{\partial}{\partial x} f(0,0)\,, B = \frac{\partial}{\partial u} f(0,0)\ ,\ Q = \frac{1}{2}\frac{\partial^2}{\partial x^2} l(0,0)\ , \\ &R = \frac{1}{2}\frac{\partial^2}{\partial u^2} l(0,0)\ ,\ S = \frac{1}{2}\frac{\partial^2}{\partial x \partial u} l(0,0) \end{aligned} \tag{2}$$

are well-defined. All these matrices can be pre-computed offline.

Remark 1 *Because this paper is about local stability analysis of a parallel nonlinear MPC scheme, inequality constraints on the control and state can be taken into account by adding indicator functions of the feasible set to the stage cost l and terminal cost M. As long as the point $(0,0)$ is strictly feasible, this is not in conflict with Assumption 1, as differentiability is only needed in a local neighborhood of this point.*

B. Terminal cost

In order to ensure that the limit function

$$V_\infty = \lim_{N\to\infty} V_N$$

is well defined in an open neighborhood of 0, the following standard assumption is introduced (see [12], [24]).

Assumption 2 *The pair (A, B) is (asymptotically) stabilizable.*

Assumptions 1 and 2 also imply that the Riccati equation,

$$\begin{aligned} P =& A^\intercal P A + Q \\ &- \left[APB + S\right]\left(R + B^\intercal P B\right)^{-1}\left[A^\intercal P B + S\right]^\intercal \end{aligned}$$

has a positive definite solution $P \succ 0$ and the terminal cost

$$M(x) = x^\intercal P x$$

satisfies $V_\infty(x) = M(x) + \mathbf{O}\left(\|x\|^3\right)$. In order to simplify notation, the following sections assume that M is indeed constructed in this way, although, in principle, the considerations below could also be generalized for other locally accurate terminal cost functions.

C. Time-splitting of MPC

In order to develop efficient algorithms for solving (1), we assume that the time horizon is split into m pieces with $N = m \cdot n$, $m, n \in \mathbb{N}_{>0}$. Let $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R} \cup \{\infty\}$ denote the cost-to-travel function,

$$\begin{aligned} J(a,b) = \min_{x,u} \quad & \sum_{k=0}^{n-1} l(x_k, u_k) \\ \text{s.t.} \quad & \forall\, k \in \{0, \ldots, n-1\}, \\ & x_{k+1} = f(x_k, u_k)\ , \\ & x_0 = a\ ,\ x_n = b\ , \end{aligned} \tag{3}$$

for all $a, b \in \mathbb{R}^{n_x}$. Notice that the properties of J have been analyzed in [12], where it is also shown that

$$\begin{aligned} V_N(\hat{x}) \;=\; \min_{z} \quad & \sum_{j=1}^{m} J(z_j^a, z_j^b) + M(z_m^b) \\ \text{s.t.} \quad & \forall\, j \in \{1, ..., m-1\}, \\ & z_{j+1}^a \;=\; z_j^b \qquad \mid \lambda_j\ , \\ & z_1^a \;=\; \hat{x}\ . \end{aligned} \tag{4}$$

In the above optimization problem, we have introduced the primal optimization variable $z = [z_1^a, z_1^b, \ldots, z_m^a, z_m^b]$ and $\lambda = [\lambda_1, \ldots, \lambda_{m-1}]$ to denote the Lagrangian multipliers of the consensus constraints.

III. Real-time Parallelizable NMPC

This section proposes a real-time algorithm to solve (4) in a distributed manner. The algorithm is alternating between a preparation phase, in which an LQR problem is solved on the long horizon in order to construct parameterized arrival and terminal cost functions for MPC problems on smaller horizons. The decoupled MPC problems on the smaller horizons are then (approximately) solved in a feedback phase. The construction of the arrival and terminal costs is introduced below, while details on the feedback and preparation phase can be found in Sections III-B and III-C.

A. Parametric Arrival and Terminal Costs

Let

$$\begin{aligned} &\Psi_j : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R} \\ \text{and} \quad &\Phi_j : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R} \end{aligned}$$

denote the arrival and terminal cost,

$$\Psi_j(x, z_j^a, \lambda_{j-1}) \;=\; \frac{1}{2} \left\| x - z_j^a - P^{-1}\lambda_{j-1} \right\|_P^2 ,$$

for $j \in \{2, \dots, m\}$ and

$$\Phi_j(x, z_j^b, \lambda_j) \;=\; \frac{1}{2} \left\| x - z_j^b + P^{-1}\lambda_j \right\|_P^2 ,$$

for $j \in \{0, \dots, m-1\}$. The functions $\Psi_j(\cdot, z_j^a, \lambda_{j-1})$ and $\Phi_j(\cdot, z_j^b, \lambda_j)$ can be interpreted as augmented Lagrangian terms [3] that are parameterized in z_j^a, λ_{j-1} and z_j^b, λ_j. The main idea is to adjust the parameters during a preparation phase that takes the coupling between the different horizon intervals into account. Here, the scaling matrix P is equal to the positive definite solution of the algebraic Riccati equation, as outlined in the previous section. The first arrival and last terminal cost terms

$$\Psi_1(x) \;=\; I(\hat{x}, x) \;,\; \Phi_m(x) \;=\; M(x) = \|x\|_P^2$$

are fixed and not parametric. Now, the objective function of the j-th decoupled problem is given by

$$\begin{aligned} &\mathcal{J}_j(a, b, z_j, \lambda_{j-1}, \lambda_j) \;= \\ &\qquad \Psi_j(a, z_j^a, \lambda_{j-1}) + J(a, b) + \Phi_j(b, z_j^b, \lambda_j) \;, \end{aligned} \tag{5}$$

where λ_{j-1}, λ_j, and $z_j = [z_j^a, z_j^b]$ are parameters. The following section explains how to use these objective functions to define a nonlinear feedback law.

B. Feedback Phase

Algorithm 1 outlines the feedback phase. Notice that this routine is called as soon as the current state measurement $\hat{x}$ is avaiable. In order to compute the control reaction v_0^1, one only needs to (approximately) solve the first parametric minimization problem on the short horizon by applying one or more than one SQP steps. Recall that the arrival and terminal costs of the short horizon problems are parameterized. Therefore, Algorithm 1 accepts a parameter $w = (z, \lambda)$ as an input, which is determined during a preparation phase, as disussed in the section below. A discussion of how to choose the tuning parameter $\gamma > 0$ can be found in Section IV.

Algorithm 1 Real-Time ALADIN: Feedback Phase

Input:

- The current measurement $\hat{x}$.
- Tuning constant $\gamma > 0$ and parameters $w = (z, \lambda)$.

Main Steps:

1) If $\|w\|_2 \geq \gamma \|\hat{x}\|_2$, rescale
$$w \leftarrow \gamma \cdot w \frac{\|\hat{x}\|_2}{\|w\|_2} \;.$$
2) Approximately solve the decoupled NLPs by applying one or more SQP iterations (with pre-computed exact Hessians at the reference trajectory),
$$(y^j, v^j) \;=\; \texttt{SQPMin}\left(\mathcal{J}_j(y_0^j, v_n^j, z_j, \lambda_{j-1}, \lambda_j) \right) \tag{6}$$
for all $j \in \{1, \dots, m\}$ in parallel.
3) Send v_0^1 to the real process.

Algorithm 2 Real-Time ALADIN: Preparation Phase

Input:

- Inexact solutions of the decoupled NLPs,

$$\xi = \left[y_0^1, ..., y_{n-1}^1, y_0^2, ..., y_{n-1}^{m-1}, y^m \right] \;,\; \zeta = \left[v^1, ..., v^m \right] .$$

Main Steps:

1) Compute the gradients and residuals in parallel
$$\begin{aligned} q_k \;&=\; \frac{\partial}{\partial x} l(\xi_k, \zeta_k) \;,\; r_k \;=\; \frac{\partial}{\partial u} l(\xi_k, \zeta_k) \;, \\ c_k \;&=\; f(\xi_k, \zeta_k) - A\xi_k - B\zeta_k \;, \end{aligned}$$
for all $k = 0, ..., N-1$ as well as $q_N = 2P\xi_N$.
2) Solve the coupled quadratic programming problem,
$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N-1} \ell_k(x_k, u_k) + \ell_N(x_N) \\ \text{s.t.} \quad & \forall\, k \in \{0, \dots, N-1\} \\ & x_{k+1} = Ax_k + Bu_k + c_k \;\; \mid \;\; \lambda_k^{\text{QP}} \\ & x_0 = \hat{x} \;. \end{aligned} \tag{7}$$
where for all $k = 0, \dots, N-1$
$$\begin{aligned} \ell_k(x, u) \;&=\; \begin{bmatrix} x - \xi_k \\ u - \zeta_k \end{bmatrix}^\intercal \begin{bmatrix} Q & S \\ S & R \end{bmatrix} \begin{bmatrix} x - \xi_k \\ u - \zeta_k \end{bmatrix} \\ &\quad + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^\intercal \begin{bmatrix} x - \xi_k \\ u - \zeta_k \end{bmatrix} \;, \\ \ell_N(x) \;&= \|x - \xi_N\|_P^2 + q_N^\intercal (x - \xi_N) \;. \end{aligned}$$
3) Set $z_j^b = z_{j+1}^a \leftarrow x_{nj+1}$ and $\lambda_j \leftarrow \lambda_{nj+1}^{\text{QP}}$ for all $j \in \{1, \dots, m-1\}$ as well as $z_m^b = 0$.

C. Preparation Phase

During the preparation phase, as outlined in Algorithm 2, the coupled equality constrained QP (7) is solved. Because this QP is equivalent to an extended linear quadratic regulator (LQR) problem on the full (long) horizon, this QP can be solved efficiently by backward and forward sweeps [24]. Here, all the matrix valued linear algebra operations can be prepared offline as the matrix P has been pre-computed. The solution of this LQR problem is used to update the parameters (z, λ), which are needed to adjust the arrival and terminal costs of the NLPs (6) in the next feedback phase.

Remark 2 *Notice that the proposed real-time scheme is different from the traditional SQP-based real-time iteration scheme for NMPC [5]. In fact, in [5] a QP is solved during the feedback phase while the linearization and linear algebra preparation happens during the preparation phase. In this sense, one could call the above scheme a reversed real-time iteration scheme, because the long-horizon QP is actually solved in the preparation phase. Here, the main advantage compared to [5] is that the CPU time of the feedback phase is independent of the horizon length N, as the short-horizon problems are solved in parallel—and only the first problem needs to be solved to compute the feedback reaction.*

IV. Local Stability and Performance Analysis

In this section, the local stability of the proposed real time scheme in Section III is analyzed. This is non-trivial in the sense that the proposed scheme is different from the standard real-time scheme in [5] (see Remark 2). In the following, we use the notation $u_\infty^0(\hat{x})$ to denote the optimal infinite-horizon feedback law; that is, the first element of the optimal solution of (1) for $N = \infty$. The following proposition is a direct consequence of the definition of V_N in (1). A more formal proof can be found in [24], [26].

Proposition 1 *If Assumptions 1 and 2 be satisfied, then*

$$V_\infty(f(\hat{x}, u_\infty^0(\hat{x}))) \leq V_\infty(\hat{x}) - c\|\hat{x}\|_2^2 \tag{8}$$

for all $\hat{x}$ in an open neighborhood of 0. Moreover, we have

$$|V_\infty(\hat{x}) - V_\infty(\hat{y})| = \mathbf{O}(\|\hat{x}\| + \|\hat{y}\|)\|\hat{x} - \hat{y}\| \tag{9}$$

for all $\hat{x}, \hat{y}$ in an open neighborhood of 0.

In the following, we use the notation $w_m(\hat{x})$ to denote the optimal primal-dual solution of (4). Moreover, we use the notation

$$w_\infty^m(\hat{x}) = \left([[w_\infty(\hat{x})]_1]_{[1:m]}, [[w_\infty(\hat{x})]_2]_{[1:m-1]}\right)$$

to denote the components of the primal $[w_\infty(\hat{x})]_1$ and dual $[w_\infty(\hat{x})]_2$ optimal infinite horizon solution that correspond to the horizon $[1 : m]$. Because the matrix P is found by solving the algebraic Riccati equation, we have [24]

$$w_\infty^m(\hat{x}) = w_m(\hat{x}) + \mathbf{O}(\|\hat{x}\|^2) \, . \tag{10}$$

Lemma 1 *Let Assumptions 1 and 2 be satisfied and let v_0^1 denote the feedback control input—as computed in Algorithm 1. If $w = (z, \lambda)$ denotes the input parameter after the re-scaling step of Algorithm 1, then we have*

$$\|f(\hat{x}, v_0^1(\hat{x})) - f(\hat{x}, u_\infty^0(\hat{x}))\| = \mathbf{O}(\|\hat{x}\|^2) + \mathbf{O}(\|w - w_\infty^m(\hat{x})\|).$$

Proof. Recall that the objective functions $\mathcal{J}_j$ of the decoupled short-horizon MPC problems have been constructed by adding augmented Lagrangian terms in the form of parametric arrival and terminal costs. Thus, if we could set $w = w_\infty^m(\hat{x})$ and if we solve these decoupled problems to optimality, we would find the optimal feedback $v_0^1 = u_\infty(\hat{x})$. However, in general there are two sources of errors: firstly, if we apply only a finite number of SQP steps, a local approximation error of order $\mathbf{O}(\|\hat{x}\|^2)$ must be taken into account, because SQP has a locally quadratic convergence rate [20]. Here, local KKT regularity is ensured by Assumptions 1 and 2. Moreover, the fact that the parametric minimizer of the augmented Lagrangian problems is Lipschitz continuous in w has been proven in [2]. Thus, by combining these existing facts, one obtains the statement of the lemma. ■

Theorem 1 *Let Assumptions 1 and 2 be satisfied. If the tuning constant $\gamma > 0$ is sufficiently large, then the proposed real-time scheme (Algorithms 1 and 2) yields a locally asymptotically stable closed-loop controller.*

Proof. First notice that—although one might intuitively expect that the statement of this theorem holds—a formal stability proof is not entirely straightforward, as the real-time iteration is reversed. Here, the key idea is to apply Lemma 1 twice for two consequtive iterations. Let $\hat{x}$ be the current measurement, $\hat{x}^+$ the next, and $\hat{x}^{++}$ the measurement after the next. Let us first apply Lemma 1 for the current feedback step. Due to our particular construction of the scaling step and since we are solving the full asymptotically accurate QP (up to negligible terms of order 3), see (10), in the preparation phase, we have

$$\|w^+ - w_\infty^m(\hat{x})\| = \mathbf{O}(\|\hat{x}\|^2) \, ,$$

where w^+ denotes the primal dual solution of the QP in Step 2 (before the shifting in Step 3 and before the next measurement $\hat{x}$ becomes available). Thus, since we assume that γ is sufficiently large, such that the scaling step is inactive, we can apply Lemma 1 a second time in order to find

$$\|\hat{x}^{++} - f(\hat{x}^+, u_\infty^0(\hat{x}^+))\| = \mathbf{O}(\|\hat{x}^+\|^2) + \mathbf{O}(\|\hat{x}\|^2) \, . \tag{11}$$

Now, it remains to apply Proposition 1 to find that

$$\begin{aligned} V_\infty(\hat{x}^{++}) &\overset{(9),(11)}{=} V_\infty(f(\hat{x}^+, u_\infty^0(\hat{x}^+))) \\ &\quad + \mathbf{O}(\|\hat{x}^+\|(\|\hat{x}^+\|^2 + \|\hat{x}\|^2)) \\ &\overset{(8)}{\leq} V_\infty(\hat{x}^+) - c\|\hat{x}^+\|_2^2 \\ &\quad + \mathbf{O}(\|\hat{x}^+\|(\|\hat{x}^+\|^2 + \|\hat{x}\|^2)) \, . \end{aligned}$$

The latter inequality is sufficient to establish local asymptotic stability with V_∞ being a local Lyapunov function, since

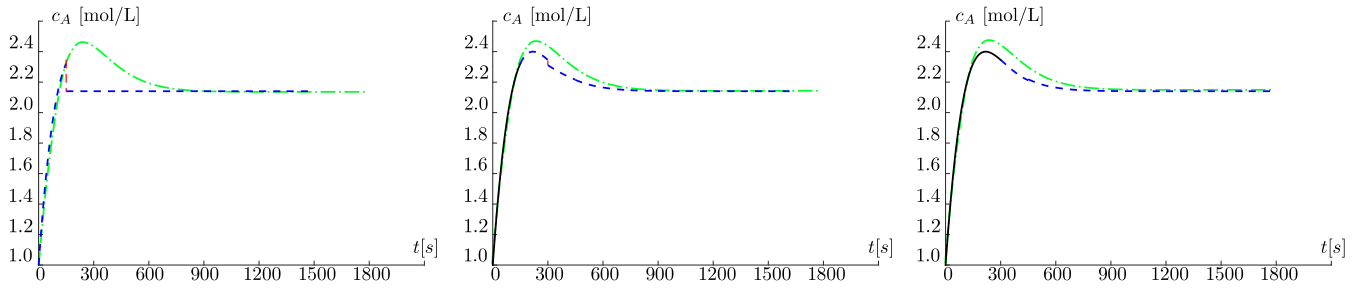

Fig. 1. Real time trajectory of state c_A, black solid line: measured trajectory, blue dashed line: predictive trajectory, red dashed line: gap at coupled node, green dash-dotted line: closed-loop trajectory by ACADO (one SQP iteration per sampling time).

the error terms are all of order 3, while the descent has convergence order 2. Here, the relation $\|\hat{x}\| = \mathbf{O}(\|\hat{x}^+\|)$ holds due to the fact that we are evaluating all functions in the neighborhood of a regular KKT point. ■

Remark 3 *Notice that $\gamma > 0$ should be a sufficiently large constant. However, the statement of Theorem 1 does not hold, if the scaling step in Algorithm 1 is left away. For example, if we have $\hat{x} = 0$, but $w \neq 0$, it is important to implement this scaling—as the decoupled NLP solutions would return $v_0^1 \neq 0$ otherwise and, thus, destabilize the system, although the current state is exactly equal to 0. A discussion of how to choose γ can—in a slighly different context—be found in [13]. Moreover, a more detailed discussion of how the presented ALADIN algorithm performs in the global phase, where active set changes are present, can be found in [11].*

V. Numerical case study

In this section, we apply the proposed parallelizable real time scheme to a benchmark problem, namely a continuously stirred tank reactor (CSTR) [10]. This CSTR model has four states and two controls and can be written in the form

$$\begin{aligned}
\dot{c}_A(t) &= u_1(c_{A0} - c_A(t)) - k_1(\vartheta(t))c_A(t) \\
&\quad -k_3(\vartheta(t))(c_A(t))^2 \\
\dot{c}_B(t) &= -u_1(t)c_B(t) + k_1(\vartheta(t))c_A(t) - k_2(\vartheta(t))c_B(t) \\
\dot{\vartheta}(t) &= u_1(t)(\vartheta_0 - \vartheta(t)) + \tfrac{k_w A_R}{\rho C_p V_R}(\vartheta_K(t) - \vartheta(t)) \\
&\quad -\tfrac{1}{\rho C_p}[k_1(\vartheta(t))c_A(t)H_1 + k_2(\vartheta(t))c_B(t)H_2] \\
&\quad +k_e(\vartheta(t))(c_A(t))^2 H_3] \\
\dot{\vartheta}_K(t) &= \tfrac{1}{m_K C_{PK}}(u_2(t) + k_w A_R(\vartheta(t) - \vartheta_K(t))) \,,
\end{aligned}$$

Here, the first two states, c_A and c_B, are the concentrations of cyclopentadiene (substance A) and cyclopentenol (substance B), respectively, while the other two states, ϑ and ϑ_K, denote the temperature in the reactor and the temperature at the cooling jacket. The control inputs are denoted by u_1 and u_2.

We set up a closed-loop scenario using the above model, where the parameters and control bounds are set to exactly the same values as in [10]. The prediction horizon is set to $T = 1500\,\mathrm{s}$. It is divided into $m = 10$ control intervals of equal length. Moreover, in order to showcase the run-time performance of the proposed scheme, the number of SQP iterations per iteration in the decoupled NLP solvers (Algorithm 1) has been limited to 1.

Fig. 1 shows the result for the state c_A after the one, five, and ten real-time iterations. Clearly, in the first iteration, we can observe a significant consensus gap between the first and second interval. However, already after 5 real-time iterations the gaps are almost closed and the controller starts to operate close to optimality. In order to assess the closed-loop performance of the algorithm, we introduce the relative closed-loop performance degradation, defined as

$$\sum_{k=0}^{\infty} \frac{\ell(x_k, u_k) - V_\infty(\hat{x})}{V_\infty(\hat{x})} \,.$$

This closed-loop performance degradation takes the value 6.7% for the proposed real-time ALADIN iteration and 7.2% for the classical real time algorithm from [5]. Thus, one state that the suboptimality of these two algorithms is almost the same.

In order to illustrate the run-time performance of the proposed parallel real-time scheme, Table I and Table II list the run-times of the traditional real-time iteration scheme, as implemented ACADO Toolkit. The run-times in Table I do not exactly coincide with the times in [10], as we have run both algorithms on the same computer in order to arrive at a fair comparison. Both implementations are using the open-source code generation tools in ACADO Toolkit to export optimized C-code [10]. Notice that the run-time of the proposed parallel real-time scheme for a complete iteration is about a factor 16 faster than the associated run-times of the traditional real-time iteration scheme, as proposed in [5], [10]. In this example, the run-time of the feedback phase is approximately $6\,\mu$s, which must be compared to the time that it takes to solve a QP online—in this case almost $59\,\mu$s. If we implement NMPC controllers with longer prediction horizons, the run-time benefits of the proposed scheme become even more significant, as the proposed feedback step does not depend on N, as long as the interval of the first splitting interval is kept constant.

TABLE I

RUN-TIME OF ACADO CODE GENERATION [10]

	CPU time	Percentage
Integration & sensitivities	117 μs	65 %
QP (Condensing + qpOASES)	59 μs	33 %
Complete real-time iteration	181 μs	100 %

TABLE II

RUN-TIME OF THE PROPOSED REAL-TIME SCHEME WITH 10 THREADS

	CPU time	Percentage
Parallel decoupled MPC	6 μs	55 %
QP sweeps	3 μs	27 %
Communication overhead	2 μs	18 %
Complete real-time iteration	11 μs	100 %

VI. CONCLUSIONS

This paper has introduced a parallelizable real time scheme for nonlinear model predictive control. In contrast to [5], a structured equality-constrained QP is solved in the preparation phase, while the feedback phase solves small-scale MPC problems on short horizons. This has the advantage that the run-time of the feedback phase is independent of the prediction horizon. A main theoretical contribution of this paper has been presented in Theorem 1, where we have established a local closed-loop stability result for the presented scheme. Moreover, we have compared the run-time performance of the proposed controller compared to traditional SQP-based real-time iterations for NMPC by implementing a CSTR benchmark case study, where run-time speed-ups of up to a factor 16 have been observed.

ACKNOWLEDGMENT

YJ and BH were supported by ShanghaiTech University, Grant-Nr. F-0203-14-012. CJ was supported by the EU via FP7-ITN-TEMPO (607 957) and H2020-ITN-AWESCO (642 682). In addition, YJ thanks Moritz Diehl and Andrea Zanelli for inspiring discussions.

REFERENCES

[1] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, volume 6, pages 4555–4559, 1984.

[2] J.F. Bonnans and A. Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[3] 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*, 3(1):1–122, 2011.

[4] C. Conte, T. Summers, M.N. Zeilinger, M. Morari, and C.N. Jones. Computational aspects of distributed optimization in model predictive control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 6819–6824. IEEE, 2012.

[5] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[6] J.V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, 7(3):289–329, 2015.

[7] P. Giselsson, M.D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829–833, 2013.

[8] N.I.M. Gould, D. Orban, and P.L. Toint. Galahad, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Softw.*, 29(4):353–372, December 2003.

[9] A. Hamdi and S. K. Mishra. Decomposition methods based on augmented lagrangians: a survey. In *Topics in nonconvex optimization*, pages 175–203. Springer, 2011.

[10] B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 47:2279–2285, 2011.

[11] B. Houska, J. Frasch, and M. Diehl. An augmented lagrangian based algorithm for distributed nonconvex optimization. *SIAM Journal on Optimization*, 26(2):1101–1127, 2016.

[12] B. Houska and M.A. Müller. Cost-to-travel functions: a new perspective on optimal and model predictive control. *Systems & Control Letters*, 106:79–86, 2017.

[13] Y. Jiang, J. Oravec, B. Houska, and M. Kvasnica. Parallel explicit model predictive control. *arXiv preprint*, 2019.

[14] M. Kögel and R. Findeisen. Cooperative distributed mpc using the alternating direction multiplier method. *IFAC Proceedings Volumes*, 45(15):445–450, 2012.

[15] D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl. A block based aladin scheme for highly parallelizable direct optimal control. In *In Proceedings of the 2016 American Control Conference, Boston, USA*, page 1124–1129, 2016.

[16] A. Kozma, J.V. Frasch, and M. Diehl. A distributed method for convex quadratic programming problems arising in optimal control of distributed systems. In *52nd IEEE Conference on Decision and Control*, pages 1526–1531. IEEE, 2013.

[17] J. Mattingley and S. Boyd. Automatic code generation for real-time convex optimization. *Convex optimization in signal processing and communications*, pages 1–41, 2009.

[18] J.F.C. Mota, J.M.F. Xavier, Pedro P.M.Q. Aguiar, and M. Püschel. Distributed admm for model predictive control and congestion control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 5110–5115. IEEE, 2012.

[19] I. Necoara, C. Savorgnan, D.Q. Tran, J. Suykens, and M. Diehl. Distributed nonlinear optimal control using sequential convex programming and smoothing techniques. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 543–548. IEEE, 2009.

[20] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 2nd edition, 2006.

[21] B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6):2432–2442, 2013.

[22] Y. Pu, M.N. Zeilinger, and C.N. Jones. Inexact fast alternating minimization algorithm for distributed model predictive control. In *53rd IEEE Conference on Decision and Control*, pages 5915–5921. IEEE, 2014.

[23] S.J. Qin and T.A. Badgwell. An overview of nonlinear model predictive control applications. In *Nonlinear model predictive control*, pages 369–392. Springer, 2000.

[24] J.B. Rawlings, D.Q. Mayne, and M.M. Diehl. *Model predictive control: Theory and design, 2nd Edition*. Madison, WI: Nob Hill Publishing, 2017.

[25] S. Richter, M. Morari, and C.N. Jones. Towards computational complexity certification for constrained mpc based on lagrange relaxation and the fast gradient method. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5223–5229. IEEE, 2011.

[26] P.O.M. Scokaert, J.B. Rawlings, and E.S. Meadows. Discrete-time stability with perturbations: Application to model predictive control. *Automatica*, 33(3):463–470, 1997.

[27] D.Q. Tran, I. Necoara, C. Savorgnan, and M. Diehl. An inexact perturbed path-following method for lagrangian decomposition in large-scale separable convex optimization. *SIAM Journal on Optimization*, 23(1):95–125, 2013.

[28] V.M. Zavala and L.T. Biegler. The advanced-step nmpc controller: Optimality, stability and robustness. *Automatica*, 45(1):86–93, 2009.