Block BFGS Based Distributed Optimization for NMPC Using PolyMPC

Yuning Jiang†, Petr Listov†, and Colin N. Jones

Abstract— **This paper presents a block BFGS based distributed optimization approach for nonlinear model predictive control (NMPC). The proposed method is a variant of the augmented Lagrangian based alternating direction inexact Newton method (ALADIN), which achieves a locally superlinear convergence rate. To deal with the NMPC problem in continuous time by employing the proposed method, we elaborate on a systematic implementation based on the C++ library `PolyMPC`. The performance and advantages of the proposed method are illustrated by applying the algorithm to a benchmark continuously stirred tank reactor case study.**

I. Introduction

Model predictive control (MPC) as an advanced control technique has attracted many interests in both industry [26] and academia [28]. The idea of MPC is to solve an optimization problem at each sampling instant, which optimizes a given performance criterion on a finite prediction horizon subject to potentially nonlinear system dynamics as well as state and control constraints. In practice, the optimization problems are scaled up if the dynamic system has distributed structure comprising a number of subsystems [19], [31] or if a long prediction horizon is considered [18], [24]. This leads a significant challenge to the online numerical solver.

When the dynamic system is linear, the online optimal control problems (OCP) are quadratic programs (QP). A number of distributed optimization algorithms based on dual decomposition, have been proposed to deal with OCP QPs. For example, the gradient ascent method is applied to solve the dual problem [11], [27]. Some other alternates employ for example a semi-smooth Newton method [10], [21] or smoothing techniques based on self-concordant barrier functions [9]. Another class of distributed optimization methods for convex optimization problems are based on the alternating direction method of multipliers (ADMM) [3]. In [24] an optimal control algorithm based on ADMM has been developed. Here, the authors applied ADMM in order to decompose the optimal control problem in time. Other authors have investigated applications of ADMM to cooperative control, where a model decomposition is needed [4], [25]. Recently, an open-source ADMM based QP solver, `OSQP` [30], has been released and applied to design linear MPC controllers.

Concerning nonlinear model predictive control (NMPC), sequential convex programming (SCP) [17], [33] based algorithms, have been investigated for solving the resulting nonlinear programs (NLP). For example, the software library `GALAHAD` [12] implements a number of augmented Lagrangian based methods. Another alternate is to use interior point (IP) method. A successful IP based optimization toolkit is `IPOPT`. In order to save time in practice by avoiding the evaluation of exact sensitivities, [17] proposed a quasi-Newton based sequential quadratic programming (SQP) method exploiting the block-diagonal structure of the Hessian matrix. Besides, approximate closed-loop schemes, have been proposed [7], [33] to trade control performance for speed. In these methods, a fixed number of iterations of optimization algorithm are employed at each sampling time such that a suboptimal control input is yielded online. [7] proposed to apply a single SQP step at each sampling time. Local stability properties of such a scheme was further investigated in [8] while the corresponding open-source toolkit `ACADO` [15] provides a framework for direct optimal control in real-time iteration. Recently, a C++ library `PolyMPC` is proposed in [22] for pseudospectral based real-time NMPC. A more general framework of real-time predictive control has been investigated in [33], [34].

In the context of NMPC, although most of the function evaluation and sensitivity computation can be parallelized in both SCP and IP methods, distributed NMPC methods, which solve smaller scale nonlinear programming (NLP) problems as part of their iteration, remain scarce. This is mostly due to the fact that only a very few methods exists [14] for nonconvex distributed optimization. Dual decomposition methods are not applicable to nonconvex optimization problem, because we may have a duality gap. Similarly, ADMM methods are in general divergent when applied to nonconvex optimization problems, as discussed in [16]. Recently, a notable exception, the alternating direction augmented Lagrangian based inexact Newton (ALADIN) method, has been proposed [16]. ALADIN is a distributed nonconvex optimization problem solver, which is capable of achieving local quadratic convergence rates under mild assumptions if the exact Hessian is evaluated. Initial attempts to apply ALADIN in the context of nonlinear optimal control can be found in [20]. Furthermore, [18] proposed a parallelizable real-time iteration for NMPC without state and control constraints, which runs a single ALADIN iteration at each sampling time.

Section II introduces a generic form of distributed optimization problem that distinguishes the locally decoupled and coupled variables. The main contribution of this paper can be found in Section III and IV. Section III proposes a variant of ALADIN based on block inverse BFGS updates. The computational complexity and convergence properties

†The first two authors contributed equally.

This work is support by the Swiss National Science Foundation under the RISK project (Risk Aware Data-Driven Demand Response), grant number 200021–175627. All authors are with Laboratoire d'Automatique, EPFL, Switzerland. {yuning.jiang, petr.listov, colin.jones@epfl.ch}

of the method is discussed. We prove that the complexity of BFGS-ALADIN is smaller than the complexity of BFGS-SQP. Furthermore, compared with BFGS-SQP, which can only make primal iterates have local superlienar convergence rate, we show that the primal-dual iterates of BFGS-ALADIN converge superlinearly. Section IV considers the NMPC problem in continuous time and introduces a time splitting based distributed reformulation. Then, we discuss the implementation details of BFGS-ALADIN for NMPC based on `PolyMPC`. Section V presents numerical results that have been obtained for a benchmark continuously stirred tank reactor (CSTR) reactor control problem and which show that the proposed scheme outperforms BFGS-SQP methods while being able to stabilize in real time iteration. Section VI concludes the paper.

Notation: We use notation $\mathbb{S}^n_{++}(\mathbb{S}^n_{+})$ to denote real symmetric and positive (semi-)definite matrices in $\mathbb{R}^{n\times n}$. For a given matrix $\Sigma \in \mathbb{S}^n_{++}$, we use notation $\|x\|^2_\Sigma = x^\top \Sigma x$. A local minimizer is called a regular KKT point if the *linear independence constraint qualification* (LICQ), *strict complementarity* (SC) and the *second order sufficient conditions* (SOSC) are satisfied [23]

II. Problem Formulation

This paper concerns distributed optimization problems consisting of M subproblems with decision variables $x_i \in \mathbb{R}^{n_{x_i}}$ and $y_i \in \mathbb{R}^{n_{y_i}}$,

$$\min_{x,y} \quad \sum_{i=1}^{M} f_i(x_i, y_i) \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} A_i x_i = 0 \quad | \ \lambda \tag{1b}$$

$$h_i(x_i, y_i) \le 0 \quad | \ \kappa_i \quad i = 1, ..., M \ , \tag{1c}$$

with decoupled objective $f_i : \mathbb{R}^{n_{x_i}} \times \mathbb{R}^{n_{y_i}} \to \mathbb{R}$ and decoupled inequality constraint $h_i : \mathbb{R}^{n_{x_i}} \times \mathbb{R}^{n_{y_i}} \to \mathbb{R}^{n_{h_i}}$. Both are potentially non-convex. Throughout the paper, we write down the Lagrangian multipliers immediately after the constraints such that $\lambda \in \mathbb{R}^m$ and $\kappa_i \in \mathbb{R}^{n_{h_i}}$ denote the Lagrange multipliers of the coupled affine equality (1b) and decoupled inequality constraints (1c), respectively. (1) distinguishes between y_i that are entirely local and x_i that are coupled via an affine constraint (1b) given by matrices $A_i \in \mathbb{R}^{m\times n_{x_i}}$. Note that in practice, the number of coupled variables is often much smaller than the number of fully decoupled variables, i.e. $n_x \ll n_y$. In order to exploit such structure, we define the parametric optimization problem

$$F_i(x_i) \ := \ \min_{y_i} \ \ f_i(x_i, y_i) \quad \text{s.t.} \ \ h_i(x_i, y_i) \le 0 \ , \tag{2}$$

over x_i while additionally enforcing $F_i(x_i) = \infty$ for all x_i for which (2) is infeasible. As a result, $F_i : \mathbb{R}^{n_{x_i}} \to \mathbb{R} \cup \{\infty\}$ is an extended-value function. In this paper, we assume:

- **A1** The functions f_i and h_i are three times continuously differentiable in both arguments.
- **A2** Matrix $A = [A_1 \ ... \ A_M]$ has full row rank.

Note that F_i are, however, non-differentiable, even if **A1** is satisfied. This happens because there may be changes of active set. **A2** is equivalent to the standard LICQ condition for coupled equality constraints. Accordingly, Problem (1) can be rewritten in the form of

$$\min_{x} \ \ \sum_{i=1}^{M} F_i(x_i) \quad \text{s.t.} \ \ \sum_{i=1}^{M} A_i x_i = 0 \ . \tag{3}$$

Here, the main motivation for considering problems of the form (3) is that the original (1) might be a large-scale optimization problem that is difficult to solve directly. To this end, an efficient distributed algorithm is developed in the next section to solve (3), where the functions F_i can be evaluated in parallel.

III. Block BFGS Based ALADIN

This section proposes a block BFGS based ALADIN variant to solve (3). Similar to existing ALADIN methods [16], [20], the proposed variant also alternates between solving decoupled small-scale NLPs in parallel and dealing with a equality constrained QP for consensus. However, our main advantage is that the consensus QP is constructed only w.r.t. the coupled local variable x_i and does not need consider the active constraints at local solutions. The two main steps of the proposed approach is summarized as follows:

1) *Parallelizable Step:* The small-scale NLPs

$$\xi_i^{k+1} = \arg\min_{\xi_i} F_i(\xi_i) + (A_i\xi_i)^\top \lambda^k + \frac{1}{2}\left\|\xi_i - x_i^k\right\|^2_{\Sigma_i} \tag{4}$$

with potentially adjustable scaling matrices $\Sigma_i \in \mathbb{S}^{n_{x_i}}_{++}$ are solved in parallel for all $i = 1, ..., M$. Here, we use k to denote the iteration counter. The first-order optimality condition of (4) yields

$$b_i^{k+1} = \Sigma_i(x_i^k - \xi_i^{k+1}) - A_i^\top \lambda_i^k \in \partial F_i(\xi_i^{k+1})$$

with $\partial F_i(\xi_i^{k+1})$ the sub-differential [29] of F_i at the point ξ_i^{k+1}. Note that only if F_i are differentiable at ξ_i^{k+1}, b_i^{k+1} are equal to the gradient $\nabla F_i(\xi_i^{k+1})$. However, F_i are not always differentiable. In such case, we only know that $b_i^{k+1} \in \partial F_i(\xi_i^{k+1})$. Then, the BFGS Hessian approximation is computed as follows,

$$H_i^{k+1} = H_i^k - \frac{H_i^k s_i^k (H_i^k s_i^k)^\top}{(s_i^k)^\top H_i^k s_i^k} + \frac{d_i^k (d_i^k)^\top}{(d_i^k)^\top s_i^k} \tag{5}$$

with $s_i^k = \xi_i^{k+1} - \xi_i^k$ and $d_i^k = b_i^{k+1} - b_i^k$.

2) *Consensus Step:* The equality-constrained QP

$$x^{k+1} = \arg\min_{x} \ \ \sum_{i=1}^{M} \frac{1}{2}\left\|x_i - \xi_i^{k+1}\right\|^2_{H_i^{k+1}} + x_i^\top b_i^{k+1}$$

$$\text{s.t.} \ \ \sum_{i=1}^{M} A_i x_i = 0 \quad | \ \ \lambda^{k+1} \ , \tag{6}$$

based on the local information is solved. In the following, we use compact notations

$$H = \text{diag}(H_1, ..., H_M) \ , \ b = \begin{pmatrix} b_1^\top & ... & b_M^\top \end{pmatrix}^\top$$

such that the linear KKT system of (6) can be written as

$$\begin{bmatrix} H^{k+1} & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} x^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} H^{k+1}\xi^{k+1} - b^{k+1} \\ 0 \end{bmatrix}. \quad (7)$$

Accordingly, the solution of (7) can be worked out analytically as

$$\lambda^{k+1} = \left[\sum_{i=1}^{M} A_i (H_i^k)^{-1} A_i^\top\right]^{-1} \cdot \left[\sum_{i=1}^{M} A_i \left(\xi_i^{k+1} - (H_i^{k+1})^{-1} b_i^{k+1}\right)\right], \quad (8a)$$

$$x_i^{k+1} = \xi_i^{k+1} - \left[H_i^{k+1}\right]^{-1} (A_i^\top \lambda^{k+1} + b_i^{k+1}). \quad (8b)$$

As BFGS update (5) is a rank-2 update while the solution map (8) only requires the inverse of H_i^k, the Sherman-Morrison-Woodbury formula could be applied to directly evaluate the inverse $B_i^k = (H_i^k)^{-1}$, $i = 1, ..., M$ with a rank-2 update as well [23, Chapter 6]. The computation detail is discussed in the next section.

A. Algorithm Structure

Algorithm 1 elaborates the proposed distributed method for solving (3) in detail. In Step 1), as functions F_i have the form (2), one should not introduce a bilevel structure for solving the decoupled NLPs (4), but solve the joint augmented Lagrangian minimization problem (10) instead. As a result, b_i^{k+1} evaluated in Step 3) is equal to the partial derivative

$$b_i^{k+1} = \frac{\partial}{\partial x_i}\left\{ f_i(\xi_i^{k+1}, y_i^{k+1}) + (\kappa_i^{k+1})^\top h_i(x_i^{k+1}, y_i^{k+1})\right\}$$

where κ_i^{k+1} denotes the dual solution of decoupled problem (10). Step 2) implements a termination criterion. If the condition holds, we can have the current iteration satisfy the stationary condition and primal condition of (3) up to an error of order $\mathbf{O}(\epsilon)$, i.e.,

$$\mathbf{O}(\epsilon) = \sum_{i=1}^{M} A_i \xi_i^{k+1} \quad \text{and} \quad \mathbf{O}(\epsilon) \in \partial F_i(\xi_i^{k+1}) + A_i^\top \lambda^k$$

for all $i = 1, ..., M$. The Sherman-Morrison-Woodbury formula is applied to evaluate the sensitivities (11) in Step 3) and to update the dual Hessian in Step 4). Note that the update of D^k requires the forward sweep (12) to apply the Sherman-Morrison-Woodbury formula recursively for each block $A_i B_i A_i^\top$ in (9). Step 5) updates the primal and dual iterates following (8). The complexity of the block BFGS update in Algorithm 1 is $\mathbf{O}\left(\sum_{i=1}^{M} n_{x_i}^2\right)$. Compared to this, the BFGS based SQP methods for solving (1) requires at least $\mathbf{O}\left(\sum_{i=1}^{M}(n_{x_i} + n_{y_i})^3\right)$. Because a direct BFGS update has to be used with linearizing the inequality constraint in sub-QPs such that the standard active set solvers cannot exploit rank-2 updates.

The standard ALADIN method [16] requires one to consider the linearization of active inequality constraints at the

Algorithm 1 Block BFGS-based ALADIN

Input:

- Initial guesses $x_i^0 \in \mathbb{R}^{n_{x_i}}$ and $\lambda^0 \in \mathbb{R}^m$.
- Choose a terminal tolerance $\epsilon > 0$, scaling parameters $\rho > 0$ and scaling matrices $\Sigma_i \in \mathbb{S}_{++}^{n_{x_i}}$.

Initialization:

- Set $k = 0$ and choose initial inverse Hessian approximation $B_i^0 \in \mathbb{S}_{++}^{n_{x_i}}$ and set

$$D^0 = \left[\sum_{i=1}^{M} A_i B_i^0 A_i^\top\right]^{-1}. \quad (9)$$

Repeat:

1) Solve decoupled NLPs

$$(\xi_i^{k+1}, y_i^{k+1}) := \arg\min_{\xi_i, y_i} \; f_i(\xi_i, y_i) + (A_i^\top \lambda^k)^\top \xi_i + \frac{1}{2}\left\|\xi_i - x_i^k\right\|_{\Sigma_i}^2 \quad \text{s.t.} \quad h_i(\xi_i, y_i) \le 0 \quad | \; \kappa_i^{k+1} \quad (10)$$

for all $i = 1, \ldots, M$ in parallel.

2) If $\left\|\sum_{i=1}^{N} A_i \xi_i^{k+1}\right\| \le \epsilon$ and $\max_i \|\xi_i^{k+1} - x_i^k\| \le \epsilon$, terminate with

$$(x^*, y^*) = (\xi_1^{k+1}, ..., \xi_M^{k+1}, y_1^{k+1}, \ldots, y_M^{k+1}).$$

3) Evaluate sensitivities

$$b_i^{k+1} = \Sigma_i(x_i^k - \xi_i^{k+1}) - A_i^\top \lambda^k, \quad (11a)$$

$$B_i^{k+1} = (I - \alpha_i^k s_i^k (d_i^k)^\top)^\top B_i^k (I - \alpha_i s_i^k (d_i^k)^\top) + \alpha_i^k d_i^k (d_i^k)^\top \quad (11b)$$

$$S_i^{k+1} = \begin{bmatrix} \left(\sqrt{\beta_i^k} A_i d_i^k + \frac{\alpha_i^k}{\sqrt{\beta_i^k}} A_i B_i^k s_i^k\right)^\top \\ \left(-\frac{\alpha_i^k}{\sqrt{\beta_i^k}} A_i B_i^k s_i^k\right)^\top \end{bmatrix}^\top \quad (11c)$$

for all $i = 1, ..., M$ in parallel, where

$$s_i^k = \xi_i^{k+1} - \xi_i^k, \; d_i^k = b_i^{k+1} - b_i^k,$$

$$\alpha_i^k = \frac{1}{(d_i^k)^\top s_i^k}, \; \beta_i^k = (\alpha_i^k)^2 \left((s_i^k)^\top B_i^k s_i^k + 1\right).$$

4) Set $D^{k+1} = D^k$ and compute
For $1 : M$ **do**

$$D^{k+1} \leftarrow D^{k+1} - D^{k+1} S_i^{k+1} \cdot \left(I + (S_i^{k+1})^\top D^{k+1} S_i^{k+1}\right)^{-1} (S_i^{k+1})^\top D^{k+1} \quad (12)$$

End

5) Update the primal and dual iterates by computing

$$\lambda^{k+1} = D^{k+1}\left[\sum_{i=1}^{M} A_i \left(\xi_i^{k+1} - B_i^{k+1} b_i^{k+1}\right)\right], \quad (13a)$$

$$x_i^{k+1} = \xi_i^{k+1} - B_i^{k+1}(A_i^\top \lambda^{k+1} + b_i^{k+1}). \quad (13b)$$

6) Set $k \leftarrow k + 1$ and go to Step 1).

local solution of (10). Therefore, the changes of local active set causes that the scale of the coupled QP in the standard ALADIN method is changed in iterations. Moreover, all variables x_i and y_i are included in the coupled QP. In order to solve the associated large scale KKT system, as discussed in [20], every iteration requires the null space method to eliminate the linearization of the active constraints. In contrast to this, QP (6) is only related to variables x_i such that its scale is small and fixed. Each loop of Algorithm 1 only needs to compute the dual update (13a) with updating D, and then broadcasts λ^{k+1} such that (13b) can be evaluated in parallel.

B. Local Convergence Analysis

If Algorithm 1 is initialized far from the local minimizers, an advanced globalization routine has to be used in order to enforce convergence, more details refer to [16, Sec. 6]. The following discussion is about the local convergence properties of Algorithm 1.

A3 Function $F(x) = \sum_{i=1}^{M} F_i(x_i)$ satisfies $F(x^* + Ps) < \infty$ for all s in a neighborhood of x^* a local minimizer of Problem (3). Here, matrix P spans the null space of constraint Jacobian A, i.e., $AP = 0$.

In order to establish the local convergence, we introduce the following technical result.

Lemma 1 *Let **A1–3** be satisfied. If the local minimizer (x^*, λ^*) is a regular KKT point, there exist neighborhoods $\mathbb{X}_i = \{x_i \mid \|x_i - x_i^*\| \leq \epsilon\}$ with $\epsilon > 0$ in which F_i is twice continuously differentiable for all $i = 1, ..., M$.*

A3 ensures that gradients ∇F_i exist in a neighborhood of the solution (x^*, λ^*). If (x^*, λ^*) is further a regular KKT point of (3), there exists sets $\mathbb{X}_i$ in which the active set of (2) is fixed at any $x_i \in \mathbb{X}_i$. If **A1** also holds, the proof of Lemma 1 follows by applying a generalized version of the implicit function theorem for parametric programming [2].

A4 The BFGS update H_i^k and its inverse B_i^k are bounded for all $i = 1, ..., M$.

Here, **A4** further assumes the BFGS updates are bounded such that the following Lemma can be introduced.

Lemma 2 *Let **A1–4** be satisfied and the initialization satisfies $x_i^0 \in \mathbb{X}_i$ while $\|H_i^0 - \nabla^2 F_i(x_i^*)\|$ is sufficiently small. If Algorithm 1 converges to a regular KKT point and the curvature condition $(s_i^k)^\top d_i^k \geq 0$ locally holds for all $k > 0$, the iterates of Algorithm 1 satisfy the following inequalities*

$$\|x^k - x^*\| \leq \mathbf{o}(\|\xi^k - x^*\|),\ \|\lambda^k - \lambda^*\| \leq \mathbf{o}(\|\xi^k - x^*\|). \quad (14)$$

We provide a proof of Lemma 2 in the Appendix. Then, the local super-linear convergence of Algorithm 1 is given by the following theorem.

Theorem 1 *Let the assumptions in Lemma 2 be satisfied. Furthermore, The scaling matrices Σ_i in (10) locally are assumed to satisfy $\nabla^2 F_i(x_i^*) + \Sigma_i \succ 0$ for all $i = 1, ..., M$. If the local minimizer is a regular KKT point, then (x^k, λ^k) converges to (x^*, λ^*) superlinearly.*

Proof. In Algorithm 1, the solution of decoupled problems (10) is a parametric map over the current primal dual iterates (x^k, λ^k). According to [16, Lem. 3], the solution map is Lipschitz continuous under the assumptions, i.e., there exist constants $\chi_1, \chi_2 > 0$ such that

$$\|\xi^{k+1} - x^*\| \leq \chi_1 \|x^k - x^*\| + \chi_2 \|\lambda^k - \lambda^*\| . \quad (15)$$

Based on Lemma 2, substituting (14) into (15) yields

$$\|x^{k+1} - x^*\| + \|\lambda^{k+1} - \lambda^*\| \leq \mathbf{o}(\chi_1 \|x^k - x^*\| + \chi_2 \|\lambda^k - \lambda^*\|),$$

which indicates the superlinear convergence rate. □

Remark 1 *In practice, the curvature condition $d_i^\top s_i > 0$ in Lemma 2, which may be violated even when the iterates are close to the minimizer [23]. A damped variant of inverse BFGS for non-convex optimization can be used to overcome this difficulty [5]. In contrast to standard inverse BFGS, damped BFGS replaces d_i by*

$$v_i = \theta_i d_i + (1 - \theta_i) B_i s_i ,$$

where the scalar θ_i is computed as

$$\theta_i = \begin{cases} 1 & \text{if } d_i^\top s_i \geq 0.2 s_i^\top B_i s_i , \\ \frac{0.8 s_i^\top B_i s_i}{s_i^\top B_i s_i - d_i^\top s_i} & \text{if } d_i^\top s_i < 0.2 s_i^\top B_i s_i . \end{cases} \quad (16)$$

Then, we only have R-superlinear rate [23, Sec. 18.7].

IV. Distributed Nonlinear Model Predictive Control Using PolyMPC

This section presents using Algorithm 1 as an online solver in a nonlinear model predictive control (NMPC) scheme while elaborating on its implementation details based on the `PolyMPC` library.

A. Time Splitting for NMPC

The NMPC problem in continuous time is given by

$$\begin{aligned} \min_{s(\cdot), u(\cdot)} \quad & \int_0^T \ell(s(t), u(t)) dt + \Phi(x(T)) \\ \text{s.t.} \quad & \forall t \in [0, T] , \ \ \dot{s}(t) = f(s(t), u(t)) , \\ & \forall t \in [0, T] , \ \ c(s(t), u(t)) \leq 0 , \\ & s(0) = \hat{s} , \ \Psi(s(T)) \leq 0 \end{aligned} \quad (17)$$

with state $s : [0, T] \to \mathbb{R}^{n_s}$, control inputs $u : [0, T] \to \mathbb{R}^{n_u}$, stage cost $\ell : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geq 0}$, and terminal cost $\Phi : \mathbb{R}^{n_s} \to \mathbb{R}$. Here, function $f : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ denotes the process governing differential equation and $c : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ denotes state and control inequality path constraints while $\Psi : \mathbb{R}^{n_s} \to \mathbb{R}^{n_f}$ the terminal constraints. The initial state is given by $\hat{x} \in \mathbb{R}^{n_x}$. In order to handle (17) in a distributed manner, we split the time horizon $[0, T]$ into M intervals $[t_i, t_{i+1}]$ for all $i = 0, ..., M - 1$ with $t_0 = 0$

and $t_M = T$. Next, we write down short horizon NMPC problems

$$\begin{aligned} V_i(a,b) = \min_{s(\cdot),u(\cdot)} & \int_{t_i}^{t_{i+1}} \ell(s(t),u(t))dt \\ \text{s.t. } & \forall t \in [t_i, t_{i+1}] \ , \ \dot{s}(t) = f(s(t),u(t)) \ , \\ & \forall t \in [t_i, t_{i+1}] \ , \ c(s(t),u(t)) \leq 0 \ , \\ & s(t_i) = a \ , \ s(t_{i+1}) = b \ , \end{aligned} \tag{18}$$

with parametric initial state a and terminal state b. Moreover, we enforce $V_i(a,b) = \infty$ if (18) is infeasible. As a result, (17) can be rewritten into the dense form of (3),

$$\begin{aligned} \min_{x} \ & \sum_{i=0}^{M-1} V_i(x_i^a, x_i^b) + \Phi(x_{M-1}^b) \\ \text{s.t. } & \begin{cases} x_i^b = x_{i+1}^a \ , \ i = 0,1,...,M-2 \\ x_0^a = \hat{s} \end{cases} \end{aligned} \tag{19}$$

such that Algorithm 1 can be used as the online solver to deal with (17). Note that solving the decoupled problem requires discretization first and direct method [1] is commonly used.

B. Implementation using PolyMPC

The diagram shown in Figure 1 elaborates the implementation details of applying Algorithm 1 to NMPC in `PolyMPC`. There are three main modules in our implementation: initialization, local solver, consensus update.

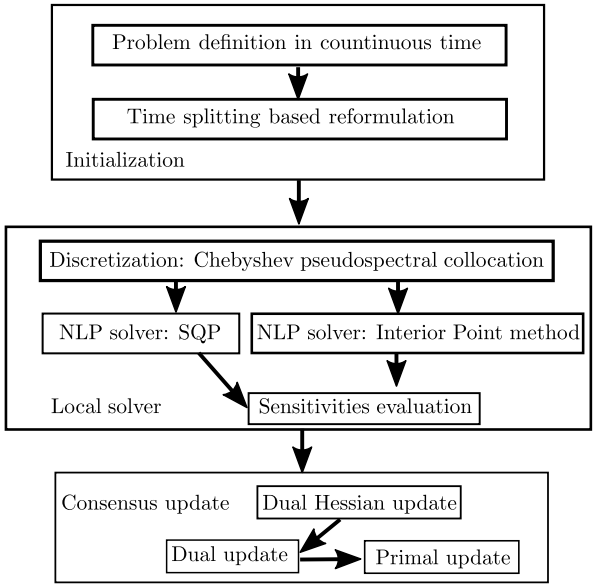

Fig. 1. Implementation of Algorithm 1 in `PolyMPC`

First, users could denote the OCP (17) via the interface of `PolyMPC` and define the setting of time splitting method. Second, the local solver module comprises the optimized implementation of spectral and pseudo-spectral methods for discretizing the continuous time OCP. The collocation code supports several types of Chebyshev and Legendre nodes of arbitrary order while the spectral module implements the Galerkin projection on Chebyshev and Legendre orthogonal bases. Once the discretization is finished, an implementation of a generic SQP solver with ADMM-based sub-QP solver and interfaces to several other open-source QP solvers can be used to deal with (10). Another option is to use interior point method based solver `IPOPT` [32], which is accessed through the `Casadi` interface of `PolyMPC`. The consensus update module uses `Eigen` [13] as linear algebraic routine to deal with the matrix and vector multiplication. This implementation leverages the expression templates mechanism of `Eigen` to avoid unnecessary copying in complex mathematical expressions and static polymorphism in C++ to cut runtime overhead linked to the inheritance. Moreover, a special care is taken for the memory management and exploitation of sparsity associated with pseudo-spectral collocation methods.

V. Numerical Case Study

This section presents the numerical results of applying Algorithm 1 to a benchmark–continuously stirred tank reactor (CSTR) [6], [15]. The associated dynamic differential equation is given by

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

Here, states c_A and c_B denote the concentrations of cyclopentadiene (substance A) and cyclopentenol (substance B), respectively, while states ϑ and ϑ_k are the temperature in the reactor and temperature in the cooling jacket of the tank reactor. The parameter of system model and setup of NMPC problem can be found in [6, Section 1.2]. The horizon length for the experiments is chosen as $T = 1500$ seconds with $N = 50$ control intervals. For the numerical study we employ the Chebyshev-Gauss-Lobatto (CGL) pseudo-spectral collocation scheme to approximate the subproblems (18) as described in the last section. The number of segments is chosen to be 2 and the order of polynomials is 5. Simulations were performed on a 3.1 GHz Intel i7 processor (i7-7920HQ) with 4 cores, 8 threads and 16 GB of RAM, running `Casadi-3.4.5` and `Eigen-3.3.7`. All local NLPs (10) in Step 1 of Algorithm 1 are solved to a high accuracy by using `PolyMPC` interfacing `IPOPT` solver.

In order to illustrate the local convergence of Algorithm 1. Figure 2 shows the performance of Algorithm 1 with respect to different block size n. Here, $w^k = (x^k, \lambda^k)$ denotes the primal and dual iterates. The results illustrate that the local convergence is improved with enlarging the block. As discussed in Section II, (1) in the form of reformulation (3)

might cause F_i be nondifferentiable in certain iterations. In such cases, only a linear convergence rate can be observed.

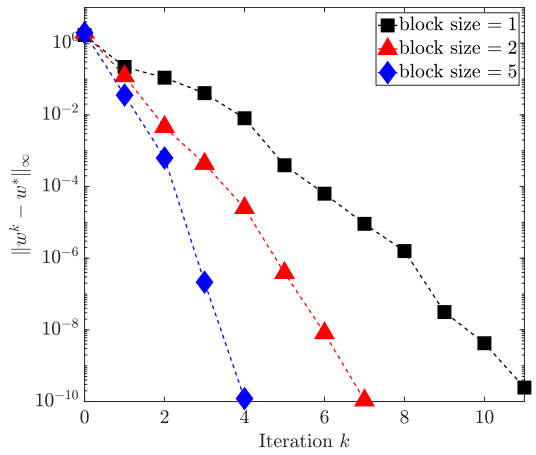

Fig. 2. Open loop convergence with different block sizes n

Table I provides the detailed CPU time for one iteration of Algorithm 1 with different block sizes. Here, it is assumed that there is no communication delay, and the parallelizable operations are divided equally between the computational units. It shows that the smaller n yields a more parallelizable implementation and, thus faster run-time of one iteration. In particular, increasing the block size n increases the run-time in the first row. The operations in the second and the last rows are parallelizable and independent of the block size n. Besides, the operations in the third and fourth rows are centralized. Since a smaller n makes the number of blocks M be larger, these operations require more time with decreasing n.

TABLE I

DETAILED CPU TIME IN [MS] FOR ONE ITERATION OF ALGORITHM 1 WITH DIFFERENT BLOCK SIZES n

	Operation	$n=1$	$n=2$	$n=5$
P[1]	Solve local NLP (10)	1.003	1.976	3.095
P	Update b_i and B_i	0.007	0.007	0.007
C	Update D	0.126	0.097	0.043
C	Update dual iterates (13a)	0.010	0.008	0.002
P	Update primal iterates (13b)	0.003	0.003	0.003
	Total time per iteration	1.149	2.091	3.150

Figure 3 shows the comparison with block BFGS based SQP method and standard ALADIN [20] with BFGS Hessian approximation. All algorithms use a damped block BFGS update. Here, the dual iterate of Algorithm 1 is initialized with zero. As discussed in [17], small block size yields poor performance in BFGS-SQP. Therefore, in order to make the comparison be fair enough, we choose $n = 5$. The results show that both Algorithm 1 and standard ALADIN [20] converge faster in terms of number of iterations.

[1] **P**: parallelizable, **C**: centralized

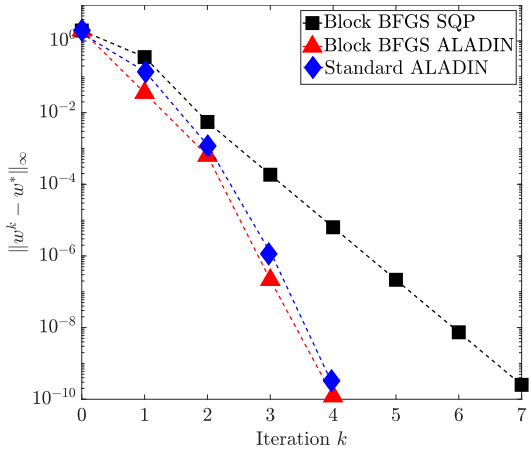

Fig. 3. Convergence comparison with block size $n = 5$

TABLE II

TOTAL TIME UNTIL CONVERGENCE IN [MS] FOR BLOCK BFGS-ALADIN (3 ITERATIONS) VS. BLOCK BFGS-SQP (5 ITERATIONS) VS STANDARD ALADIN [20] (3 ITERATIONS) WITH $n = 5$

	Parallel time	Centralized time	Total time
block BFGS-SQP	14.763	6.274	21.037
block BFGS-ALADIN	9.924	0.185	10.109
standard ALADIN [20]	10.113	3.263	13.376

For the particular example from Figure 3, Table II shows that if we set the stop tolerance be 10^{-6}, the total computational time of Algorithm 1 is approximate two times faster than block BFGS-SQP. Moreover, as discussed in [20], the standard ALADIN requires to compute the Jacobian matrices of the active constraints at local solutions and uses null space method to solve the large-scale coupled QP such that it spends much more time in the consensus step compared to Algorithm 1.

VI. CONCLUSION

This paper has presented a block BFGS based ALADIN method for solving (1). As established in Theorem 1, this BFGS-ALADIN variant can achieve a superlinear convergence rate while maintaining a favorable computational complexity. We have discussed the implementation details of the proposed method based on C++ library `PolyMPC` for NMPC. Our case study indicates that BFGS-ALADIN converges faster than a comparable BFGS-SQP method and the computation is more efficient than the standard ALADIN method when applied to a CSTR benchmark case study.

REFERENCES

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

[2] J. 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. *Found. Trends Mach. Learn.*, 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 *Proc. 51st IEEE Conference on Decision and Control (CDC)*, pages 6819–6824, Dec. 2012.

[5] F. E. Curtis and X. Que. A quasi-Newton algorithm for nonconvex, nonsmooth optimization with global convergence guarantees. *Math. Program. Comput.*, 7(4):399–428, 2015.

[6] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, University of Heidelberg, 2001.

[7] 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. *J. Process Control*, 12(4):577–585, 2002.

[8] M. Diehl, R. Findeisen, and F. Allgöwer. A stabilizing real-time implementation of nonlinear model predictive control. In *Real-Time PDE-Constrained Optimization*, pages 25–52. SIAM, 2007.

[9] Q. T. Dinh, I. Necoara, C. Savorgnan, and M. Diehl. An inexact perturbed path-following method for lagrangian decomposition in large-scale separable convex optimization. *SIAM J. Optim.*, 23(1):95–125, 2013.

[10] J. V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Math. Program. Comput.*, 7(3):289–329, 2015.

[11] 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.

[12] N. I. 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, 2003.

[13] G. Guennebaud, J. Benoît, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[14] A. Hamdi and S. K. Mishra. Decomposition methods based on augmented Lagrangians: a survey. In *Topics in Nonconvex Optimization*, pages 175–203. Springer, 2011.

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

[16] B. Houska, J. Frasch, and M. Diehl. An augmented Lagrangian based algorithm for distributed non-convex optimization. *SIAM J. Optim.*, 26(2):1101–1127, 2016.

[17] D. Janka, C. Kirches, S. Sager, and A. Wächter. An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal Hessian matrix. *Math. Program. Comput.*, 8(4):435–459, 2016.

[18] Y. Jiang, C. Jones, and B. Houska. A time splitting based real-time iteration scheme for nonlinear MPC. In *Proc. 58th IEEE Conference on Decision and Control (CDC)*, pages 2350–2355, Dec. 2019.

[19] Y. Jiang, J. Oravec, B. Houska, and M. Kvasnica. Parallel mpc for linear systems with input constraints. *IEEE Transactions on Automatic Control*, pages 1–1, 2020.

[20] D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl. A block based ALADIN scheme for highly parallelizable direct optimal control. In *Proc. 2016 American Control Conference (ACC)*, pages 1124–1129, Jul. 2016.

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

[22] P. Listov and C. Jones. Polympc: An efficient and extensible tool for real-time nonlinear model predictive tracking and path following for fast mechatronic systems. *Optimal Control Applications and Methods*, 41(2):709–727, 2020.

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

[24] B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Trans. Control Syst. Technol.*, 21(6):2432–2442, Nov. 2013.

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

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

[27] A. Rantzer. Dynamic dual decomposition for distributed control. In *Proc. 2009 American Control Conference (ACC)*, pages 884–888, Jun. 2009.

[28] J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory and Design, 2nd Edition*. Madison, WI: Nob Hill Publishing, 2017.

[29] R. T. Rockafellar. *Convex Analysis*. Princeton university press, 2015.

[30] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *Math. Prog. Comp.*, pages 1–36, 2020.

[31] A. Venkat, J. Rawlings, and S. Wright. Distributed model predictive control of large scale systems. *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 591–605, 2007.

[32] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.

[33] A. Zanelli, Q. Tran, and M. Diehl. Stability analysis of real-time methods for equality constrained MPC. *arXiv preprint arXiv:1912.03140*, 2019.

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

APPENDIX A. PROOF OF LEMMA 2

If Algorithm 1 converges to a regular KKT point while the curvature condition locally always holds, Lemma 1 indicate that BFGS update H^k satisfies

$$\left\|P(H^k - \nabla^2 F(x^*))\Delta x^k\right\| = \mathbf{o}\left(\left\|\Delta x^k\right\|\right) + \mathbf{O}\left(\left\|\xi^k - x^*\right\|^2\right) \quad (20)$$

with $\Delta x^k = x^k - \xi^k$. Here, [23, Theorem 6.6] can be used to construct (20) analogously. Note that compared to the proof of BFGS based SQP method, there is an addition term $\mathbf{O}\left(\left\|\xi^k - x^*\right\|^2\right)$ in the right hand side as the local solution ξ_i^k is not feasible with respect to the coupled affine equality constraint (1b). Here, we use a fact $Ax^k = Ax^*$. In order to show the primal inequality in (15), we denote by $\Delta\hat{x}^k = \hat{x}^k - \xi^k$ with $\hat{x}^k$ the solution of (6) with exact Hessian, $\nabla^2 F_i(\xi_i^k)$ for all $i = 1, ..., M$. Then, we have that

$$\begin{aligned}
&\|\Delta x^k - \Delta\hat{x}^k\| \\
&= \|(P\nabla^2 F(\xi^k))^{-1}(P\nabla^2 F(\xi^k)\Delta x^k + Pb^k)\| \\
&= \mathbf{O}(\|P(H^k - \nabla^2 F^k)\Delta x^k\|) + \mathbf{O}\left(\left\|\xi^k - x^*\right\|^2\right) \\
&\overset{(20)}{=} \mathbf{o}(\|\Delta x^k\|) + \mathbf{O}\left(\left\|\xi^k - x^*\right\|^2\right)
\end{aligned}$$

Here, we use the fact that $\|(P\nabla^2 F(\xi^k))^{-1}\|$ is bounded when ξ^k is sufficiently close to x^*. Then, we apply triangular inequality such that

$$\begin{aligned}
\|x^k - x^*\| \leq& \|\Delta x^k - \Delta\hat{x}^k\| + \|\xi^k + \Delta\hat{x}^k - x^*\| \\
\leq& \mathbf{o}(\|\Delta x^k\|) + \mathbf{O}(\|\xi^k - x^*\|^2)
\end{aligned} \quad (21)$$

where the first part of last inequality follows one standard Newton step. Thus, the primal inequality in (15) can be obtained according to $\|\Delta x^k\| = \mathbf{O}(\|\xi^k - x^*\|)$ [23, Thm. 3.7]. Next, in order to prove the dual inequality, we use (13a) twice and have

$$\begin{aligned}
\lambda^k &= D^k\left[\sum_{i=1}^{M} A_i\left(\xi_i^k - B_i^k b_i^k\right)\right], \\
\text{and} \quad \lambda^* &= D^k\left[\sum_{i=1}^{M} A_i\left(x_i^* - B_i^k b_i^*\right)\right].
\end{aligned}$$

Then, we have that

$$\begin{aligned}
&\|\lambda^k - \lambda^*\| \\
&\leq \left\|D^k A(\xi^k - x^*)\right\| + \left\|D^k AB^k(b^k - b^*)\right\| \\
&\leq \|D^k AB^k(H^k - \nabla^2 F(x^*))(\xi^k - x^*)\| + \mathbf{O}(\|\xi^k - x^*\|^2) \\
&\leq \mathbf{o}(\|\Delta x^k\|) + \mathbf{O}(\|x^k - x^*\|) + \mathbf{O}(\|\xi^k - x^*\|^2) \\
&\leq \mathbf{o}(\|\xi^k - x^*\|) + \mathbf{O}(\|\xi^k - x^*\|^2)
\end{aligned}$$

which indicates the dual inequality in (15) is satisfied.