Robust Distributed and Localized Model Predictive Control

Carmen Amo Alonso, Jing Shuang (Lisa) Li, Nikolai Matni and James Anderson

Abstract—We present a robust Distributed and Localized Model Predictive Control (rDLMPC) framework for large-scale structured linear systems. The proposed algorithm uses the System Level Synthesis to provide a distributed closed-loop model predictive control scheme that is robust to exogenous disturbances. The resulting controllers require only local information exchange for both synthesis and implementation. We exploit the fact that for polytopic disturbance constraints, SLS-based distributed control problems have been shown to have structure amenable for distributed optimization techniques. We show that similar to the disturbance-free DLMPC algorithm, the computational complexity of rDLMPC is independent of the size of the global system. To the best of our knowledge, robust DLMPC is the first MPC algorithm that allows for the scalable distributed computation of distributed closed-loop control policies in the presence of additive disturbances.

I. INTRODUCTION

Model Predictive Control (MPC) has proven to be very successful in many industrial applications. It provides an effective way to control systems that are subject to hard state and input constraints while ensuring good performance. However, one key consideration when dealing with constrained control in real-world applications is the presence of disturbances. While nominal MPC enjoys some intrinsic robustness [1], the resulting closed-loop can be destabilized by an arbitrary small disturbance [2]. For this reason, efforts have been made to extend nominal MPC approaches to the robust setting.

Our focus is on designing robust distributed MPC controllers for heterogeneous networks with additive disturbances. Broadly speaking, there are two existing methods to tackling this problem: the first builds on centralized robust MPC techniques based on a pre-computed stabilizing controller (tube-MPC [3], constraint tightening [4], etc.), and ports them to the distributed setting. Although this extension results in computationally efficient algorithms, they are usually too restrictive in their assumptions: for example, the subsystems must be dynamically decoupled [5], or a pre-computed structured stabilizing controller is needed [6] (which requires solving an NP-hard problem [7]). The second approach is based on computing a dynamic structured feedback policy using a suitable parameterization. In the seminal paper [8], the problem of synthesizing robust control policies is solved through convex optimization by virtue of using a disturbance based parameterization of the control policy. Other works have been developed based on these ideas [9], and although they are able to formulate the synthesis of a structured closed-loop control policy as a convex problem, the resulting optimization is not amenable for distributed optimization techniques. Along this line, recent works exploit the System Level Synthesis (SLS) parameterization to design robust MPC controllers [10], [11]. However, it is not yet clear how these formulations can be applied in a distributed setting.

Despite these efforts, synthesizing and implementing a closed-loop MPC controller that is robust to additive disturbances in a distributed fashion remains an open problem. For large-scale networks, it is desirable for the control scheme to respect communication limitations among sub-controllers. Here we present an algorithm based on the SLS framework that achieves both these goals. To do so, we exploit the SLS parameterization, which characterizes achievable closed-loop system responses as linear maps from disturbances to input and states, and integrate our prior results on Distributed and Localized MPC (DLMPC) [12] and robust invariant linear-time-invariant control [13] into a novel robust Distributed and Localized MPC (rDLMPC) algorithm.

Our rDLMPC algorithm provides a convex formulation, has robust stability guarantees, and has local (i.e., per sub-system) computational and implementation complexity that scales independently of the number of states in the system model. Moreover, it allows for the incorporation of locality constraints, where information exchanges are restricted to local sub-controllers only. The concept of locality is central to the SLS framework: it permits scalable synthesis and algorithms and is a desirable property for the closed-loop system. Broadly speaking, a localized system response limits the impact of a disturbance affecting a subsystem in the network to a small neighborhood of connected subsystems. We formalize this idea in Section III. A by-product of localization is that the resulting MPC sub-controllers only have to solve a low-dimensional optimization problem, and only exchange information with controllers in a local neighborhood. We demonstrate through simulation that the computational overhead and decrease in performance due to conservatism is only slightly higher than for the nominal case [12], and that the complexity of the subproblems solved by each subsystem is independent of the size of the system.

Notation: Bracketed indices denote time-step of the real system, i.e., the system input is $u(t)$ at time $t$, not to be confused with $x_t$ which denotes the predicted state $x$ at time
We use \( w \) of exogenous disturbances within a set of constraints that must be robustly satisfied for all realizations of a predictive controller optimizing a nominal objective, subject to control problem over a finite prediction horizon \( T \). We define a submatrix of \( K \) or \( K^T \) a matrix of compatible dimension representing the value of \( T \) time-step computed at time \( t \). We further assume that the information \( G \) is an directed graph, i.e., \( G \subseteq \{ 1, \ldots, n \} \). We consider a discrete-time linear time invariant (LTI) system, with dynamics:

\[
\dot{x}(t+1) = Ax(t) + Bu(t) + w(t),
\]

where \( x(t) \in \mathbb{R}^n \) and \( u(t) \in \mathbb{R}^p \) are the state and control input respectively, and \( w(t) \in \mathbb{R}^n \) is an exogenous disturbance. We view system (1) as being composed of interconnected subsystems, so the state, control input and disturbance can be partitioned into \( \{ x_i(t) \}, \{ u_i(t) \}, \{ w_i(t) \} \) for each subsystem \( i \) inducing a compatible block structure \( [A]_{ij}, [B]_{ij} \) in the dynamic matrices \( (A, B) \). We model the interconnection topology as a time-invariant unweighted directed graph \( G_{(A, B)}(E, V) \), where each subsystem \( i \) is identified with a vertex \( v_i \in V \) and an edge between subsystems \( i \) and \( j \) \( e_{ij} \in E \) exists whenever \( \{ A \}_{ij} \neq 0 \) or \( \{ B \}_{ij} \neq 0 \). We further assume that the information exchange topology between sub-controllers matches that of the underlying system, and can thus be modeled by the same graph \( G_{(A, B)}(E, V) \).

We study the case where the control input is a model predictive controller optimizing a nominal objective, subject to constraints that must be robustly satisfied for all realizations of exogenous disturbances within a set \( w(t) \in \mathcal{W} \). As is standard, at each time step the controller solves an optimal control problem over a finite prediction horizon \( T \) using the current state as the initial condition. Hence, at time step \( \tau \), we solve:

\[
\begin{align*}
\text{minimize} & \sum_{t=0}^{T-1} f_t(x_t, u_t) + f_T(x_T) \\
\text{subject to} & \quad x_T = x(\tau), \quad x_{t+1} = Ax_t + Bu_t + w_t, \\
& \quad x_T \in \mathcal{X}_T, \quad x_\tau \in \mathcal{X}_\tau, \quad u_t \in \mathcal{U}_t, \forall w_t \in \mathcal{W}_t \\
& \quad \gamma_t(x_{0:t}, u_{0:t-1}), \quad t = 0, \ldots, T - 1,
\end{align*}
\]

where the \( f_t(\cdot, \cdot) \) and \( f_T(\cdot) \) are convex cost functions, \( \mathcal{X}_\tau \), \( \mathcal{U}_t \) and \( \mathcal{W}_t \) are convex polytopes containing the origin, and \( \gamma_t(\cdot) \) is the control policy we optimize over.

Our goal is to define an algorithm that allows us to solve the robust MPC problem \( \mathcal{P} \) in a distributed manner while respecting local communication constraints. To achieve this goal, we impose that information exchange - as defined by the graph \( G_{(A, B)}(E, V) \) - is localized to a subset of neighboring sub-controllers. Analogous to previous work [12], [14], we use the notion of a \( d \)-local information exchange constraint via the d-outgoing and d-incoming sets of the subsystems to restrict sub-controllers exchanges of their state and control actions to only neighbors at most \( d \)-hops away.

**Definition 1.** For a graph \( G(\mathcal{V}, \mathcal{E}) \), the d-outgoing set of subsystem \( i \) is \( \text{out}_i(d) := \{ v_j | \text{dist}(v_i \rightarrow v_j) \leq d \in \mathbb{N} \} \). The d-incoming set of subsystem \( i \) is \( \text{in}_i(d) := \{ v_j | \text{dist}(v_j \rightarrow v_i) \leq d \in \mathbb{N} \} \). Note that \( v_i \in \text{out}_i(d) \cap \text{in}_i(d) \) for all \( d \geq 0 \).

Hence, we can achieve \( d \)-local information exchange on the distributed MPC problem (2) by enforcing the following constraint on each local control policy:

\[
[u_t]_{ij} = \gamma_t \left( [x_{0:t}]_{ij}, [u_{0:t-1}]_{ij}, [A]_{ij,k} \in \text{in}_i(d), [B]_{ij,k} \in \text{in}_i(d) \right),
\]

for all \( t = 0, \ldots, T \) and \( i = 1, \ldots, N \), where \( \gamma_t \) is a measurable function of its arguments. This implies that the closed-loop control policy for sub-controller \( i \) can be computed using only states, control actions, and system models collected from \( d \)-hop incoming neighbors according to the communication topology \( G_{(A, B)} \).

In the remaining sections of this paper, we will show that by exploiting this information exchange topology together with suitable structural compatibility assumptions between the cost function and state and input constraints, we achieve an MPC controller robust to additive disturbances in which both synthesis and implementation of a control action at each subsystem are localized, i.e., depend only on state, control action, and plant model information from \( d \)-hop neighbors, where the size of the local neighborhood \( d \) is a design parameter. Analogous to [12], we achieve this by imposing appropriate \( d \)-local structural constraints on the closed-loop system responses of the system by leveraging the SLS framework to reformulate the MPC problem (2). In addition, in this work we take advantage of recent robustness results in constrained SLS [13], which allows us to extend the results in [12] to the robust setting.

**III. System Level Synthesis: MPC Reformulation**

In this section we introduce relevant concepts of the SLS framework [15]–[17] such as locality [17], [18] and how to encode state and input constraints [13], and discuss their central role in enabling a rDLMPC framework. We end by using these tools to formulate the robust Distributed and Localized MPC problem in the SLS framework.
A. Time Domain System Level Synthesis

Given the dynamics of system 1, let \( u(t) = K_t(x(0), ..., x(t)) \) be the control input from a causal linear-time varying state feedback controller where \( K_t \) is to be designed. The closed-loop dynamics over a finite time horizon \( t = 0, ..., T \) can be compactly written as

\[
x = Z(\hat{A} + \hat{B}K)x + w,
\]

where \( Z \) is the block-downshift matrix

\[
\hat{A} := \text{blkdiag}(A, A, ..., A), \quad \text{and} \quad \hat{B} := \text{blkdiag}(B, B, ..., B).
\]

The block lower-triangular operator \( K \) represents the control law defined by the linear-time-varying gains \( \{ K_t \}_{t=0}^T \) and the finite horizon signals \( x, u, w \) correspond to the state, control input and disturbance respectively. By convention, we define the disturbance to contain the initial condition, i.e. \( w = [x_0^T, w_1^T, ..., w_{T-1}^T]^T \).

By noting that \( u = Kx \), the closed-loop behavior of the system can entirely be characterized by the block lower triangular operators \( \Phi_x \) and \( \Phi_u \) (mapping disturbance to state and control action, respectively) defined as:

\[
\begin{align*}
\Phi_x &= \Phi(u(x)) = (I - Z(A + BK))^{-1}\Phi_x \Phi_u \Phi_w =: \Phi_x w \\
\Phi_u &= K(I - Z(A + BK))^{-1}w =: \Phi_u w.
\end{align*}
\]

The system responses \( \Phi_x \) and \( \Phi_u \) parameterize the set of achievable closed-loop maps from the disturbance \( w \) to state \( x \) and control input \( u \) respectively, if and only if they are constrained to lie in the affine subspace

\[
Z_{AB}\Phi := [I - Z\hat{A} - Z\hat{B}] \begin{bmatrix} \Phi_x \\ \Phi_u \end{bmatrix} = I,
\]

where we use \( Z_{AB} \) and \( \Phi \) as short-hand notation for the constraint matrix and joint state/input system response, respectively. In particular, any system responses \( \Phi_x \) and \( \Phi_u \) satisfying equation (5) can be achieved via the control law \( K(\Phi_x, \Phi_u) = \Phi_x^{-1}\Phi_u \), where \( K(\Phi_x, \Phi_u) \) is also block lower triangular. For a formal statement and a proof, please see [15, Theorem 2.1].

The SLS framework relies on the affine parametrization to reformulate optimal control problems as a search over system responses \( \Phi_u \) satisfying equation (5), rather than an optimization problem over states and inputs \( \{x, u\} \). Using parametrization 4, we reformulate the MPC subroutine in terms of the system responses as

\[
\begin{align*}
\text{minimize} & \quad f(\Phi(1)x_0) \\
\text{s.t.} & \quad x_0 = x(\tau), \quad Z_{AB}\Phi = I, \quad \Phi w \in \mathcal{P} \quad \forall w \in \mathcal{W},
\end{align*}
\]

where the polytope \( \mathcal{W} := \bigotimes_{t=0}^T\mathcal{W}_t \) and the polytope \( \mathcal{P} \) is defined so that \( \Phi w \in \mathcal{P} \) if and only if \( x_T \in \mathcal{X}_T, x_t \in \mathcal{X}_t, \text{ and } u_t \in \mathcal{U}_t, \forall t = 0, ..., T-1, \forall w \in \mathcal{W} \). Note that \( \Phi(1)x_0 \) appears in the objective function as it corresponds to the nominal (disturbance-free) state and input responses.

1Without loss we assume the control policy is linear, as an affine control policy \( u(t) = K_t(x(0) : t) + v_t \) can always be written as a linear policy acting on the augmented state \( \hat{x} = [x^T \ 1]^T \).

2A matrix with identity matrices along its first block sub-diagonal and zeros elsewhere.

In what follows we discuss the advantages of this parametrization, and show how local structure can be imposed in this reformulation in a transparent manner via affine constraints. We also discuss how one can deal with the robust constraints while maintaining the local structure of the problem.

1) Locality: Here we illustrate how to enforce the information sharing constraint (3) in the SLS framework, and how localized system responses result in a localized controller implementation.

A key advantage of using the SLS framework is that the system responses not only parametrize the closed-loop map but also provide a controller realization. In particular, the controller achieving the system responses (4) can be implemented as

\[
u = \Phi_u \hat{w}, \quad x = (I - \Phi_x)\hat{w}, \quad \hat{w} = x - \hat{x},
\]

where \( \hat{x} \) is the nominal state trajectory, and \( \hat{w} = Z\hat{w} \) is a delayed reconstruction of the disturbance. The advantage of this controller implementation, as opposed to \( u = \Phi_x\Phi_u^{-1}x \), is that any structure imposed on the system response \( \{ \Phi_u, \Phi_x \} \) translates directly to structure on the controller implementation (7). This is particularly relevant for imposing locality constraints, and we will show how locality in system responses translates into locality of the controller implementation.

Definition 2. Let \( [\Phi_x]_{ij} \) be the submatrix of system response \( \Phi_x \) describing the map from disturbance \( [w]_j \) to state \( [x]_i \) of subsystem \( i \). The map \( \Phi_x \) is d-localized if and only if for every subsystem \( j, [\Phi_x]_{ij} = 0 \ \forall \ i \notin \text{out}_d(j). \) The definition for d-localized \( \Phi_u \) is analogous (but maps disturbance \( [w]_j \) to the input \( [u]_i \)).

When the state system response \( \Phi_x \) is d-localized, subsystem \( i \) only needs to collect the local subset \( \{w\}_{j \in \text{in}_d(i)} \) of disturbance estimates \( \hat{w} \) from its d-incoming set of neighbors to compute its local disturbance estimate \( \hat{w}_i \). Similarly, when the input system response \( \Phi_u \) is d-localized, then only a local subset \( \{w\}_{j \in \text{in}_d(i)} \) of the estimated disturbances \( \hat{w} \) is needed by subsystem \( i \) to compute its control action \( [u]_i \). Therefore, if \( \Phi_x \) and \( \Phi_u \) are d-localized each subsystem only needs to collect information from its d-incoming set to implement the control law and it only needs to share information with its d-outgoing set to allow for its neighboring subsystems to implement their local control laws.

Definition 3. A subspace \( \mathcal{L}_d \) enforces a d-locality constraint if \( \Phi_x, \Phi_u \in \mathcal{L}_d \) implies that \( \Phi_x \) is d-localized and \( \Phi_u \) is \((d + 1)\)-localized.\(^3\)

Remark 1. Despite locality constraints always being an affine constraint, an SLS problem subject to locality constraints might have an empty solution space since not all systems are d-localizable. The size of the locality region \( d \)

\(^3\)\( \Phi_u \) is enforced to be \((d + 1)\)-localized because to confine the effects of a disturbance within the region of size \( d \), controllers at distance \( d + 1 \) must take action.
is a design parameter, and for the remainder of the paper we assume that system (1) is d-localizable for some $d < N$. The results presented here extend naturally to virtually localizable systems via the robust variant of SLS [15], [19].

2) State and Input Constraints: Next we present a method for dealing with the robust state and input constraints. We emphasize how the resulting SLS reformulation retains the locality structure of the original problem, which will be key to enabling distribution of the problem across the different subsystems as we will discuss in Section IV.

As stated, problem (6) is a robust optimization problem since the convex constraint $\Phi_w \in \mathcal{P}$ has to hold for all $w \in \mathcal{W}$. Let us define the polytope $\mathcal{P}$ as $\mathcal{P} := \{[x^T \ u^T]^T : H[x^T \ u^T]^T \leq h\}$, and since $w$ is a polytopic constraint $\mathcal{W} := \{w : Gw \leq g\}$. A simple duality argument allows for a tractable reformulation of problem (6) (see [13] for details and references therein):

**Theorem 1** (Theorem 1 of [13]), Problem (6) can be reformulated as

$$\begin{align*}
\min_{\Phi, \Xi \geq 0} & \quad f(\Phi\{1\}x_0) \\
\text{s.t.} & \quad x_0 = x(t), \quad Z_{AB}\Phi = I, \\
& \quad H\Phi\{1\}x_0 + \Xi g \leq h, \quad H\Phi\{2 : T\} = \Xi G,
\end{align*}$$

where

$$G := \text{blkdiag}(G_1, ..., G_{T-1}) \text{ and,}$$

$$H := \text{blkdiag}(H_{x,1}, ..., H_{x,T}, H_{u,1}, ..., H_{u,T-1}),$$

with the constraint $\Xi \geq 0$ satisfied component-wise. Further, if each of the $\{G_i, H_{x,t}, H_{u,t}\}_{t=1}^{T}$ are block-diagonal with structure compatible with subsystem-wise decompositions of $x$ and $u$, then imposing that $\Phi \in \mathcal{L}_d$ allows us to take $\Xi \in \mathcal{L}_d$ without loss of generality.

Theorem 1 allows us to solve the robust MPC problem (2) using standard convex optimization methods, and further preserves the locality structure of the original problem so long as the constraint matrices $H$ and $G$ defining the polytopes $\mathcal{P}$ and $\mathcal{W}$, respectively, have suitable block-diagonal structure. We formalize the implications of these structural decoupling assumptions on the constraint sets, and introduce an analogous decoupling assumption on the objective function, here.

**Assumption 1.** The objective function $f_i$ in formulation (2) is such that $f_i(x) = \sum f_j([x_i], [u_i])$. The constraint sets in formulation (2) are such that $x \in \mathcal{X}_t = \mathcal{X}_i^1 \times ... \times \mathcal{X}_i^N$, where $x \in \mathcal{X}_i$ if and only if $[x_i]_t \in \mathcal{X}_i^t$ for all $i$ and $t \in \{0, ..., T\}$, and $i \in \mathcal{U}_t$. In particular, this implies that the matrices $H$ and $G$ defining the polytopes $\mathcal{P}$ and $\mathcal{W}$, respectively, are block-diagonal as described in Theorem 1.

**Remark 2.** Although Theorem 1 focuses on disturbances constrained to polytopic constraint sets, the argument can be extended to any setting for which a closed-form expression for $\sum_{i=0}^{N} \sup_{[w]_i} e_j[H\Phi\{i\}[w]_i] \leq h$ can be computed for $e_j$ the standard basis unit. For example, if $\mathcal{W}_i = \{w : ||w|| \leq \sigma_w\}$, then this reduces to $\sum_{i=0}^{N} \sigma_w ||[H\Phi\{i\}[w]_i] || \leq h$, for $|| \cdot ||$, the dual norm. Note however, that for block-diagonal $H$ and localized $\Phi$, this constraint will reduce to $\sum_{i=0}^{N} ||[H\Phi\{i\}[w]_i] || \leq h$, which can be similarly enforced using only local information.

**B. System Level Synthesis based Robust MPC**

Here we combine the previously introduced locality and state/input robust constraints with the SLS reformulation of the MPC subroutine (2), and present the robust Distributed and Localized MPC problem. We reformulate subroutine (2) in terms of the SLS variables as in equation (6), and impose appropriate $d$-local structural constraints on the closed-loop system responses of the system. Hence, both the synthesis and implementation of the control action at each subsystem is localized, i.e., the control policy depends only on state, control, and plant model information from $d$-hop neighbors.

While it was not possible to incorporate locality as expressed in equation (5) into the original problem (2) in a convex and computationally tractable manner, it is straightforward to do so via the affine constraint $\Phi_x, \Phi_u \in \mathcal{L}_d$ as per Definition 2 with the only requirement that some mild compatibility assumptions as per Assumption 1 between the cost functions, state and input constraints, and $d$-local information exchange constraints are satisfied.

In particular, Assumption 1 imposes that neither the constraints nor the objective function can couple two subsystems in the network, and therefore they are fully separable among the different subsystems. Notice that even though this assumption is written in terms of the objective function and the constraints of problem (2), it directly translates to the objective function and constraint set of the SLS reformulation (3). Moreover, this local structure holds for the duality based reformulation of the robust SLS problem described in Theorem 1. Therefore, the rDLMPC problem (2) subject to local information constraint (3) can be written as

$$\begin{align*}
\min_{\Phi, \Xi \geq 0} & \quad f(\Phi\{1\}x_0) \\
\text{s.t.} & \quad x_0 = x(t), \quad Z_{AB}\Phi = I, \quad \Phi, \Xi \in \mathcal{L}_d, \\
& \quad H\Phi\{1\}x_0 + \Xi g \leq h, \quad H\Phi\{2 : T\} = \Xi G.
\end{align*}$$

One of the advantages of using the SLS framework is that this approach is automatically optimizing over closed-loop policies – as opposed to open-loop control inputs – since the optimization is carried out over the closed-loop maps $\Phi_x$ and $\Phi_u$. In [12], we discuss and formalize in detail the advantages of using SLS over other feedback parameterizations such as [8], [9] for distributed MPC. In summary, SLS allows for localized communication constraints (3) as well as the robust state and input constraints to be transparently enforced through convex constraints on the system responses. This in turn results in a problem structure that is amenable to using distributed optimization for the computing of the resulting control policies, and as we will show in the next
section, sub-controllers only need to collect local information to do so.

IV. ROBUST DISTRIBUTED AND LOCALIZED MPC

We present a brief overview of the ADMM algorithm, and then show how it can be used to distribute the rDLMPC subroutine into local subproblems requiring d-local information only. We finish by analyzing the convergence and complexity of the resulting rDLMPC algorithm.

A. The Alternating Method of the Multipliers

The varying penalty parameter ADMM algorithm [20, §3.4] solves problems of the form

\[
\text{minimize } f(x) + g(y) \text{ s.t. } Ax + By = c.
\]

The ADMM iterates are as follows:

\[
\begin{align*}
    x^{k+1} &= \arg \min_x f(x) + \frac{\rho^k}{2} \|Ax + By - c + z^k\|^2_2 \\
y^{k+1} &= \arg \min_y g(y) + \frac{\rho^k}{2} \|Ax^{k+1} + By - c + z^k\|^2_2 \\
z^{k+1} &= z^k + Ax^{k+1} + By^{k+1} - c,
\end{align*}
\]

where

\[
\rho^k := \begin{cases} 
\tau \rho & \text{if } \|r^{k+1}\|_2 > \mu_1 \|s^{k+1}\|_2, \\
\tau^{-1} \rho & \text{if } \|r^{k+1}\|_2 < \mu_2 \|s^{k+1}\|_2, \\
\rho^k_{\text{max}} & \text{otherwise,}
\end{cases}
\]

with

\[
\rho_{\text{max}} := \rho \frac{\text{dim}}{\text{dim}^{1/2}} \text{A}\frac{\text{dim}}{\text{dim}^{1/2}} B(y^k - y^{k-1}),
\]

and \(\tau, \mu_1, \mu_2 > 0\). This version of ADMM is known as the varying penalty parameter.

The algorithm is said to have converged at iteration \(k\) if \(\|r^k\|_2 \leq \epsilon_p\) and \(\|s^k\|_2 \leq \epsilon_d\) for some small primal and dual tolerances, \(\epsilon_p > 0\) and \(\epsilon_d > 0\). To guarantee convergence, \(\rho\) is set to a constant value \(\rho_{\text{max}}\) after a certain number of iterations. Residual, objective, and dual variable convergence are guaranteed under mild regularity assumptions, assuming a non-empty feasible set (as this implies strong duality holds, as Slater’s condition always holds when constraints are proper, and convex function for all choices of \(x_0 \neq 0\).

Assumption 2. The objective function \(f(\Phi x_0)\) is a closed, proper, and convex function for all choices of \(x_0 \neq 0\).

B. The Robust DLMPC Algorithm

We will show how the ADMM algorithm can be used to distribute \(\Phi\) into a local subproblem per system. To do so, we exploit the localization of the cost function and the constraints, which by Assumption [I] imply that

\[
f(x, u) = \sum_{i=1}^N f_i(x_i, u_i),
\]

\[
[x^T, u^T]^T \in \mathcal{P} \text{ if and only if } [H_i][x_i, u_i] \leq [h_i], \forall i,
\]

with each submatrix \(H_i\), corresponding to the constraints of subsystem \(i\) for \(i = 1, \ldots, N\), i.e., \(H = \text{blkdiag}([H_1], \ldots, [H_N])\). Using the definition of the SLS system responses [I], we can equivalently write these conditions in terms of \(\Phi\) as

\[
f(\Phi) = \sum_{i=1}^N f_i([\Phi_i]_1, \ldots, [\Phi_i]_N),
\]

\[
\Phi \in \mathcal{P} \text{ if and only if } [H_i]([\Phi]_i, \ldots, [\Phi]_N) \leq [h_i], \forall i,
\]

where \(\tau_i\) is the set of rows in \(\Phi\) corresponding to subsystem \(i\), i.e., those that parametrize \([x_i]\) and \([u_i]\). These separability features are formalized in the following definition:

Definition 4. Given the partition \(\{c_1, \ldots, c_k\}\), a functional/set is column-wise separable if:

- For a functional, \(g(\Phi) = \sum_{i=1}^k g_i(\Phi(:, c_i))\) for some functionals \(g_i\) for \(i = 1, \ldots, k\).
- For a set, \(\Phi \in \mathcal{P}\) if and only if \([\Phi(:, c_i)]_i \in \mathcal{P}_i\) for some sets \(\mathcal{P}_i\) for \(i = 1, \ldots, k\).

An analogous definition exists for row-wise separable functionals and sets, where the partition \(\{t_1, \ldots, t_k\}\) entails the rows of \(\Phi\), i.e., \([\Phi(t_i, :)]_i\).

When the objective function and all the constraints of an optimization problem share the same type of separability with respect to a partition of cardinality \(k\), then the optimization trivially decomposes into \(k\) independent subproblems. However, this is not the case for problem [9], where the constraint \(\Phi\) is neither row-wise nor column-wise separable. To make it amenable for distribution, we propose the following reformulation:

\[
\min_{\Phi, \Psi, \Xi, \Omega \geq 0} f(\Phi x_0) \quad \text{s.t.} \quad x_0 = x(t), \ Z_{AB} \Psi = I, \ \Phi, \Xi, \Psi, \Omega \in \mathcal{L}_d, \ \Omega x_0 + \Xi g \leq h,
\]

\[
\begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} [\Psi]_1 & 0 \\ [\Phi]_1 & [\Xi]_2 : [T] \end{bmatrix} = \begin{bmatrix} \Phi & 0 \\ \Omega & \Xi G \end{bmatrix},
\]

where \(\Phi, \Omega\) to refer to only the first block-column of the homonymous of the matrices \([\Phi]_1\), \([\Omega]_1\).

In this reformulation, it is straightforward to distinguish between the column-wise separable components (everything involving \(\Psi\)), and the row-separable components (the objective function and the remaining constraints that do not contain \(\Psi\)). Problems of this nature are known as partially separable problems, and are amenable for distribution via ADMM. In particular, we can decompose problem [11] as:

\[
[\Phi^{k+1}, \Omega^{k+1}, \Xi^{k+1}] = \begin{cases} \arg \min_{\Phi, \Omega, \Xi \geq 0} f(\Phi x_0) + \frac{\rho}{2} \|\Phi - \tilde{H} \Psi^k + \Lambda^{k}\|_F^2 \\ \text{s.t.} \quad x_0 = x(t), \ \Omega x_0 + \Xi g \leq h, \ \Phi, \Xi, \Omega \in \mathcal{L}_d \end{cases}
\]

\[
[\Psi^{k+1}] = \begin{cases} \arg \min_{\Psi} \|\Phi - \tilde{H} \Psi^k + \Lambda^{k}\|_F^2 \\ \text{s.t.} \quad Z_{AB} \Psi = I, \ \Psi \in \mathcal{L}_d \end{cases}
\]

\[
\Lambda^{k+1} = \Lambda^{k} + \Phi^{k+1} - \tilde{H} \Psi^{k+1},
\]
where for ease of notation we define
\[ \tilde{\Phi} := \begin{bmatrix} \Phi & 0 \\ \Omega & \Xi_G \end{bmatrix} \] and \( \tilde{H} \tilde{\Psi} := \begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} \Psi \{1\} & 0 \\ \Psi \{2 : T\} \end{bmatrix} \).

In equations (12), iterates (12a) is row-wise separable, (12b) is column-wise separable, and (12c) is both row and column-wise separable. We take advantage of the structure of these subproblems and separate them with respect to a row and column partition induced by the subsystem-wise partitions of the state and control inputs, \([x_i]_r\) and \([u_i]_r\) for each subsystem \(i\). Each of these row and column subproblems resulting from the distribution across subsystems can be solved independently and in parallel, where each subsystem solves for its corresponding row and column partition. Furthermore, since locality constraints are imposed, the decision variables \(\Phi, \Psi, \Xi, \Lambda\) have a sparse structure. This means that the length of the rows and columns that a subsystem solves for is much smaller than the length of the rows and columns of \(\Phi\). For instance, when considering the column-wise subproblem evaluated at subsystem \(i\), the \(j^{th}\) row of the \(i^{th}\) column partitions of \(\Phi_i\) and \(\Phi_o\) is nonzero only if \(j \in \cup_{t \in \text{out},(d)} \Omega_k\) and \(j \in \cup_{t \in \text{in},(d)} \Omega_k\) respectively. An algorithm to find the relevant components for each subsystem rows and columns can be found in Appendix A of [17].

Therefore, the subproblems that subsystem \(i\) solves for are:
\[
[\Phi]_{i_r}^{k+1}, [\Omega]_{i_r}^{k+1}, [\Xi]^{k+1} = \arg \min \begin{bmatrix} \Psi \{1\} \end{bmatrix}_{i_r}^{k+1} + \begin{bmatrix} \Psi \{2 : T\} \end{bmatrix}_{i_r}^{k+1} \begin{bmatrix} I \end{bmatrix}_{i_r}^{k+1} = \begin{bmatrix} \Psi \{1\} \end{bmatrix}_{i_r}^{k+1} + \begin{bmatrix} \Psi \{2 : T\} \end{bmatrix}_{i_r}^{k+1}
\]

\[
\begin{bmatrix} \Psi \{1\} \end{bmatrix}_{i_r}^{k+1} = \begin{bmatrix} \Psi \{2 : T\} \end{bmatrix}_{i_r}^{k+1} = \begin{bmatrix} \Psi \{1\} \end{bmatrix}_{i_r}^{k+1} + \begin{bmatrix} \Psi \{2 : T\} \end{bmatrix}_{i_r}^{k+1}
\]

where to simplify notation we denote as \([\Phi]_{i_r}\) the submatrix of \(\Phi\) formed by the nonzero components of the relevant rows for subsystem \(i, r_i\), and as \([\Phi]_{i_r}\) the submatrix of \(\Phi\) formed by the nonzero components of the relevant columns for subsystem \(i, c_i\). We use a similar bracket notation for the vectors and matrices that multiply the decision variables to indicate that those are just composed from the relevant components of their global versions.

The following lemma will be useful later for speeding up the ADMM iterates.

**Lemma 1.** Let \(z^*(M, v, P, q) := \arg \min \|Mz - v\|_2^2\) s.t. \(Pz = q\). Then
\[
\begin{bmatrix} z^* \\ \mu^* \end{bmatrix} = \begin{bmatrix} MM^T & P^T \\ P & 0 \end{bmatrix}^{\dagger} \begin{bmatrix} M^Tv \\ q \end{bmatrix},
\]

where \(\dagger\) denotes pseudo-inverse, is the optimal solution and \(\mu^*\) is the corresponding optimal Lagrange multiplier.

**Proof.** The proof follows directly from applying the KKT conditions to the optimization problem. By the stationarity condition, \(M^TMz^* - M^Tv + PT\mu^* = 0\), where \(z^*\) is the solution to the optimization problem and \(\mu^*\) the optimal Lagrange multiplier vector. From the primal feasibility condition, \(Pz^* = q\). Hence, \(z^*\) and \(\mu^*\) are computed as the solution to this system of two equations. \(\square\)

Applying lemma 1 the solution to (13b) is:
\[
[\Psi \{1\}]_{i_r}^{k+1} = z^* \begin{bmatrix} [I]_{i_r} \end{bmatrix}_{i_r} + \begin{bmatrix} \Psi \{2 : T\} \end{bmatrix}_{i_r}^{k+1} = \begin{bmatrix} [I]_{i_r} \end{bmatrix}_{i_r} + \begin{bmatrix} [I]_{i_r} \end{bmatrix}_{i_r}^{k+1}
\]

where we split \(\Lambda\) into block matrices consistent with the structure of \(\Phi\), i.e. \(\Lambda = \begin{bmatrix} \Lambda_1 & 0 \\ \Lambda_2 & \Lambda_3 \end{bmatrix}\).

Notice that the number of nonzero components for both the rows and the columns is much smaller than the size of the network \(N\) since it is determined by the size of the local neighborhood \(d\) through the locality constraints. In turn, this implies that the subsystem only requires small submatrices from the plant matrices and the constraints \(A, B, H,\) etc., to perform the computations. Therefore, the robust MPC subroutine can be solved in parallel in a distributed manner across the subsystems of the network, where each solves for a local patch of the system responses using local information only.

In Algorithm 1 we present the rDL MPC algorithm that each sub-controller executes.

**Algorithm 1** Subsystem \(i\) rDL MPC implementation

1. Measure local state \([x(t)]_i, k \leftarrow 0\).
2. Share the measurement with neighbors in \(\text{out},(d)\).
3. Solve optimization problem (13a).
4. Share \([\Phi]_{i_r}^{k+1}\) with \(\text{out},(d)\). Receive the corresponding \([\Phi]_{j_r}^{k+1}\) from \(\text{in},(d)\) and build \([\Phi]_{i_r}^{k+1}\).
5. Solve optimization problem (13b) via the closed form solution (14).
6. Share \([\Psi]_{i_r}^{k+1}\) with \(\text{out},(d)\). Receive the corresponding \([\Psi]_{j_r}^{k+1}\) from \(\text{in},(d)\) and build \([\Psi]_{i_r}^{k+1}\).
7. Perform the multiplier update step (13c).
8. if \(\|\begin{bmatrix} \Phi]_{i_r}^{k+1} - [I]_{i_r} \|_2 \leq \epsilon_p\) and \(\|\begin{bmatrix} [\Psi]_{i_r}^{k+1} - \begin{bmatrix} [\Psi]_{i_r}^{k+1} \end{bmatrix}_{i_r} \|_2 \leq \epsilon_u\) 
   Apply control action \([u_0]_i = [\Phi_{u,0}]_i, [x_0]_r\), and return to step 1.
   else:
   Set \(k \leftarrow k + 1\), update \(\rho\) according to (10), and return to step 3.

**C. Convergence and Complexity Analysis**

1. **Complexity Analysis:** We focus on the computational overhead since communication (as described by steps 2, 4 and 6, of Algorithm 1) is limited to information exchange.
between $d$-local neighbors. Steps 3, 5 and 7 dominate the computational complexity of the algorithm. The optimization problems in step 5 has a closed form solution (c.f. Lemma 1), and step 7 requires only simple matrix operations. As a result, the computational overhead is significantly reduced. In particular, their evaluation reduces to addition and multiplication of matrices of dimension $O(d^2T^2)$, independent of the global size of the network $N$. Step 3 has to be computed by means of an optimization solver, since no closed-form solution exists and finding an explicit solution is no longer efficient (in contrast to the nominal case [14]) given the structure of the robust constraints. However, the size of the problem remains much smaller than the size of the network, specially when $d \ll N$. Each local row-wise subproblem optimizes over $O(d^2T^2)$ optimization variables subject to $O(dT)$ constraints. Note that in the nominal case [12], complexity of the DLMPC algorithm scales with order $O(d^2T)$. The additional $T$ factor is due to the fact that in the noise-free setting we only work with the first block-column of the system responses, whereas in the robust case all the matrices composing the system responses become relevant. As in the nominal case, both the local radius $d$ and the time horizon $T$ are independent design parameters.

2) Convergence Analysis: Convergence of Algorithm [1] follows directly from convergence results from ADMM [20] which require Assumption [2] and strong duality, which follows directly since all constraint sets are polytopes.

3) Recursive Feasibility and Stability: As is standard, recursive feasibility and stability can be guaranteed by setting a robust invariant set as the terminal set and a global Lyapunov function as the terminal cost [21], [22]. We leave finding methods for efficiently computing decoupled robust invariant sets and terminal costs satisfying Assumption [1] by exploiting locality to future work.

V. SIMULATION EXPERIMENTS

We evaluate performance and computational overhead of Algorithm [1]. We choose a chain system composed of $N$ subsystems where each has one scalar state. The local dynamics are given by:

$$[x(t+1)]_i = \alpha ([x(t)]_i + \sum_{j=i \pm 1} \kappa [x(t)]_j) + \beta_i [u(t)]_i + [w(t)]_i, \tag{15}$$

where $\alpha = 0.8$, $\kappa = 2$, $\beta_i = 1$ for $i = \{1, 3, 5, 6, 8, 10\}$ and $\beta_i = 0$ for $i = \{2, 4, 7, 9\}$ (notice the symmetry in the chain). We set random uniform noise to $\|\{w\}_i\| \le 1$ for all $i$. The system is subject to constraints $[x^{\text{min}}(t)]_i \le [x(t)]_i \le [x^{\text{max}}(t)]_i$ for all $t$. Simulations are performed with SLS toolbox [23] and code to reproduce the experiments is at [https://github.com/ unstable-zeros/dl-mpc-sls](https://github.com/unstable-zeros/dl-mpc-sls). Standard values for the ADMM parameters are $\tau = 1.5$, $\mu_1 = \mu_2 = 10$, $\rho_{\text{max}} = 5$. Some variation exists across different scenarios, values can be found in the code.

We analyze the trajectory obtained when computing the control input with Algorithm [1] versus when it is computed with the nominal DLMPC Algorithm in [12]. We choose $N = 10$, $d = 3$, $T = 5$, and simulate the system over a time horizon of 20 time steps. We set $[x^{\text{max}}(t)]_i = 1.5$ and $[x^{\text{min}}(t)]_i = -1.5$ for all nodes with actuation, i.e., $\beta_i = 1$, and $[x^{\text{max}}(t)]_i = 20$ and $[x^{\text{min}}(t)]_5 = -20$ for all other nodes. We run simulations for 5 different random noise realizations, and plot a sample resulting trajectory for subsystem 3 in Fig. 1. In the presence of disturbance, the state of the subsystem remains within the bounds for all times when the control input is computed via Algorithm [1] whereas it significantly violates the bounds when the control input is computed via the nominal scheme of DLMPC [12]. Moreover, when no disturbance is present both algorithms provide the same result, which illustrate that Algorithm [1] is a generalization of [12] in the case of affine constraints. The algorithm converges to the same value as CVX in all cases.

We also investigate the conservativeness of Algorithm [1] through a performance comparison with nominal DLMPC [12]. We present in Table I the different values of the average cost (and standard deviation) after running for the 5 different disturbance realizations. We observe that the cost relative difference between the two algorithms is around 30%. Generally, rDLMPC requires a more aggressive control action at the beginning, which leads to higher costs. But since in cases where the disturbance leads to constraint violations, the state in DLMPC reaches higher values and results in additional control actions to correct the deviations. The resulting performance of the rDLMPC algorithm is comparable to its nominal counterpart.
TABLE I
AVERAGE COST AND STANDARD DEVIATION OF DLMPC IN [15].

| Nominal MPC (Alg. I in [12]) | Nominal MPC (via CVX) | Robust MPC (Alg. I) | Robust MPC (via CVX) |
|-----------------------------|-----------------------|---------------------|----------------------|
| 1344 ± 160                 | 1343 ± 160            | 1804 ± 314          | 1807 ± 323           |

* denotes constraint violation

---

B. Computational overhead

We analyze Algorithm [1] and compare it with the overhead of its nominal counterpart Algorithm I in [12], both when run with the explicit solution [14] as well as when an optimization solver is needed. We perform simulations with varying parameters, and we analyze how runtime is affected. In particular, we choose a range of values for the size of the network $N$ as well as for the size of the locality region $d_l$. Simulation results are presented in Fig. 2. As predicted by the complexity analysis in Section IV, runtime is not dominated by the size of the network and it does increase with the size of the locality region. The slight increase observed with the size of the network stabilizes for large network sizes, and this is a feature of ADMM previously observed in [24]. The increase with the size of the network is more apparent than in the nominal case because in that case complexity was $O(d^2 T)$ while in the robust case we have $O(d^2 T^2)$ due to the need for additional optimization variables. Therefore any increase in $d$ will be magnified by an additional factor of $T$.

When comparing runtime values for the different schemes, the nominal approach is more efficient since runtime is faster by approximately two orders of magnitude. However, part of this efficiency is due to the use of an explicit solution, which as demonstrated in [14] provides a significant reduction in runtime. When using the Gurobi solver [25] for the nominal approach, runtimes are faster than the ones of rDLMPC by one order of magnitude. This indicates that the fact that the rDLMPC has significantly more variables than its nominal counterpart accounts for a 10-fold runtime increase, where the other 10-fold is due to the fact that no explicit solution is being used in the rDLMPC computation.

VI. CONCLUSION AND FUTURE DIRECTIONS

We provide a robust closed-loop MPC approach that is distributed and localized in both synthesis and implementation. Complexity of the algorithm is dominated by the size of the local region, and we corroborate through simulation that this approach is suitable for large-scale networks. To be best of our knowledge, this is the first DMPC algorithm that allows for the distributed synthesis of robust closed-loop policies. Future work will investigate exploiting locality for the scalable distributed computation of robust invariant sets and terminal cost functions.

REFERENCES

[1] D. Limon Marruedo, T. Alamo, and E. F. Camacho, “Stability analysis of systems with bounded additive uncertainties based on invariant sets: Stability and feasibility of MPC,” in Proc. IEEE ACC, vol. 1, 2002, pp. 364–369.
[2] G. Grimm et al., “Examples when nonlinear model predictive control is nonrobust,” Automatica, vol. 40, no. 10, pp. 1729–1738, 2004.
[3] W. Langson et al., “Robust model predictive control using tubes,” Automatica, vol. 40, no. 1, pp. 125–133, 2004.
[4] A. Richards and J. How, “Robust stable model predictive control with constraint tightening,” in Proc. IEEE ACC, 2006, p. 6.
[5] A. Richards and J. P. How, “Robust distributed model predictive control,” Int. J. Control, vol. 80, no. 9, pp. 1517–1531, 2007.
[6] C. Conte et al., “Robust distributed model predictive control of linear systems,” in Proc. ECC, 2013, pp. 2764–2769.
[7] V. Blondel and J. N. Tsitsiklis, “NP-hardness of some linear control design problems,” SIAM J. Control Optim., vol. 35, no. 6, pp. 2118–2127, 1997.
[8] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, “Optimization over state feedback policies for robust control with constraints,” Automatica, vol. 42, no. 4, pp. 523–533, Apr. 2006.
[9] L. Furieri and M. Kamgarpour, “Robust control of constrained systems given an information structure,” in Proc. IEEE CDC, Melbourne, Australia, 2017, pp. 3481–3486.
[10] J. Sieber, S. Bennani, and M. N. Zeilinger, “A system level approach to robust model predictive control,” 2021. [Online]. Available: https://arxiv.org/abs/2103.02460
[11] S. Chen et al., “Robust closed-loop model predictive control via system level synthesis,” in Proc. IEEE CDC, 2020, pp. 2152–2159.
[12] C. Amo Alonso and N. Matni, “Distributed and localized closed loop model predictive control via system level synthesis,” in Proc. IEEE CDC, 2020, pp. 5598–5605.
[13] Y. Chen and J. Anderson, “System level synthesis with state and input constraints,” in Proc. IEEE CDC, 2019, pp. 5258–5263.
[14] C. Amo Alonso, N. Matni, and J. Anderson, “Explicit distributed and localized model predictive control via system level synthesis,” in Proc. IEEE CDC, 2020, pp. 5606–5613.
[15] J. Anderson et al., “System level synthesis,” Annual Reviews in Control, vol. 59, no. 12, pp. 3238–3251, 2019.
[16] Y.-S. Wang, N. Matni, and J. C. Doyle, “A system level approach to controller synthesis,” IEEE Trans. Autom. Control, vol. 34, no. 8, pp. 982–987, 2019.
[17] ——, “Separable and localized system-level synthesis for large-scale systems,” IEEE Trans. Autom. Control, vol. 63, no. 12, pp. 4234–4249, 2018.
[18] ——, “Localized LQR optimal control,” in 53rd IEEE Conference on Decision and Control. IEEE, 2014, pp. 1661–1668.
[19] N. Matni, Y.-S. Wang, and J. Anderson, “Scalable system level synthesis for virtually localizable systems,” in Proc. IEEE CDC. IEEE, 2017, pp. 3473–3480.
[20] S. Boyd et al., “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” Found. Trends Mach. Learn., vol. 3, no. 1, pp. 1–122, 2010.
[21] F. Borrelli, A. Bemporad, and M. Morari, Model Predictive Control, Cambridge, UK New York, NY: Cambridge Univ. Press, 2016.
[22] J. Löfberg, “Minimix approaches to robust model predictive control,” 2003. [Online]. Available: http://www.control.isy.liu.se/research/reports/Ph.D.Thesis/PhD812.pdf
[23] J. S. Li, “SLS-MATLAB: Matlab toolbox for system level synthesis,” 2019. [Online]. Available: https://github.com/sls-caltech/sls-code
[24] C. Conte et al., “Computational aspects of distributed optimization in model predictive control,” in Proc. IEEE CDC, 2012, pp. 6819–6824.
[25] L. Gurobi Optimization, “Gurobi optimizer reference manual,” 2021. [Online]. Available: http://www.gurobi.com