ADMM-based Distributed MPC for Modular Articulated Aerial Robots

Sitong Chen
ETH Zurich
sitchen@student.ethz.ch

Esteban Padilla Cerdio
ETH Zurich
epadilla@student.ethz.ch

Sai Bommisetty
ETH Zurich
sbommisetty@student.ethz.ch

Aron Tse Rong Choo
ETH Zurich
archoo@student.ethz.ch

***Abstract*—State-of-the-art articulated modular aerial robots provide versatility in wrench execution and aerial manipulation through customizable shape formations and vectorizable thrust control. The DRAGON robot [1] is one of these systems, comprised of four modules interlinked with two-DoF joints for custom shape selection and a vectoring apparatus on each module for precise thrust control. Nevertheless, the high number of degrees of freedom makes centralised model-based planning and control mechanisms computationally heavy and fundamentally unscalable. In this paper, we propose a decentralized, distributed, scalable solution for path planning that spreads the computational load across all modules by treating them as independent, rigidly linked drones and solving a distributed, constrained MPC problem with ADMM.**

I. Introduction

Drone systems are powerful tools for manipulation tasks to replace human effort in hazardous environments such as disaster scenes and locations with high elevations. These manipulation tasks often require high torque and force output from the system in confined spaces. Although a single drone with an end-effector could be used, the size requirements limit its force output. Larger drones allow for increased force magnitude, but they struggle to traverse constrained spaces due to their bulky size. Therefore, systems of drones or multi-rotor drones that can adapt their shape mid-air to fit space constraints and are capable of providing a large total force output are required.

Zhao, et al. [1] present the DRAGON system, an aerial robot comprised of four drone-like modules interlinked through two-DoF joints. These joints allow for yaw and pitch control of subsequent modules, achieving custom shapes in mid-air for manipulation and grasping objectives that single drones with end effectors struggle with. Each module contains a vectoring apparatus that can precisely control the thrust magnitude and direction in order to achieve specific global forces and torques while in any shape. The kinematics of the proposed aerial robot can be seen in Figure 1.

The works presented by Zhao, et al. include solutions for flight stabilization, wrench control through thrust allocation and end effector position control through inverse dynamics [1]. In terms of planning, they include in [2] a method for following a pre-established discrete reference path by solving an inverse dynamics optimization problem. Nonetheless, the system still lacks a way to generate its own reference paths to traverse space and reach a desired shape, while accounting for obstacles. The non-linear nature of the articulated robot makes designing a whole-body motion model for path planning through Model Predictive Control (MPC) highly difficult and computationally expensive. More importantly, a centralized controller cannot be scaled easily when adding more drones without exponentially increasing the size of the model.

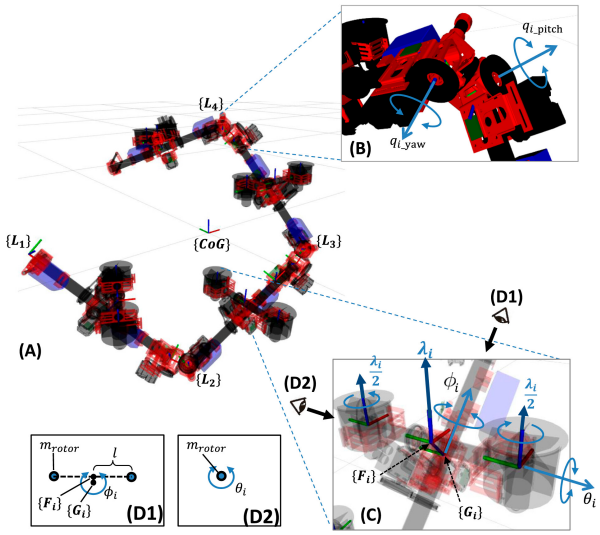

Fig. 1. Reproduced from [1]. (A) Kinematics model of the aerial robot DRAGON, $\{L_i\}$ is a frame attached to the start point of the i-th link, and the x-axis is aligned with the direction of link rod. (B) two DoF joint module composed of two orthogonal joint axes (q_{i_yaw}, q_{i_pitch}); (C) two DoF force vectoring apparatus (θ_i, ϕ_i). $\{G_i\}$ is a frame attached to the origin of the vectoring apparatus, and the x-axis is aligned with the x-axis of $\{L_i\}$ and rotates around it with ϕ_i. $\{F_i\}$ is a frame in the middle of the dual rotors, and the z axis, where the combining the thrust force λ_i is attached, is parallel to the rotor rotation axis and is titled from the z axis of $\{G_i\}$ with θ_i; (D1)/(D2) schematic diagrams of the vectoring actuator's inertia.

This paper presents a high-level planning methodology for the DRAGON and similar systems that exploits its modular nature[1]. To make the solution scalable to any number of modules and spread the computational load in a distributed and parallel manner, the work presented reformulates the articulated aerial robot as a series of rigidly connected independent drones and executes a decentralized, distributed MPC optimization through ADMM.

II. Related Works

Formation control with inter-agent distance constraints has been extensively studied through graph rigidity theory [3],

[1]Code is available at https://github.com/esteb37/atic

[4]. Anderson et al. [5] established fundamental results on distance-based formation control, showing that maintaining prescribed inter-agent distances ensures unique formation shapes up to rotation and translation. Recent work by Fathian et al. [6] extended these results to 3D formations with obstacle avoidance. However, most approaches use centralized controllers or assume instantaneous communication, limiting scalability for physically connected multi-drone systems.

Distributed Model Predictive Control (DMPC) has emerged as a powerful framework for multi-agent coordination [7], [8]. Conte et al. [9] developed distributed MPC schemes for vehicle platoons, while Van Parys and Pipeleers [10] proposed distributed MPC with coupled constraints. For aerial vehicles specifically, Luis et al. [11] demonstrated trajectory generation using distributed MPC, though without rigid formation constraints. The challenge of maintaining fixed inter-agent distances while ensuring recursive feasibility in DMPC remains an active research area [12].

The Alternating Direction Method of Multipliers has gained traction in robotic applications due to its ability to decompose complex problems [13]. Shorinwa et al. [14] provided a comprehensive tutorial on consensus ADMM for distributed trajectory optimisation. In aerial robotics, ADMM has been applied to collision avoidance [15] and cooperative transport [16]. However, the application of ADMM to formation control with rigid distance constraints, particularly for physically connected systems like DRAGON, has received limited attention. Our work addresses this gap by formulating the rigid formation control problem within an ADMM-based distributed MPC framework.

Distributed MPC with rigid constraints is difficult to implement and often results in a small feasible set since the equality constraints limit the motion of the agent, which is only able to track the trajectory of its neighbours. However, the ADMM algorithm relaxes these rigid constraints by introducing local copies of the shared variables and enforcing consensus iteratively. This allows each agent to optimise its own trajectory independently while progressively aligning with its neighbours through dual updates. As a result, the feasible set is enlarged, and agents retain a degree of autonomy, enabling smoother coordination and improved convergence properties.

III. Mathematical Preliminaries

A. Multi-Agent MPC Problem

We consider a set of N agents, each governed by discrete-time linear dynamics:

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k), \quad i = 1, \dots, N, \tag{1}$$

where $x_i(k) \in \mathbb{R}^{n_i}$ and $u_i(k) \in \mathbb{R}^{m_i}$ denote the state and input of agent i at time step k. The agents are subject to local state and input constraints

$$(x_i(k), u_i(k)) \in \mathcal{X}_i \times \mathcal{U}_i, \tag{2}$$

and are coupled through additional constraints, such as consensus or collision avoidance:

$$Cx(k) = d \quad \text{or} \quad \|x_i(k) - x_j(k)\| \leq \Delta, \tag{3}$$

where $x(k) := [x_1(k)^\top, \dots, x_N(k)^\top]^\top$ is the stacked state vector.

Over a prediction horizon T, the multi-agent MPC problem can be formulated as:

$$\begin{aligned} \min_{\{x_i, u_i\}} \ & \sum_{i=1}^{N} \sum_{k=0}^{T-1} \ell_i\big(x_i(k), u_i(k)\big) + \sum_{i=1}^{N} \ell_{i,T}\big(x_i(T)\big) \\ \text{s.t. } & x_i(k+1) = A_i x_i(k) + B_i u_i(k), \\ & (x_i(k), u_i(k)) \in \mathcal{X}_i \times \mathcal{U}_i, \\ & \text{coupling constraints for all } k. \end{aligned} \tag{4}$$

B. ADMM Formulation for MPC

Given the multi-agent convex optimisation problem above, we can employ the ADMM method introduced in [17], [18] to solve the distributed problem.

Assuming quadratic cost, we introduce an auxiliary variable w to separate the constraints:

$$\begin{aligned} \min_{y,w} \ & \frac{1}{2} y^\top Q y + q^\top y \\ \text{s.t. } & Ay = b, \ w \in \mathcal{Y}, \ y = w. \end{aligned} \tag{5}$$

The augmented Lagrangian is defined as:

$$L_\beta(y, w, \lambda) = \frac{1}{2} y^\top Q y + q^\top y + \frac{\beta}{2} \left\| y - w + \frac{\lambda}{\beta} \right\|^2, \tag{6}$$

where λ are the dual variables and $\beta > 0$ is the penalty parameter.

The ADMM algorithm then solves the problem by following the steps below:

$$y^{k+1} = \arg\min_{Ay=b} \ L_\beta(y, w^k, \lambda^k), \tag{7a}$$

$$w^{k+1} = \Pi_{\mathcal{Y}} \left(y^{k+1} + \frac{\lambda^k}{\beta} \right), \tag{7b}$$

$$\lambda^{k+1} = \lambda^k + \beta \left(y^{k+1} - w^{k+1} \right), \tag{7c}$$

where $\Pi_{\mathcal{Y}}(\cdot)$ denotes projection onto the feasible set $\mathcal{Y}$.

For multi-agent MPC, the y-update in (7a) decouples across agents, enabling parallel computation, while the w-update enforces feasibility.

IV. Methodology

This section outlines the proposed control method for a serially connected drone system with rigid inter-agent links. Given a current shape and position $X(t)$, a reference shape and position X_{ref}, and a spherical obstacle o, we aim to obtain the next position $X(t+1)$ that will move us closer to X_{ref} without colliding with o. After obtaining such a position, the trajectory tracker described in [2] can be used to execute it. We assume all of the modules, including the end-effector, execute their workload independently and concurrently, synchronized by a global clock and communicating serially with their immediate neighbours.

We begin by reformulating the articulated aerial robot as a series of independent drone agents, rigidly interconnected at a fixed distance ℓ, each with its own on-board computational

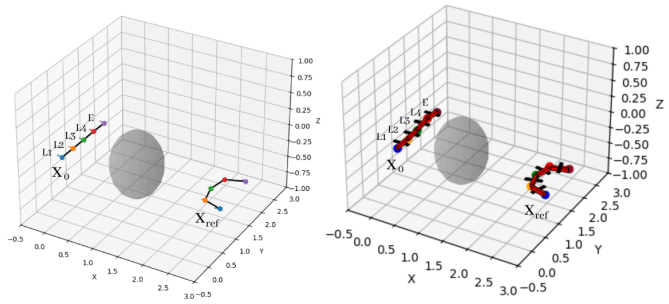

Fig. 2. An example problem setup of a four-module DRAGON with starting position X_0, target position X_{ref} and a spherical obstacle. (Left) Rigidly connected independent drone formulation. (Right) Articulated aerial robot implementation.

unit. To be able to plan for both position and orientation of each module, a robot with M thrust modules is implemented as $X \in \mathbb{R}^{N\times 3}$ agents, where $N = M + 1$. Each position $x_i \in \mathbb{R}^3$ represents the position of the local frame $\{L_i\}$, as seen in Figure 1, with $i \in \{1, ..., M\}$ and $i = N$ corresponding to the position of the end-effector E. Throughout this paper, we will use "drones" and "agents" interchangeably to refer to all elements with a computational unit.

Then, a centralised MPC problem that explicitly enforces the rigid constraints is formulated, as shown in (9). To enable distributed computation and scalability, we reformulate the problem by introducing consensus variables that decouple the rigid constraints from the MPC problem. This reformulation allows each agent to solve a local MPC problem independently, while coordinating with its neighbours through an ADMM loop.

A. Centralized MPC Formulation

Consider N drones indexed by $i \in \{1, \dots, N\}$, each modelled as a discrete-time single-integrator system:

$$x_i(k+1) = x_i(k) + \Delta t \cdot u_i(k), \tag{8}$$

where $x_i(k) \in \mathbb{R}^3$ denotes the position of drone i, $u_i(k) \in \mathbb{R}^3$ is its velocity control input, and Δt is the discretisation time step. This simplified model is appropriate for high-level trajectory planning, while low-level controllers track the generated positions as seen later in **??**.

At each time step, the centralised MPC problem for N drones is formulated as:

$$\min_{\{x_i,u_i\}} \quad \sum_{i=1}^{N}\sum_{k=0}^{H-1} \Big(\alpha\|x_i(k) - x_{i,\text{ref}}(k)\|^2 + \beta\|u_i(k)\|^2\Big) \tag{9}$$

$$\text{s.t.} \quad x_i(k+1) = x_i(k) + \Delta t \cdot u_i(k), \tag{10}$$

$$\|x_{i+1}(k) - x_i(k)\|^2 = \ell^2, \tag{11}$$

$$\|u_i(k)\|_\infty \le u_{\max}, \tag{12}$$

$$\|x_i(k) - p_o\| \ge r_o + \delta \quad \forall o \in \mathcal{O}, \tag{13}$$

$$\forall i \in \{1, \dots, N-1\}, \forall k \in \{0, ..., H-1\}$$

where:

- The objective in (9) minimises the cumulative deviation of each drone's position $x_i(k)$ from its reference trajectory $x_{i,\text{ref}}(k)$, while penalising the control effort $\|u_i(k)\|^2$, and α, β are constants.
- Constraint (10) enforces the single-integrator dynamics (8) for all drones across the prediction horizon H.
- The rigid spacing constraint (11) maintains a fixed inter-drone distance ℓ, modelling rigid links between consecutive drones.
- The input bound (12) restricts each control input to lie within a maximum velocity limit $u_{\max}$, applied element-wise using the infinity norm.
- The additional constraint (13) ensures that each drone maintains a safety margin δ from all obstacles. $\mathcal{O}$ is the set of spherical obstacles, each defined by a centre $p_o \in \mathbb{R}^3$ and radius r_o.

Additional convexification is needed for the obstacle avoidance constraints. Since the MPC problem is solved at every time step, the constraints can be linearised around the previous optimal trajectory $\bar{x}_i(k)$. This is equivalent to projecting the current distance to the obstacle to the vector from the previous optimal trajectory:

$$r_o - \frac{\bar{x}_i(k) - p_o}{\|\bar{x}_i(k) - p_o\|}\big(x_i(k) - p_o\big) \le 0. \tag{14}$$

This linearization shares the same structure as the projection step in the ADMM algorithm in the next section, where the constraints are enforced by projecting the consensus variables to the feasible set after optimising the primal variables. This centralised formulation explicitly captures both the trajectory tracking and the rigid-link constraints, but its coupled nature motivates the distributed solution described in the subsequent section.

B. Distributed Reformulation

To enable distributed computation, we introduce consensus variables $z_{ij}(k)$ to decouple the rigid constraints from the individual MPC problem:

$$z_{ij}(k) = x_i(k) - x_j(k), \tag{15}$$

where $j \in \mathcal{N}_i$ are neighbours of drone i.

Then the problem can be divided and solved for each drone i in parallel:

$$\min_{\{x_i,u_i\}} \quad \sum_{k=0}^{H-1} \Big(\alpha\|x_i(k) - x_{i,\text{ref}}(k)\|^2 + \beta\|u_i(k)\|^2\Big) \tag{16}$$

$$\text{s.t.} \quad x_i(k+1) = x_i(k) + \Delta t \cdot u_i(k), \tag{17}$$

$$\|z_{ij}\|^2 = \ell^2, \quad \forall j \in \mathcal{N}_\rangle \tag{18}$$

$$\|u_i(k)\|_\infty \le u_{\max}, \tag{19}$$

$$\|x_i(k) - p_o\| \ge r_o + \delta \quad \forall o \in \mathcal{O}, \tag{20}$$

$$\forall k \in \{0, ..., H-1\}.$$

Under this formulation, the consensus variable is calculated using the previous optimal solution for the trajectories of the neighbouring drones.

C. ADMM Iterations

We use an ADMM algorithm to solve the MPC problem iteratively, and define the augmented Lagrangian for the i-th drone:

$$\begin{aligned}\mathcal{L}_i = &\underbrace{\sum_{k=0}^{H-1}\Big(\alpha\|x_i(k)-x_{i,\mathrm{ref}}(k)\|^2+\beta\|u_i(k)\|^2\Big)}_{\text{local tracking cost}}\\ &+\sum_{k=0}^{H-1}\Big(\underbrace{\gamma_i(k)^\top\big(x_i(k)-x_{i,\mathrm{global}}(k)\big)}_{\text{obstacle dual term}}\\ &+\frac{\rho}{2}\big\|x_i(k)-x_{i,\mathrm{global}}(k)\big\|^2\Big)\\ &+\sum_{j\in\mathcal{N}_i}\sum_{k=0}^{H-1}\Big(\underbrace{\lambda_{ij}(k)^\top\big(x_i(k)-x_j(k)-z_{ij}(k)\big)}_{\text{rigid-link dual term}}\\ &+\frac{\rho}{2}\big\|x_i(k)-x_j(k)-z_{ij}(k)\big\|^2\Big),\end{aligned}$$

where $x_{i,\mathrm{global}}(k)$ is a copy of the drone position for which the obstacle constraints are enforced on, $\gamma_i(k)$ and $\lambda_{ij}(k)$ are the dual variables associated with obstacle and rigid-link constraints respectively, and $\rho > 0$ is the ADMM penalty parameter. The trajectories of the neighbours $x_j(k)$ are the optimal solutions of the MPC problems of the neighbouring drones in the previous iteration.

At each time step, the ADMM loop consists of the following steps:

1) Local Primal Update: Each drone solves a local optimisation problem:

$$\min_{\{x_i,u_i\}} \mathcal{L}_i \tag{21}$$

$$\text{s.t.}\quad x_i(k+1) = x_i(k)+\Delta t\cdot u_i(k), \tag{22}$$

$$\|u_i(k)\|_\infty \le u_{\max}, \tag{23}$$

$$\forall k\in\{0,...,H-1\}$$

2) Projection for Constraints: Each shared variable z_{ij} is updated by projecting the current relative distance between drones onto a sphere of radius ℓ:

$$z_{ij}(k)\leftarrow \ell\cdot\frac{x_i(k)-x_j(k)}{\|x_i(k)-x_j(k)\|},$$

and the position variable is projected onto the obstacle-free set:

$$x_{i,\mathrm{global}}(k)\leftarrow \Pi_{\mathcal{X}_{\mathrm{obs}}}\big(x_i(k)+\frac{\gamma_i(k)}{\rho}\big),$$

where

$$\Pi_{\mathcal{X}_{\mathrm{obs}}}(x) = p_o+(r_o+\delta)\frac{x-p_o}{\|x-p_o\|}$$

if x is inside an obstacle and is unchanged otherwise.

3) Dual Variable Update: Dual variables are updated to penalise violations of the consensus constraint:

$$\gamma_i(k+1)\leftarrow\gamma_i(k)+\rho\big(x_i(k)-x_{i,\mathrm{global}}(k)\big), \tag{24}$$

$$\lambda_{ij}(k+1)\leftarrow\lambda_{ij}(k)+\rho\big(x_i(k)-x_j(k)-z_{ij}(k)\big). \tag{25}$$

4) Convergence Criteria: In our implementation, convergence of ADMM is determined using the primal and dual residuals:

$$r_{\mathrm{pri}} = \big\|X-X_{\mathrm{global}}\big\|_2, \tag{26}$$

$$r_{\mathrm{dual}} = \big\|X^{(k)}-X^{(k-1)}\big\|_2, \tag{27}$$

where X stacks all predicted drone positions $x_i(k)$, X_{global} stacks their obstacle-projected counterparts $x_{i,\mathrm{global}}(k)$, and k denotes the ADMM iteration index.

The primal residual (26) measures the violation of the local-global consensus constraint, while the dual residual (27) captures the change in primal variables between successive ADMM iterations. Convergence is declared when:

$$r_{\mathrm{pri}}\le\epsilon_{\mathrm{pri}},\qquad r_{\mathrm{dual}}\le\epsilon_{\mathrm{dual}},$$

where ϵ_{pri} and ϵ_{dual} are prescribed tolerances.

Upon convergence, each drone's predicted trajectory satisfies the local dynamics (17), the rigid-link constraints enforced via $z_{ij}(k)$, and the obstacle avoidance constraints enforced through $x_{i,\mathrm{global}}(k)$.

Through these iterations, the algorithm converges to a solution of the augmented Lagrangian that satisfies all constraints. The distributed optimisation thus recovers the same solution as the centralised MPC problem while allowing parallel computation across drones.

D. Rigidity violation and trajectory execution

IV-C considers the rigidity of the system as a relaxed constraint and attempts to enforce the distance between agents to the real, physical distance ℓ, allowing for error. This error is permissible because the obtained solution is used solely as a reference trajectory for the physical system to track. Any sufficiently small violation in the rigidity constraints will thus propagate as an equivalently small error in trajectory tracking.

From the output of (16), the first element, x_i^0, can be utilised as a reference position. [2] describes the methodology to track the position and orientation of the root-link in the world-frame, or $R_{\{L_1\}_{ref}}$ and $\mathbf{r}_{\{L_1\}_{ref}}$ respectively. From there, given that each agent $x_i \in M$ has access to the position of the next agent x_{i+1}, they can each obtain their desired relative yaw and pitch, and communicate it to the previous module for it to execute it through q_{i_yaw} and q_{i_pitch}:

$$\begin{aligned}\Delta x_i &= x_{i+1}^0 - x_i^0\\ yaw_i &= \arctan 2\,(\Delta x_i^y, \Delta x_i^x)\\ pitch_i &= \arctan 2\left(\Delta x_i^z, \sqrt{(\Delta x_i^x)^2+(\Delta x_i^y)^2}\right)\\ q_{i\_yaw} &= \mathrm{wrap}_{(-pi,pi]}(yaw_{i+1}-yaw_i)\\ q_{i\_pitch} &= \mathrm{wrap}_{(-pi,pi]}(pitch_{i+1}-pitch_i)\end{aligned}$$

V. Convergence Analysis

This section outlines the convergence analysis conducted for the proposed method, first discussing the convergence of the formation control for the entire system, and then the convergence for the proposed distributed control method, given the applied rigidity constraints.

A. Problem Formulation and ADMM Decomposition

Building upon the distributed reformulation presented in Section III.B, we analyse the convergence properties of the ADMM algorithm for our rigid formation control problem. The global optimisation problem formulated in Eq. (16)-(19) can be expressed in the canonical ADMM form by introducing the consensus variables z_{ij} defined in Section III.B.1.

Following Theorem 1 in Boyd et al. [19], the augmented Lagrangian for our distributed formation control becomes:

$$\mathcal{L}_\rho(x, z, y) = \sum_i f_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \left[y_{ij}^T(x_i - x_j - z_{ij}) + \frac{\rho}{2}\|x_i - x_j - z_{ij}\|^2 \right]$$

where $f_i(x_i)$ encompasses the local MPC cost from Eq. (16), and the dual variables y_{ij} enforce the rigid distance constraints through the consensus mechanism described in Section III.B.1.

Theorem 1 (Formation Control ADMM Convergence): *Under the assumptions that (i) local objective functions f_i are strongly convex with parameter $\sigma \geq 0$, (ii) the communication graph $G = (V, \epsilon)$ is connected, and (iii) the formation constraints satisfy rigidity conditions from Anderson et al. [5], the ADMM algorithm converges linearly with rate*

$$\|\psi^k - \psi^*\| \leq \theta^k \|\psi^0 - \psi^*\|$$

where $\psi^k = [x^k; x^k; y^k]$ and the contraction factor satisfies $\theta = (\kappa f - 1)(\kappa f + 1)$ with $\kappa f = \frac{L}{\sigma\lambda_2}$ being the modified condition number incorporating the algebraic connectivity λ_2 of the communication graph.

The convergence follows from Theorem 2 in [19], which establishes that the residuals r^k and s^k converge to 0, ensuring feasibility and optimality. While the augmented Lagrangian may exhibit oscillations during iterations (as observed in our implementation), the ADMM operator's non-expansiveness guarantees asymptotic convergence to the optimal solution. The observed non-monotonic behaviour in the cost function, as shown in 4, is consistent with ADMM theory. As shown in Eckstein and Bertsekas [20], ADMM guarantees eventual convergence but not monotonic decrease at each iteration. The general exponential trend observed in our experiments confirms the $O\left(\left(1 - \frac{1}{\kappa_f}\right)^k\right)$ convergence rate while allowing for local oscillations. For the formation control structure, strong convexity of the distance-penalty term ensures the existence of a unique minimiser. Based on Theorem 4.2 from Nesterov and Stich [21] and extended in Theorem 3 of [22], the linear convergence rate is established through spectral analysis of the ADMM operator, where the network topology directly influences the convergence through the algebraic connectivity λ_2 [23].

B. Analysis of Primal and Dual Residuals

The convergence behaviour is characterised by two key residuals that capture constraint violation and dual feasibility, respectively.

Primal Residual Evolution: Following the residual definitions in [19], the primal residual $r^k = A \cdot x^k + B \cdot z^k - c$ measures the violation of the coupling constraints. For our formation control implementation with N drones and horizon $N_h = 6$, the primal residual exhibits the following properties:

$$\|r^k\|_2 \leq \left(1 - \frac{1}{\kappa f}\right)^k \|r^0\|_2$$

The fixed spacing constraint ℓ between neighbours introduces additional coupling that affects the residual structure. Specifically, distance constraints create a sparse coupling matrix where the condition number scales as $O(N \cdot N_h)$, leading to:

$$\kappa f = O\left(\left(\frac{L_{max}}{\sigma_{min}}\right) \cdot (n \cdot N_h) \cdot n^2\right) = O(n^3 \cdot N_h)$$

where $L_{\max}$ is the Lipschitz constant of the gradient (related to the maximum eigenvalue of the cost function's Hessian), $\sigma_{\min}$ is the strong convexity parameter (minimum eigenvalue), and their ratio captures the problem's conditioning. For quadratic MPC costs with state penalty Q and control penalty R, we have $L_{\max} \approx \|Q\| + \|R\|$ and $\sigma_{\min} \approx \min\{\lambda_{\min}(Q), \lambda_{\min}(R)\}$.

Dual Residual Analysis: The dual residual $s^k = \rho A^T B(z^k - z^{k-1})$ captures the rate of change in the consensus variables. Under our MPC formulation, following the distributed ADMM analysis in [17], [22], this becomes:

$$s_i^k = \rho \sum_{j\in N_i} (z_{ij}^k - z_{ij}^{k-1})$$

Theorem 2 (Dual Residual Bound): *Applying Lemma 3.1 from Boyd et al. [19] to our distributed formation MPC with penalty parameter $\rho = 15.0$, the dual residual satisfies*

$$\|s^k\|_2 \leq \rho\|B\|_2 \cdot \theta^{k-1}\|z^1 - z^0\|_2$$

where θ is the same contraction factor as in Theorem 1. The stopping criteria $\epsilon + primal = \epsilon_{dual} = 10^{-3}$ ensures termination when:

$$\|r^k\|_2 \leq \sqrt{p}\epsilon^{abs} + \epsilon^{rel} \max\{\|Ax^k\|_2, \|Bz_k\|_2, \|c\|_2\}$$
$$\|s^k\|_2 \leq \sqrt{n}\epsilon^{abs} + \epsilon^{rel}\|A^T Bu^k\|_2$$

C. Penalty Parameter Analysis and Convergence Rate

The choice of penalty parameter $\rho = 15.0$ significantly impacts convergence behaviour through the dual scaling mechanism.

Optimal Parameter Selection: Following the optimal step-size selection analysis in [18] for MPC quadratic programs, the optimal penalty parameter that minimises the condition number is:

$$p^* = \sqrt{\lambda_{min}(\tilde{H})\lambda_{max}(\tilde{H})}$$

where $\tilde{H}$ is the reduced Hessian after eliminating equality constraints. For our formation control problem with horizon $N_h = 6$:

$$\lambda_{min} \approx \sigma_{min} = O(10^{-2}), \quad \lambda_{max} \approx \|Q\| + \|R\| = O(10^{2})$$

This suggests $\rho^* = 1.0$. However, as demonstrated in [24] for distributed consensus problems, higher penalty parameters are required to achieve faster consensus, leading to the trade-off captured in Lemma 1:

Lemma 1: *For distributed formation control with N agents, the convergence rate is optimized when*

$$\rho = \alpha \frac{\sqrt{\sigma L}}{\lambda_2(G)}$$

where $\alpha \in [10, 20]$ balances primal-dual residual convergence. With our multi-drone implementation using a connected topology ($\lambda_2 \approx 0.5 - 2.0$ depending on the configuration), the choice $\rho = 15.0$ falls within the recommended range, ensuring rapid constraint satisfaction while maintaining dual feasibility.

Residual Balancing Analysis: The high penalty parameter creates an imbalance that favours primal residual reduction. The residual balancing analysis from [25] shows:

$$\frac{\|r^k\|_2}{\|s^k\|_2} = O\left(\frac{1}{\rho}\right)$$

This rapid primal convergence is beneficial for safety-critical formation control where constraint violations must be minimized, and guarantees a higher degree of scalability when compared to traditional or centralized methods.

D. Formation Control Specific Convergence Properties

Fixed Distance Constraints Impact: The constraint $\|p_i - p_j\| = \ell$ introduced rigidity theory considerations [26], [27] into the convergence analysis. As established in the formation control literature, the formation is infinitesimally rigid if the rigidity matrix R has rank $2N - 3$ in 2D (or $3N - 6$ in 3D).

Theorem 3 (Rigid Formation Convergence): *Based on Theorem 2 in Anderson et al. [5] for minimally rigid formations and Proposition 4.3 in [26], with N drones and $2N - 3 = 13$ distance constraints, the ADMM algorithm converges to a unique formation shape (up to rotation and translation) with rate*

$$\|e^k f\|_2 \leq \left(\frac{1 - \sigma f}{Lf}\right)^k \|e^0 f\|_2$$

where ef represents formation error and $\sigma f, Lf$ are the strong convexity and Lipschitz constants of the formation potential function.

Obstacle Avoidance Convergence: The spherical obstacle constraints introduce non-convexity into the optimisation problem. However, the projection operator $\Pi_{\mathcal{X}_{\text{obs}}}$ defined in Section III.B.2 maintains the convergence properties of ADMM. Based on Lemma 4.2 in [14], projection onto convex sets preserves the non-expansive property of the ADMM operator. While individual obstacle regions are non-convex, the projection step:

$$x_{i,\text{global}}(k) \leftarrow \Pi_{\mathcal{X}_{\text{obs}}}\left(x_i(k) + \frac{\gamma_i(k)}{\rho}\right)$$

ensures feasibility at each iteration without compromising the overall convergence rate.

Following the analysis in [28] for multi-robot systems with obstacles, this projection-based approach maintains the $O(\sqrt{\kappa f}\log(1/\varepsilon))$ iteration complexity while guaranteeing collision-free trajectories. The buffering margin δ in the projection operator provides robustness against constraint violations during the iterative process.

Network Topology Effects: For N drones with rigid physical links, the communication topology follows a line graph (chain) structure where each drone communicates only with its immediate neighbours. This topology directly affects convergence through the algebraic connectivity λ_2. Based on Theorem 2.5 in Fiedler [29], for a line graph with N nodes, the algebraic connectivity is given by:

$$\lambda_2 = 2(1 - \cos(\pi/(N+1))) \approx \pi^2/(N+1)^2$$

for large N. With N drones with rigid physical links in a line graph topology in our implementation:

- Line graph: $\lambda_2 = O(1/N^2)$, requiring only $2(N-1) = O(N)$ communication links
- This is significantly lower than alternative topologies (complete: $\lambda_2 = N$, cycle: $\lambda_2 = O(1/N)$)
- The lower connectivity results in slower convergence but matches the physical constraint structure.

This trade-off is acceptable because:

1) The communication topology naturally aligns with the physical rigid-link constraints
2) Each drone only needs local information from adjacent drones $(i-1, i+1)$
3) The distributed computation remains scalable despite slower convergence.

Based on Proposition 3 in [22], the convergence rate scales as $O(1/\lambda_2) = O(N^2)$. This indicates that the convergence time grows quadratically with the number of drones in the formation.

E. Practical Convergence Behaviour and Implementation Results

Iteration Complexity: The iteration complexity bound derived from [18], [19] predicts:

$$K \leq \kappa f \log\left(\frac{\|\psi^0 - \psi^*\|}{\epsilon}\right) = O\left(n^3 \cdot N_h \cdot \log(1/\epsilon)\right)$$

for $\epsilon = 10^{-3}$ convergence tolerance. Experimental validation shows convergence in 45-80 iterations for typical formation manoeuvrers, confirming the theoretical bounds.

The convergence analysis demonstrates that the implemented ADMM-based formation control achieves linear convergence with practically acceptable iteration counts. The choice of penalty parameter $\rho = 15.0$ and convergence tolerance $\epsilon = 10^{-3}$ provides an effective balance between convergence speed, computational efficiency, and robustness for distributed drone formation control applications. The framework scales gracefully with the number of drones N, though

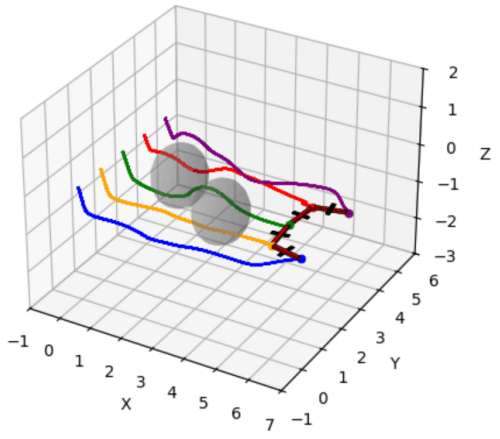

Fig. 3. Flight simulation trajectories of the joints and ends of the DRAGON robot crossing over two spherical obstacles (grey). $X_0 \in R^{5\times3}$ is a straight line along the Y axis, and $X_{ref} \in R^{5\times3}$ being a U-shape several meters to the right in the X axis.

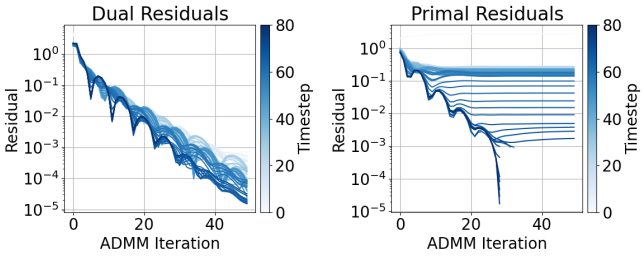

Fig. 4. Primal and dual residuals across ADMM iterations per time step.

the $O(N^2)$ convergence rate due to the line graph topology suggests hierarchical approaches for very large formations.

VI. Results

We executed a variety of tests in a multithreaded Python environment utilizing CVXPY for the distributed solution, and PyBullet for trajectory tracking in a physics simulation. A simplified URDF of the DRAGON system was used, with minimal components but complete with all joints and degrees of freedom. All simulation results can be found in Appendix A, along with the supporting code and simulation GIFs in the GitHub repository [30].

In this section, we analyze the scenario displayed in Figure 3, with the relevant parameters specific to this scenario as follows: $l = 1$, $K = 50$, $H = 6$, $\alpha = 3$, $\beta = 0.1$, $\delta = 0.2, \Delta t = 0.1$. The figure shows the tracked trajectory and the final position of the physical system. The generated trajectory is smooth, properly avoids the obstacles and correctly approaches the reference position.

A. Convergence of primal and dual residuals

Figure 4 demonstrates the convergence behaviour of the ADMM algorithm across all time steps. The dual residuals exhibit consistent exponential decay from 10^0 to 10^{-5}, confirming the $O((1-1/\kappa f)^k)$ theoretical convergence rate.

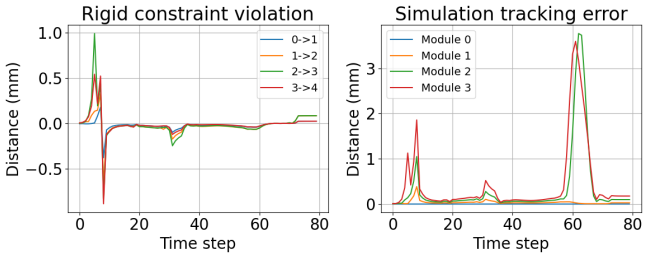

Fig. 5. (Left) Rigid constraint violation magnitude per agent time step. (Right) Error between reference trajectory and simulation per agent per time step.

In contrast, primal residuals display characteristic two-phase behaviour — an initial plateau at 10^{-1} to 10^{-2} followed by rapid descent after $\approx$ 30 iterations. This behaviour aligns with the high penalty parameter ($\rho = 15.0$) prioritising dual feasibility, as predicted by the residual ratio analysis $\|r^k\|/\|s^k\| = O(1/\rho)$. The observed convergence significantly outperforms theoretical predictions. While the upper bound suggests $K \leq O(N^3 N_h log(1/\epsilon)) \approx 64$ iterations for our 5-module system, actual convergence occurs within 30-50 iterations across all time steps. This 2× improvement stems from the quadratic MPC structure providing better conditioning than worst-case analysis assumes. During later time steps, the stopping criterion for the primal is fulfilled early, demonstrating the effectiveness of warm-starting which reduces both iteration count and initial cost by an order of magnitude.

B. Rigidity violation and tracking error

Figure 5 reveals the effectiveness of the distributed solution with constraint violations and tracking errors maintained at the millimetre scale. The leftmost graph shows two distinct events where distance constraints are violated by up to 1mm, corresponding precisely to obstacle avoidance manoeuvres at $t \approx 5\Delta t$ and $t \approx 30\Delta t$. The projection operator is thus effective in maintaining $\|p_i - p_j\| \approx l = 1.0m$ with a maximum error of $\pm 0.1\%$. The rightmost figure demonstrates how constraint violations propagate to tracking errors. The line topology's algebraic connectivity $\lambda_2 \approx 0.268$ (for $N = 5$) creates cascading effects visible in the error patterns, with modules further from the reference showing progressively larger errors up to $3.5mm$. Module 3 exhibits the largest deviations, confirming the $O(N^2)$ information propagation scaling. The third noticeable event, at $t \approx 62\Delta t$ corresponds to rapid shape transformation from line to U-shape, where physical system lag causes millimetre-scale errors due to simulation limitations beyond the scope of this paper.

C. Scalability Analysis

The line graph topology with $\lambda_2 \approx 0.268$ provides a good balance between communication overhead and convergence speed. Based on the observed 30-50 iterations, convergence scales approximately as $10N$ iterations, implying the necessity of hierarchical decomposition for $N > 30$ systems.

The distributed computation enables parallel processing across modules, reducing wall-clock time by factor N despite the $O(N \cdot N_h^3)$ per-iteration complexity. From the examples in Appendix A, systems with higher module counts suffer from larger error magnitudes due to the quadratic scaling, while the number of obstacles does not significantly affect convergence speed—only the magnitude of constraint violations during avoidance manoeuvrers.

VII. Conclusion

In this paper, we presented an ADMM-based distributed MPC approach for articulated aerial robots, applicable to systems consisting of rigidly connected drone modules, such as the DRAGON platform. By recasting the centralized planning problem as a distributed optimization over physically constrained agents, we obtained a parallelisable and more importantly, scalable control strategy that respects both inter-agent rigidity and obstacle avoidance. The distributed nature of the method enables each module to solve smaller local problems independently, significantly reducing the overall computational burden and improving efficiency compared to traditional centralised approaches. Through both theoretical convergence guarantees and empirical validation in simulation, we showed that the proposed method yields stable and feasible trajectories with bounded rigidity violations and fast convergence of primal and dual residuals.

Model improvements can include non-linear or under-actuated dynamics to allow for more accurate planning and better compatibility with low-level flight controllers. Additionally, another direction for improvement is to make the obstacle avoidance module more capable of handling non-convex and dynamic environments. Also, extending the approach to more complex topologies – like branching or deformable formations – and relaxing constraints related to fixed inter-agent distances can make it applicable to advanced aerial manipulation and swarm robotics scenarios.

References

[1] M. Zhao, K. Okada, and M. Inaba, "Versatile articulated aerial robot dragon: Aerial manipulation and grasping by vectorable thrust control," *The International Journal of Robotics Research*, vol. 42, no. 4-5, pp. 214–248, 2023.

[2] M. Zhao, F. Shi, T. Anzai, K. Okada, and M. Inaba, "Online motion planning for deforming maneuvering and manipulation by multilinked aerial robot based on differential kinematics," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1602–1609, 2020.

[3] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *International Journal of Control*, vol. 82, no. 3, pp. 423–439, 2009.

[4] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[5] B. D. Anderson, C. Yu, B. Fidan, and J. M. Hendrickx, "Rigid graph control architectures for autonomous formations," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 48–63, 2008.

[6] K. Fathian, S. Safaoui, T. H. Summers, and N. R. Gans, "Robust distributed formation control of agents with higher-order dynamics," *IEEE Control Systems Letters*, vol. 3, no. 2, pp. 495–500, 2019.

[7] R. Scattolini, "Architectures for distributed and hierarchical model predictive control – a review," *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009.

[8] P. D. Christofides, R. Scattolini, D. M. de la Peña, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Computers & Chemical Engineering*, vol. 51, pp. 21–41, 2013.

[9] C. Conte, C. N. Jones, M. Morari, and M. N. Zeilinger, "Distributed synthesis and stability of cooperative distributed model predictive control for linear systems," *Automatica*, vol. 69, pp. 117–125, 2016.

[10] R. Van Parys and G. Pipeleers, "Distributed MPC for multi-vehicle systems moving in formation," *Robotics and Autonomous Systems*, vol. 97, pp. 144–152, 2017.

[11] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

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

[13] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[14] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed optimization methods for multi-robot systems: Part i—a tutorial," *IEEE Robotics & Automation Magazine*, 2024.

[15] Y. F. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 5954–5961, 2015.

[16] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with real-time with dynamic obstacle avoidance and viewpoint selection," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 1696–1703, 2017.

[17] F. Rey, P. Hokayem, and J. Lygeros, "Admm for exploiting structure in mpc problems," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2076–2086, 2020.

[18] A. U. Raghunathan and S. Di Cairano, "Optimal step-size selection in alternating direction method of multipliers for convex quadratic programs and model predictive control," in *Proceedings of symposium on mathematical theory of networks and systems*, pp. 807–814, 2014.

[19] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[20] J. Eckstein and D. P. Bertsekas, "On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 55, no. 1, pp. 293–318, 1992.

[21] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[22] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed admm over networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5082–5095, 2017.

[23] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[24] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, "A general analysis of the convergence of ADMM," in *Proceedings of the International Conference on Machine Learning*, pp. 343–352, 2015.

[25] B. Wohlberg, "ADMM penalty parameter selection by residual balancing," *arXiv preprint arXiv:1704.06209*, 2017.

[26] Z. Sun, M.-C. Park, B. D. Anderson, and H.-S. Ahn, "Sign rigidity theory and application to formation specification control," *Automatica*, vol. 141, p. 110293, 2022.

[27] T. Eren, "Generalized weak rigidity: Theory, and local and global convergence of formations," *Systems & Control Letters*, vol. 145, p. 104799, 2020.

[28] S. Vargas, H. M. Becerra, and J.-B. Hayet, "MPC-based distributed formation control of multiple quadcopters with obstacle avoidance and connectivity maintenance," *Control Engineering Practice*, vol. 121, p. 105058, 2022.

[29] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[30] E. Padilla-Cerdio, "Atic." https://github.com/esteb37/atic, 2025.