Riemens *et al.*
Journal on Advances in Signal Processing (2026) 2026:38
https://doi.org/10.1186/s13634-026-01306-2

Journal on Advances in Signal Processing

RESEARCH Open Access

Cooperative Gaussian process-based model predictive control for safe multi-agent navigation

Ellen H. J. Riemens[1*], Alle-Jan van der Veen[1] and Raj T. Rajan[1]

*Correspondence:
E.H.J.Riemens@tudelft.nl

[1] TU Delft, Fac. EEMCS, Mekelweg 4, Delft 2628CD, The Netherlands

Abstract

Multi-agent systems, such as fleets of robots or drones, are increasingly deployed in logistics, inspection, and surveillance. These systems must reach their targets while maintaining safe separation, even under uncertain dynamics. This is challenging because unmodeled effects, disturbances, and sensor noise can degrade tracking performance and compromise safety. Model Predictive Control (MPC) is well suited for multi-agent navigation since it optimizes trajectories over a prediction horizon while enforcing input and state constraints. However, its performance depends on accurate models, and centralized formulations suffer from poor scalability and a single point of failure. We propose a cooperative Gaussian Process–augmented MPC (GP-MPC) framework that combines learning, chance-constrained safety, and distributed optimization. Each agent uses a Gaussian Process to learn its residual dynamics and quantify local uncertainty, incorporates this uncertainty into a chance-constrained collision-avoidance scheme, and coordinates only with neighbors through an ADMM-based distributed optimization method. This integration provides robustness to model errors and scalability to larger teams. The framework enables collision avoidance using only local uncertainty estimates, removing the need to share covariance information. By restricting computation and communication to each agent's neighborhood, it maintains scalability and efficiency. Simulations show that the approach yields smoother and more efficient trajectories, faster convergence to targets, and reliable probabilistic safety compared to nominal and nonlinear MPC baselines. Convergence analysis further confirms robust consensus across a range of tuning parameters.

1 Introduction

Multi-agent systems, such as teams of drones or mobile robots, need to move through shared spaces without colliding while still reaching their goals. This requires both accurate target reaching and reliable collision avoidance. In practice, this is challenging because models of the agents are never exactly known, and disturbances can lead to unsafe behavior if they are not handled properly.

Multi-agent systems can be controlled with methods ranging from Lyapunov-based schemes to reinforcement learning, but these approaches often struggle to combine constraint handling with forward-looking planning. Model Predictive Control (MPC)

Springer

is well suited to multi-agent navigation because it optimizes control inputs over a prediction horizon while respecting state and input constraints [1]. This allows agents to anticipate possible collisions and adjust their trajectories before they occur. MPC has been widely applied to multi-agent systems thanks to its ability to handle constraints explicitly and maintain safe trajectories [2, 3]. However, its performance depends on having an accurate model of the agent's dynamics, and it can degrade when unmodeled effects or disturbances are present.

Nonlinear dynamics can be represented through a variety of methods, including parametric models, differential equations, and learning-based approaches, but many of these lack systematic ways to quantify uncertainty. Gaussian Processes (GPs) provide a flexible way to model nonlinear dynamics while also estimating uncertainty in their predictions. This makes them useful for control, as the uncertainty can be incorporated into decision-making to improve safety. GP-based models have been applied in control settings ranging from system identification to reinforcement learning [4, 5], and they have been combined with MPC to improve performance under uncertainty [6, 7]. However, most existing GP-MPC approaches are centralized, requiring global data fusion or the exchange of covariance information, which limits scalability and creates single points of failure.

Several approaches have been developed to improve multi-agent MPC, but each has limitations. Deterministic distributed MPC methods scale well by convexifying collision-avoidance constraints, but they do not account for model uncertainty [8, 9]. More advanced strategies approximate nonlinear collision boundaries, use event-triggered replanning, or partition the workspace with buffered Voronoi cells, while others embed artificial potential fields or mixed-integer formulations to capture non-convex constraints [10–15]. These methods improve feasibility and scalability but remain conservative or computationally demanding. Chance-constrained MPC introduces probabilistic safety, yet typically requires sharing full covariance information between agents, creating high communication loads [16, 17]. Centralized GP-MPC methods combine learning with uncertainty handling, but rely on global data fusion or coordination [6, 18–23]. In short, no existing method provides a fully distributed GP-MPC that incorporates model uncertainty while keeping communication between agents minimal.

In this paper, we present a distributed GP-MPC framework that combines learning and coordination. We consider the dynamics of each agent to be a combination of a known nominal dynamics, and an unknown residual dynamics which include the systemic errors. This residual dynamics is then learned using a Gaussian process, the moments of which are used to improve the overall local dynamics of each agent and collision avoidance with neighboring agents. In particular, the uncertainty allows us to add chance-constrained collision-avoidance conditions, so that agents can keep safe distances even when the model is imperfect. To coordinate with others, agents use an ADMM-based distributed scheme that only requires exchanging trajectory information with neighbors, avoiding covariance sharing and central control.

The contributions of this paper are as follows:

- We propose a novel cooperative GP-MPC framework that unifies GP-based residual learning, uncertainty-aware chance-constrained safety, and an ADMM-based multi-agent coordination.
- We formulate the overall scheme in a distributed ADMM-based framework, enabling scalable multi-agent coordination with minimal communication between neighbors.

Together, these elements yield a framework that is optimal through the MPC formulation, robust through GP-based uncertainty handling, and scalable through distributed coordination.

2 Background and system model

Consider a 2D or 3D environment, and let D_x denote the number of dimensions. In this shared environment, we have a set of K agents, where each agent k, at each discrete time step t, aims to follow a predefined reference trajectory with positions given by $\mathbf{x}_k^{t,\text{ref}} \in \mathbb{R}^{D_x}$, while avoiding collisions with other agents in the environment. We assume that each agent has a partially known noisy dynamical model

$$\mathbf{x}_k^{t+1} = \mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t) + \mathbf{g}_k(\mathbf{x}_k^t) + \mathbf{w}_k^t, \quad k \leq K, \tag{1}$$

where $\mathbf{x}_k^t \in \mathbb{R}^{D_x}$ is the state vector of the kth agent at discrete time instance $t \in \mathbb{Z}$ and $\mathbf{u}_k^t \in \mathbb{R}^{D_u}$ is its control input. The known dynamics are given by the function $\mathbf{f}_k(\cdot)$ which is referred to as nominal dynamics, while $\mathbf{g}_k(\cdot)$ is an unknown nonlinear function and $\mathbf{w}_k^t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{w,k})$ is the additive noise reflecting systemic errors, which are assumed to be drawn from a zero-mean Gaussian distribution with linearly independent components.

We assume that the agents are capable of bidirectional communication with other agents within a certain known range. This communication network is represented by the graph $\mathcal{G}(\mathcal{K}, \mathcal{E})$, where $\mathcal{K} = \{1, \ldots, K\}$ denotes the vertices and $\mathcal{E}$ denotes the set of edges between the agents. There is an edge between an agent pair $(k, j) \in (\mathcal{K} \times \mathcal{K})$, if and only if their pairwise distance is less than the predefined communication radius. This communication graph additionally allows for cooperative distributed computation.

2.1 Gaussian process regression (GPR)

Recall from (1) that the dynamics of each agent are partially known. The unknown dynamics consist of both $\mathbf{g}_k(\mathbf{x}_k^t)$ and $\mathbf{w}_k^t$ and must be learned to enable efficient and optimal control of the agent. In this work, we propose to use Gaussian process regression (GPR) [24] to learn these nonlinear dynamics of each agent. Suppressing the index k in the notation, we collect N data points by observing the state transitions $(\boldsymbol{\xi}_n, \boldsymbol{\upsilon}_n, \boldsymbol{\xi}_n^+)\ \forall n = 1, \ldots, N$, where $\boldsymbol{\xi}_n$ is the current state, $\boldsymbol{\upsilon}_n$ is the control input, and $\boldsymbol{\xi}_n^+$ is the evolved state. Let $\mathbf{y}_n$ denote the residual measurements of the agent defined as

$$\mathbf{y}_n = \boldsymbol{\xi}_n^+ - \mathbf{f}_k(\boldsymbol{\xi}_n, \boldsymbol{\upsilon}_n), \quad n = 1, \ldots, N, \tag{2}$$

which is used as input to train a zero-mean Gaussian Process defined for each state dimension, i.e., $\mathcal{GP}(\mathbf{0}, \kappa_d(\boldsymbol{\xi}, \boldsymbol{\xi}'))\ \forall d = 1, \ldots, D_x$. Here, we assume the choice of a squared-exponential kernel, i.e.,

$$\kappa_d(\boldsymbol{\xi}, \boldsymbol{\xi}') = \sigma_{h,d}^2 \exp\Big(-\tfrac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\xi}')^\top \mathbf{L}_d^{-1}(\boldsymbol{\xi} - \boldsymbol{\xi}')\Big), \quad \mathbf{L}_d = \mathrm{diag}(\ell_{1,\mathrm{d}}^2, \ldots, \ell_{\mathrm{D_x},\mathrm{d}}^2),$$

where $\{\sigma_{h,d}^2, \ell_{1,d}, \ldots, \ell_{D_x,d}\}$ are the hyperparameters of the d-th GP kernel, which can be estimated by maximizing the log-marginal likelihood on the training data. Let $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_N]^\top \in \mathbb{R}^{N \times D_x}$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]^\top \in \mathbb{R}^{N \times D_x}$, then the GP predictive mean and (diagonal) variance at an arbitrary location of interest $\boldsymbol{\xi}_*$ are given by

$$\mu_d(\boldsymbol{\xi}_*) = \mathbf{k}_{*,d}^\top \big(\mathbf{K}_d + \sigma_{n,d}^2 \mathbf{I}\big)^{-1} \mathbf{Y}_d, \quad d = 1, \ldots, D_x, \tag{3}$$

$$\sigma_d^2(\boldsymbol{\xi}_*) = \kappa_d(\boldsymbol{\xi}_*, \boldsymbol{\xi}_*) - \mathbf{k}_{*,d}^\top \big(\mathbf{K}_d + \sigma_{n,d}^2 \mathbf{I}\big)^{-1} \mathbf{k}_{*,d} \in \mathbb{R}, \quad d = 1, \ldots, D_x, \tag{4}$$

$$\boldsymbol{\mu}(\boldsymbol{\xi}_*) = \big(\mu_1(\boldsymbol{\xi}_*), \ldots, \mu_{D_x}(\boldsymbol{\xi}_*)\big)^\top \in \mathbb{R}^{D_x}, \tag{5}$$

$$\boldsymbol{\Sigma}(\boldsymbol{\xi}_*) = \mathrm{diag}\big(\sigma_1^2(\boldsymbol{\xi}_*), \ldots, \sigma_{\mathrm{D_x}}^2(\boldsymbol{\xi}_*)\big) \in \mathbb{R}^{\mathrm{D_x} \times \mathrm{D_x}}. \tag{6}$$

In these expressions, for each dimension d we form the kernel matrix $\mathbf{K}_d \in \mathbb{R}^{N \times N}$ with entries $[\mathbf{K}_d]_{nm} = \kappa_d(\boldsymbol{\xi}_n, \boldsymbol{\xi}_m)$, the kernel vector $\mathbf{k}_{*,d} \in \mathbb{R}^N$ with entries $[\mathbf{k}_{*,d}]_n = \kappa_d(\boldsymbol{\xi}_n, \boldsymbol{\xi}_*)$, and the target vector $\mathbf{Y}_d \in \mathbb{R}^N$ as the d-th column of $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top \in \mathbb{R}^{N \times D_x}$. Finally, the observation-noise variance $\sigma_{n,d}^2$ regularizes the fit and captures noise in $\mathbf{y}$ and is also estimated from the training data. This GP model yields both a mean-estimate $\boldsymbol{\mu}(\boldsymbol{\xi}_*)$ of the residual for correcting the nominal dynamics and an uncertainty measure to guide robust planning.

2.2 System model

Given the dynamical model (1), our control objective is to design a Model Predictive Control (MPC) strategy that minimizes the tracking error and control effort over a selected prediction horizon T. To this end, we formulate a finite-horizon MPC problem over the interval $\tau = t, \ldots, t + T$. For each agent k, we collect all their control inputs, positions, and reference trajectory locations over this interval into matrices

$$\mathbf{U}_k^t = \big[\mathbf{u}_k^t, \ldots, \mathbf{u}_k^{t+T-1}\big], \tag{7a}$$

$$\mathbf{X}_k^t = [\mathbf{x}_k^t, \ldots, \mathbf{x}_k^{t+T}], \tag{7b}$$

$$\mathbf{X}_k^{t,\mathrm{ref}} = [\mathbf{x}_k^{t,\mathrm{ref}}, \ldots, \mathbf{x}_k^{t+T,\mathrm{ref}}]. \tag{7c}$$

Now, for the kth agent, consider as cost function the sum of the squared tracking error plus a scaled control effort:

$$J_k(\mathbf{U}_k^t, \mathbf{X}_k^t) = \|\mathbf{X}_k^t - \mathbf{X}_k^{t,\mathrm{ref}}\|_F^2 + \alpha \|\mathbf{U}_k^t\|_F^2, \tag{8}$$

where

subscript F denotes the Frobenius norm, and α is a scaling hyperparameter which balances the accuracy versus actuator usage. Smaller values of α emphasize precise trajectory following, while larger values promote smoother, energy-efficient actuation.

The overall optimization problem for multi-agent navigation using MPC with Gaussian Process (GP)-modeled dynamics [18] can then be formulated as

$$\min_{\{\mathbf{X}_k^t, \mathbf{U}_k^t\}} \quad \sum_{k=1}^{K} J_k(\mathbf{U}_k^t, \mathbf{X}_k^t) \tag{9a}$$

$$\text{s.t.} \quad \mathbf{x}_k^{\tau+1} = \mathbf{f}_k(\mathbf{x}_k^\tau, \mathbf{u}_k^\tau) + \boldsymbol{\mu}_k^\tau \qquad \forall k \leq K, \forall\, \tau \in t, \dots t+T, \tag{9b}$$

$$\|\mathbf{u}_k^\tau\| \leq u_{\max} \qquad \forall k \leq K, \forall\, \tau \in t, \dots t+T. \tag{9c}$$

Here, the objective in (9a) defines the overall cost as the sum of the K costs of each agent. The constraint (9b) enforces the system dynamics, i.e., the next state $\mathbf{x}_k^{\tau+1}$ follows the known nominal model $\mathbf{f}_k(\mathbf{x}_k^\tau, \mathbf{u}_k^\tau)$ corrected by the GP mean $\boldsymbol{\mu}_k^\tau \triangleq \boldsymbol{\mu}_k(\mathbf{x}_k^\tau)$ derived from (3). Lastly, the constraint in (9c) ensures that the control input vector $\mathbf{u}_k^\tau$ respects the actuator limit $u_{\max}$, selected dependent on the system.

3 Proposed method

The MPC problem in (9) is decoupled across all agents. However, introducing the coupling collision-avoidance constraints necessitates a central node to solve the global optimization, which requires significant computation and communication resources, particularly for an increasing number of agents. We instead propose a distributed GP-MPC framework, where each agent only relies on local communication with its neighbors within a certain communication range. In addition, to ensure a safe distance between agents at all instances, taking into account model uncertainty, we further introduce probabilistic (chance) constraints, which can be derived from the GP predictive mean and variance. After introducing the chance constraints, we rewrite the dynamical model to obtain a convex approximation. Next, following the ADMM framework, we decompose the centralized optimization problem into local MPC subproblems for each agent. This results in an iterative consensus algorithm that coordinates these local controllers to recover performance comparable to the centralized scheme while ensuring probabilistic collision avoidance under model uncertainty.

3.1 Collision avoidance

To ensure a safe distance among agents, we introduce a collision-avoidance constraint that maintains a minimum separation r_{safe} between each pair of agents,

$$\|\mathbf{x}_k^\tau - \mathbf{x}_j^\tau\| \geq r_{\text{safe}}, \quad (k,j) \in \mathcal{E},\, k \neq j,\, \forall \tau \in t, \dots t+T. \tag{10}$$

We now treat each predicted state $\mathbf{x}_k^t$ as a Gaussian random variable whose mean $\mathbb{E}[\mathbf{x}_k^t]$ and variance $\boldsymbol{\Sigma}_k^t$ follows directly from the dynamical update in (9b) and the GP predictive equations (5),(6). We then define the relative displacement between two agents j and k as $\mathbf{r}_{kj}^t = \mathbf{x}_k^t - \mathbf{x}_j^t$, which is Gaussian distributed as

$$\mathbf{r}_{kj}^{\tau} \sim \mathcal{N}\big(\mathbb{E}[\mathbf{x}_k^{\tau}] - \mathbb{E}[\mathbf{x}_j^{\tau}],\ \boldsymbol{\Sigma}_k^{\tau} + \boldsymbol{\Sigma}_j^{\tau}\big).$$

To guarantee safe separation under this uncertainty, the deterministic collision-avoidance constraint in (10) is replaced by the chance constraint [17]

$$\Pr\big(\|\mathbf{r}_{kj}^{\tau}\| \geq r_{\text{safe}}\big) \geq 1 - \varepsilon, \quad \forall\, k \neq j, \tag{11}$$

where $\varepsilon \in (0, 1)$ is the allowable risk level.

3.2 Linearization of residual dynamics and safety margins

To make the non-convex GP-augmented dynamics in Eq.(9b) and the probabilistic collision constraint in Eq.(11) solvable with a convex optimizer, we linearize the GP mean and the inter-agent distance around nominal operating points. At each prediction step τ, let the pre-chosen linearization point for agent k be $\bar{\mathbf{x}}_k^{\tau}$. To linearize $\boldsymbol{\mu}_k(\mathbf{x}_k^{\tau})$ around $\bar{\mathbf{x}}_k^{\tau}$, we define for each state dimension d

$$\mu_d = \big[\boldsymbol{\mu}_k(\bar{\mathbf{x}}_k^{\tau})\big]_d \in \mathbb{R}, \quad \nabla_{\mathbf{x}}\mu_d = \left.\frac{\partial[\boldsymbol{\mu}_k(\mathbf{x})]_d}{\partial \mathbf{x}_k^{\tau}}\right|_{\bar{\mathbf{x}}_k^{\tau}} \in \mathbb{R}^{D_x}.$$

A first-order Taylor expansion of the d-th component of the GP mean then yields [25]

$$\big[\boldsymbol{\mu}_k(\mathbf{x}_k^{\tau})\big]_d \approx \mu_d + (\nabla_{\mathbf{x}}\mu_d)^{\top}\big(\mathbf{x}_k^{\tau} - \bar{\mathbf{x}}_k^{\tau}\big). \tag{12}$$

Under the squared-exponential kernel $\kappa_d(\mathbf{x}, \mathbf{x}')$ with length-scales $\mathbf{L}_d$ (3), the gradient is

$$\nabla_{\mathbf{x}}\mu_d = \sum_{n=1}^{N} \kappa_d\big(\bar{\mathbf{x}}_k^{\tau}, \boldsymbol{\xi}_n\big)\, \mathbf{L}_d^{-1}\big(\boldsymbol{\xi}_n - \bar{\mathbf{x}}_k^{\tau}\big)[(\mathbf{K}_d + \sigma_{n,d}^2\mathbf{I})^{-1}\mathbf{y}_d]_n \quad \in \mathbb{R}^{D_x} \tag{13}$$

Substituting (12) into the dynamics (9b) yields the linearized update

$$\mathbf{x}_k^{\tau+1} = \mathbf{f}_k(\mathbf{x}_k^{\tau}, \mathbf{u}_k^{\tau}) + \boldsymbol{\mu}_k\big(\bar{\mathbf{x}}_k^{\tau}\big) + \nabla_{\mathbf{x}}\boldsymbol{\mu}_k\big(\bar{\mathbf{x}}_k^{\tau}\big)\,\big(\mathbf{x}_k^{\tau} - \bar{\mathbf{x}}_k^{\tau}\big). \tag{14}$$

For collision avoidance, let the relative linearization point be $\bar{\mathbf{r}}_{kj}^{\tau} = \bar{\mathbf{x}}_k^{\tau} - \bar{\mathbf{x}}_j^{\tau}$. A first-order expansion of the inter-agent distance [9] gives

$$\tilde{r}_{kj}^{\tau} \triangleq \|\bar{\mathbf{r}}_{kj}^{\tau}\| + \frac{(\bar{\mathbf{r}}_{kj}^{\tau})^{\top}}{\|\bar{\mathbf{r}}_{kj}^{\tau}\|}\big(\mathbf{x}_k^{\tau} - \mathbf{x}_j^{\tau} - \bar{\mathbf{r}}_{kj}^{\tau}\big). \tag{15}$$

Since $\mathbf{r}_{kj}^{\tau}$ is Gaussian under the GP model, $\tilde{r}_{kj}^{\tau}$ is also Gaussian with

$$\tilde{r}_{kj}^{\tau} \sim \mathcal{N}\Big(\|\bar{\mathbf{r}}_{kj}^{\tau}\|,\ \frac{(\bar{\mathbf{r}}_{kj}^{\tau})^{\top}\Sigma_{kj}^{\tau}\,(\bar{\mathbf{r}}_{kj}^{\tau})}{\|\bar{\mathbf{r}}_{kj}^{\tau}\|^2}\Big),$$

where $\Sigma_{kj}^{\tau} = \Sigma_k^{\tau} + \Sigma_j^{\tau}$ [26]. Imposing the chance constraint on $\tilde{r}_{kj}^{\tau}$ then yields the convex approximation of (11)

$$\tilde{r}_{kj}^{\tau} \ \geq\ r_{\text{safe}} + \Phi^{-1}(1 - \varepsilon)\,\tilde{\sigma}_{kj}^{\tau}, \tag{16}$$

where we have introduced the shorthand

$$\tilde{\sigma}_{kj}^{\tau} \triangleq \sqrt{\frac{(\bar{\mathbf{r}}_{kj}^{\tau})^{\top} \Sigma_{kj}^{\tau} (\bar{\mathbf{r}}_{kj}^{\tau})}{\|\bar{\mathbf{r}}_{kj}^{\tau}\|^2}}.$$

Equation (16) now replaces (10). The full optimization problem is then given by

$$\min_{\{\mathbf{X}_k^t, \mathbf{U}_k^t\}} \quad \sum_{k=1}^{K} \sum_{\tau=t}^{t+T} \left\| \mathbf{x}_k^{\tau} - \mathbf{x}_k^{\tau,\mathrm{ref}} \right\|^2 + \alpha \left\| \mathbf{u}_k^{\tau} \right\|^2 \tag{17a}$$

$$\text{s.t.} \quad \mathbf{x}_k^{\tau+1} = \mathbf{f}_k(\mathbf{x}_k^{\tau}, \mathbf{u}_k^{\tau}) + \boldsymbol{\mu}_k(\bar{\mathbf{x}}_k^{\tau}) + \nabla_{\mathbf{x}} \boldsymbol{\mu}_k(\bar{\mathbf{x}}_k^{\tau}) \left(\mathbf{x}_k^{\tau} - \bar{\mathbf{x}}_k^{\tau}\right), \tag{17b}$$

$$\|\mathbf{u}_k^{\tau}\| \leq u_{\max}, \tag{17c}$$

$$\tilde{r}_{kj}^{\tau} \geq r_{\mathrm{safe}} + \Phi^{-1}(1-\varepsilon)\, \tilde{\sigma}_{kj}^{\tau}, \quad \forall j \in \mathcal{N}_k. \tag{17d}$$

3.3 Distributed formulation

To eliminate any single point of failure, we reformulate the centralized MPC () as a fully distributed problem using the Alternating Direction Method of Multipliers (ADMM) [27]. Observe that the cost functions have both vertex dependent variables for each agent $k \in \mathcal{K}$, and edge dependent variables for their neighbors. Each agent k maintains its own control sequence $\mathbf{U}_k^t$ as a local variable. The shared trajectory variables are combined as a vector $\mathbf{z}_{kj}^t \in \mathbb{R}^{2(T+1)D_x}$, which are defined for each pair $(k,j) \in \mathcal{E}$, i.e.,

$$\mathbf{z}_{kj}^t \triangleq [(\mathbf{x}_k^t)^{\top}, \ldots, (\mathbf{x}_k^{t+T})^{\top}, (\mathbf{x}_j^t)^{\top}, \ldots, (\mathbf{x}_j^{t+T})^{\top}]^{\top}. \tag{18}$$

Let the local copies of this shared vector $\mathbf{z}_{kj}^t$ at agent k and j be given by $\mathbf{y}_{kj,k}^t$ and $\mathbf{y}_{kj,j}^t$, respectively. For the kth agent, we then collect all these local copies as follows

$$\mathbf{Y}_k^t = [\mathbf{y}_{k1,k}^t, \mathbf{y}_{k2,k}^t, \ldots, \mathbf{y}_{k|\mathcal{N}_k|,k}^t] \tag{19}$$

where $|\mathcal{N}_k|$ denotes the number of agents in the neighborhood of agent k.

The optimization problem is rewritten with these variables, resulting in the following separable problem

$$\begin{aligned} \min_{\{\mathbf{U}_k^t, \mathbf{Y}_k^t\}_{k=1}^K, \{\mathbf{z}_{kj}^t\}_{kj\in\mathcal{E}}} \quad & \sum_{k\in\mathcal{K}} J_k(\mathbf{U}_k^t, \mathbf{Y}_k^t) \\ \text{s.t.} \quad & \mathbf{y}_{kj,k}^t - \mathbf{z}_{kj}^t = \mathbf{0},\ \mathbf{y}_{kj,j}^t - \mathbf{z}_{kj}^t = \mathbf{0} \\ & \mathbf{U}_k^t \text{ satisfies (17}b\text{), (17}c\text{)} \quad \forall k \in \mathcal{K} \\ & \mathbf{z}_{kj}^t \text{ satisfies (17}d\text{)} \quad \forall (k,j) \in \mathcal{E}. \end{aligned} \tag{20}$$

Regarding the final constraint, note that (17d) depends on $\mathbf{z}_{kj}^t$ (and thereby $\mathbf{y}_{kj}^t$) through (15), since $\mathbf{z}_{kj}$ combines the states from agents k and j.

To enforce an overall collision-risk budget ε using only each agent's own covariance, we split ε into two nonnegative parts and impose local chance constraints as

$$\begin{aligned} &\varepsilon_k + \varepsilon_j = \varepsilon, \\ &\Pr\big(\|\mathbf{r}_{kj}^t\| \geq r_{\text{safe}}\big) \geq 1 - \varepsilon_k, \quad \text{(agent } k' \text{ s local risk limit)}, \\ &\Pr\big(\|\mathbf{r}_{kj}^t\| \geq r_{\text{safe}}\big) \geq 1 - \varepsilon_j, \quad \text{(agent } j' \text{ s local risk limit)}. \end{aligned} \tag{21}$$

Agent k models

$$\tilde{r}_{kj,k}^t \sim \mathcal{N}\Big(\|\bar{\mathbf{r}}_{kj}^t\|,\ (\tilde{\sigma}_k^t)^2\Big), \quad \tilde{\sigma}_k^t = \sqrt{\frac{(\bar{\mathbf{r}}_{kj}^t)^\top \boldsymbol{\Sigma}_k^t\, (\bar{\mathbf{r}}_{kj}^t)}{\|\bar{\mathbf{r}}_{kj}^t\|^2}}, \tag{22}$$

while agent j uses

$$\tilde{r}_{kj,j}^t \sim \mathcal{N}\Big(\|\bar{\mathbf{r}}_{kj}^t\|,\ (\tilde{\sigma}_j^t)^2\Big), \quad \tilde{\sigma}_j^t = \sqrt{\frac{(\bar{\mathbf{r}}_{kj}^t)^\top \boldsymbol{\Sigma}_j^t\, (\bar{\mathbf{r}}_{kj}^t)}{\|\bar{\mathbf{r}}_{kj}^t\|^2}}. \tag{23}$$

Hence, the local chance constraints become

$$\tilde{r}_{kj,k}^t \ \geq\ r_{\text{safe}} + \Phi^{-1}(1-\varepsilon_k)\,\tilde{\sigma}_k^t, \qquad \tilde{r}_{kj,j}^t \ \geq\ r_{\text{safe}} + \Phi^{-1}(1-\varepsilon_j)\,\tilde{\sigma}_j^t. \tag{24}$$

By the union bound $\Pr(X \cup Y) \leq \Pr(X) + \Pr(Y)$ we recover

$$\Pr\big(\tilde{r}_{kj}^t < r_{\text{safe}}\big) \leq \varepsilon_k + \varepsilon_j = \varepsilon, \tag{25}$$

thus guaranteeing $\Pr\big(\tilde{r}_{kj}^t \geq r_{\text{safe}}\big) \geq 1 - \varepsilon$, which is the linearized version of (11).

3.4 Local problems

Note that (20) is separable since only local and neighborhood information is required, and can be solved using distributed algorithms.

Let $\rho > 0$ be a known penalty parameter, and let $\boldsymbol{\lambda}_{kj,k}^t$ denote the Lagrange multiplier that agent k associates with the consensus constraint on the edge (k, j), while $\boldsymbol{\lambda}_{kj,j}^t$ denotes the corresponding multiplier held by its neighbor j [27]. Then, for each agent k at any time step $t > 0$, the augmented Lagrangian for (20) is written in the form

$$\begin{aligned} \mathcal{L}_k^t = J_k(\mathbf{U}_k^t, \mathbf{Y}_k^t) &+ \sum_{j \in \mathcal{N}_k} (\boldsymbol{\lambda}_{kj,k}^t)^\top (\mathbf{y}_{kj,k}^t - \mathbf{z}_{kj}^t) \\ &+ (\boldsymbol{\lambda}_{kj,j}^t)^\top (\mathbf{y}_{kj,j}^t - \mathbf{z}_{kj}^t) + \frac{\rho}{2}(\|\mathbf{y}_{kj,k}^t - \mathbf{z}_{kj}^t\|^2 + \|\mathbf{y}_{kj,j}^t - \mathbf{z}_{kj}^t\|^2). \end{aligned} \tag{26}$$

Note that the cost function $\mathcal{L}_k^t$ must be solved, subject to the local constraints as follows

$$\begin{aligned} \min_{\mathbf{U}_k^t,\, \mathbf{Y}_k^t,\, \{\mathbf{z}_{kj}^t, \boldsymbol{\lambda}_{kj}^t\}_{j \in \mathcal{N}_k}} \quad & \mathcal{L}_k^t(\mathbf{U}_k^t,\ \mathbf{Y}_k^t,\ \{\mathbf{z}_{kj}^t, \boldsymbol{\lambda}_{kj}^t\}_{j \in \mathcal{N}_k}) \\ \text{s.t.} \quad & (17b), (17c), (17d). \end{aligned} \tag{27}$$

3.5 ADMM updates

Now, each agent iteratively solves the problem (27) at iteration i, using the following procedure [27]. Each agent $k \in \mathcal{K}$ solves (26) with respect to its local variables $\mathbf{U}_k^t$ and the shared trajectories $\mathbf{Y}_k^t$. Then, the updated values of $\mathbf{y}$ are communicated, e.g., agent k receives

$(\mathbf{y}^t_{kj,j})^{i+1}$ from all $j \in \mathcal{N}_k$ and vice versa. Given these inputs, the agents update the consensus variables

$$(\mathbf{z}^t_{kj})^{i+1} = \frac{1}{2}((\mathbf{y}^t_{kj,k})^{i+1} + (\mathbf{y}^t_{kj,j})^{i+1}), \tag{28}$$

and finally update

$$(\boldsymbol{\lambda}^t_{kj,k})^{i+1} = (\boldsymbol{\lambda}^t_{kj,k})^{i} + \rho\left[(\mathbf{y}^t_{kj,k})^{i+1} - (\mathbf{z}^t_{kj})^{i+1}\right]. \tag{29}$$

Then, the linearization points $\bar{\mathbf{x}}^t_k$ are updated using the current solution $(\mathbf{Y}^t_k)^{i+1}$. Updating the linearization points at each iteration improves the accuracy of the approximation, since a first-order model is used. The procedure is repeated until convergence is achieved or the maximum number of iterations is reached. Convergence is achieved when $(\|(\mathbf{y}^t_{kj})^i - (\mathbf{z}^t_{kj})^i\|) < \epsilon$, where ϵ is the convergence threshold. The algorithm is summarized in Algorithm 1.

Algorithm 1 Proposed: LinearizedGP-Chance (for agent $k \in \mathcal{K}$ at $t > 0$)

Input: $\mathbf{x}^t_k$, $\mathcal{N}_k$, ρ, ε
Initialize: $(\mathbf{z}_{kj})^0 \leftarrow \mathbf{0}$, $(\boldsymbol{\lambda}_{kj,k})^0 \leftarrow \mathbf{0}$, $(\boldsymbol{\lambda}_{kj,j})^0 \leftarrow \mathbf{0}$, $i \leftarrow 0$
repeat
  Calculate $(\mathbf{Y}^t_k)^{i+1}$ and $(\mathbf{U}^t_k)^{i+1}$ using (27)
  Communicate $(\mathbf{y}^t_{kj,k})^{i+1}$ to all neighbors $j \in \mathcal{N}_k$
  Receive $(\mathbf{y}^t_{kj,j})^{i+1}$ from all neighbors $j \in \mathcal{N}_k$
  Calculate $(\mathbf{z}^t_{kj})^{i+1}$ using (28)
  Calculate $(\boldsymbol{\lambda}^t_{kj,k})^{i+1}$ and $(\boldsymbol{\lambda}^t_{kj,j})^{i+1}$ using (29)
  Update $\bar{\mathbf{x}}^t_k$ using solution $\mathbf{Y}^t_k$
  Update $i \leftarrow i + 1$
until convergence i.e., $(\|(\mathbf{y}^t_{kj})^i - (\mathbf{z}^t_{kj})^i\|) < \varepsilon$
Output: $(\mathbf{Y}^t_k)^i$ **and** $(\mathbf{U}^t_k)^i$

4 Simulations

We show the performance of the proposed algorithm in a simple simulation scenario.

4.1 Experimental setup

The simulation scenario implements $K = 3$ agents with intersecting trajectories within a square box in $D_x = 2$ dimensions of range $[-4, 4]$ in each dimension. Recollect the dynamical model (1). We consider the nominal (known) linear system dynamics for any agent $k \in \mathcal{K}$ at time t is given by

$$\mathbf{f}_k(\mathbf{x}^t_k, \mathbf{u}^t_k) = \boldsymbol{A}\mathbf{x}^t_k + \boldsymbol{B}\mathbf{u}^t_k, \tag{30}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}. \tag{31}$$

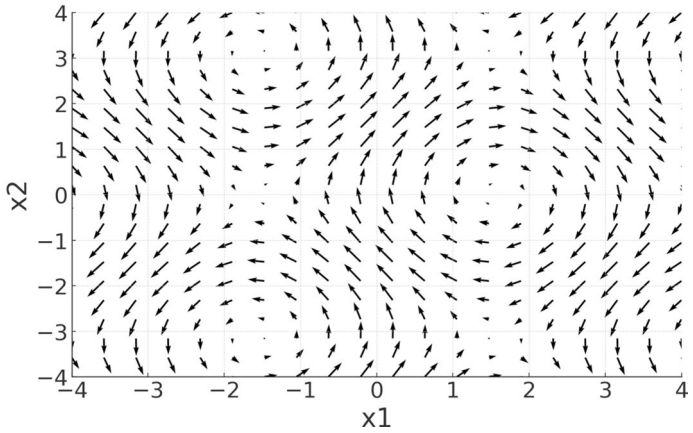

Fig. 1 The unknown nonlinear function $\mathbf{g}(\mathbf{x})$, modeling wind disturbance

Table 1 Compared methods

Method	$\mathbf{x}_k^{t+1}$	Collision avoidance
Nominal-NoChance [8]	$\mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t)$	$\tilde{r}_{kj}^t \geq r_{\text{safe}}$
Nominal-Chance	$\mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t)$	$\tilde{r}_{kj}^t \geq r_{\text{safe}} + \Phi^{-1}(1-\varepsilon)\,\tilde{\sigma}_{kj}^t$
Nonlinear-NoChance [31]	$\mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t) + \mathbf{g}_k(\mathbf{x}_k^t)$	$\tilde{r}_{kj}^t \geq r_{\text{safe}}$
Nonlinear-Chance	$\mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t) + \mathbf{g}_k(\mathbf{x}_k^t)$	$\tilde{r}_{kj}^t \geq r_{\text{safe}} + \Phi^{-1}(1-\varepsilon)\,\tilde{\sigma}_{kj}^t$
LinearizedGP-NoChance	$\mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t) + \boldsymbol{\mu}(\bar{\mathbf{x}}) + \nabla_{\mathbf{x}}\boldsymbol{\mu}(\bar{\mathbf{x}})\,(\mathbf{x}_k^t - \bar{\mathbf{x}})$	$\tilde{r}_{kj}^t \geq r_{\text{safe}}$
LinearizedGP-Chance (Proposed)	$\mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t) + \boldsymbol{\mu}(\bar{\mathbf{x}}) + \nabla_{\mathbf{x}}\boldsymbol{\mu}(\bar{\mathbf{x}})\,(\mathbf{x}_k^t - \bar{\mathbf{x}})$	$\tilde{r}_{kj}^t \geq r_{\text{safe}} + \Phi^{-1}(1-\varepsilon)\,\tilde{\sigma}_{kj}^t$

In addition, we consider a scenario where the agents experience a vortex-like wind disturbance, modeled by the unknown nonlinear function as follows.

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} 0.1\sin(x_2) \\ 0.1\cos(x_1) \end{bmatrix}, \tag{32}$$

which is visualized in Fig. 1.

Finally, $\mathbf{w}_k^t \sim \mathcal{N}(0, 0.1\mathbf{I})$ is Gaussian process noise. In a practical scenario, for platforms such as drones or rovers, training data is typically collected by operating the agent in a controlled environment and recording state transitions to compute residuals [28, 29]

To train the GP model, for each agent we collect $N = 100$ sample points $(\boldsymbol{\Xi}, \mathbf{Y})$ as in (2), with entries of $\boldsymbol{\Xi}$ selected uniformly within the box, random input $\|\boldsymbol{\upsilon}_n\| < u_{max}$, and $\mathbf{y}_n = \boldsymbol{\xi}_n^+ - \mathbf{f}_k(\boldsymbol{\xi}_n, \boldsymbol{\upsilon}_n)$. The hyperparameters $\boldsymbol{\theta}$ of the GP are optimized using L-BFGS[30].

In the simulation, the 3 agents attempt to reach a target position $\mathbf{x}_k^{\text{ref}}$, independent of t, while avoiding collision with a distance $r_{\text{safe}} = 0.5$. The parameter $\varepsilon_k = \varepsilon_j = 0.05$ dictates the allowed probability of collision for the chance constraint. The naive straight trajectories would intersect. The target is considered reached when $\|\mathbf{x}_k - \mathbf{x}_k^{\text{ref}}\| < 0.1$.

We compare six models which are listed in Table 1. The nominal algorithm implements $\mathbf{x}_k^{t+1} = \mathbf{f}_k(\mathbf{x}_k^t, \mathbf{u}_k^t)$, the nonlinear algorithm adds the nonlinear term $\mathbf{g}_k(\mathbf{x}_k^t)$, the Linearized GP uses the GP model (14). For each model, we used the deterministic form and the chance-constraint form, which is enabled by the trained GP.

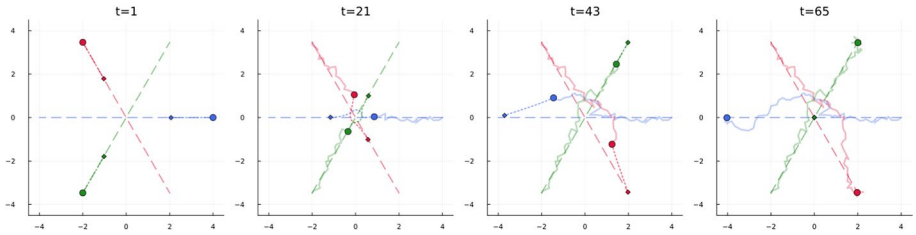

Fig. 2 Example trajectories where model nonlinearities are ignored; deterministic collision-avoidance constraint (Nominal-NoChance)

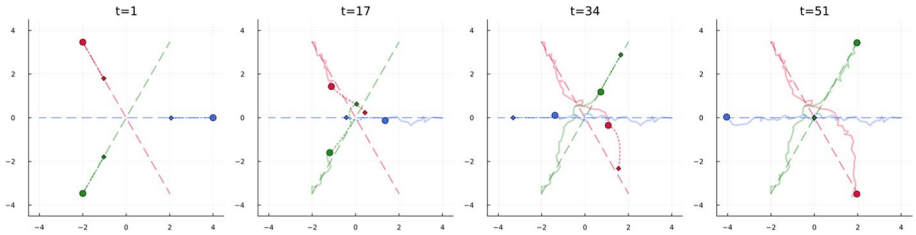

Fig. 3 Example trajectories resulting from the proposed method: Linearized GP with chance constraint

4.2 Results

Figures 2–3 show the resulting trajectories with the most simple Nominal-NoChance and the proposed method, respectively. In the plots, the heavy dot shows the current position at time t, the smaller dot shows the predicted position at time $t + T$, and the dashed line is the straight line to the target for reference. The solid line shows the trajectory up to time t, where the dashed line shows the predicted trajectory from time t to $t + T$. The trajectories appear jagged because of the simple dynamical model. The last panel shows the timestamp when the target has been reached.

Figure 2 shows the baseline case, where ignoring model nonlinearities and uncertainties leads to overcorrected trajectories once collision constraints are violated. Figure 3 presents the proposed framework, which combines learned residual dynamics with chance constraints to achieve efficient trajectories and robust collision avoidance. The target is reached quickly ($t = 51$) without any violation of collision constraints.

Figure 4 shows convergence for different hyperparameter values ρ for all described algorithms. On the y-axis, we see $(\|(\mathbf{y}_{kj}^t)^i - (\mathbf{z}_{kj}^t)^i\|)$ for each agent, and then the average of all agents for all timesteps. Higher values of ρ give slower convergence, but can give more stable results (although this behavior is not seen for all algorithms). All methods converge within 60 ADMM iterations.

Figure 5 shows the inter-agent distance over time for each evaluated method. The solid line gives the average over 10 runs and the shaded area shows the standard deviation. (Each color shows the distance of a pair of agents.) It is seen that several methods violate the safe distance constraint, and that employing chance constraints consistently results in safer inter-agent distances. This highlights the benefit of incorporating uncertainty quantification into the collision-avoidance constraint, which leads to more robust and safer agent interactions.

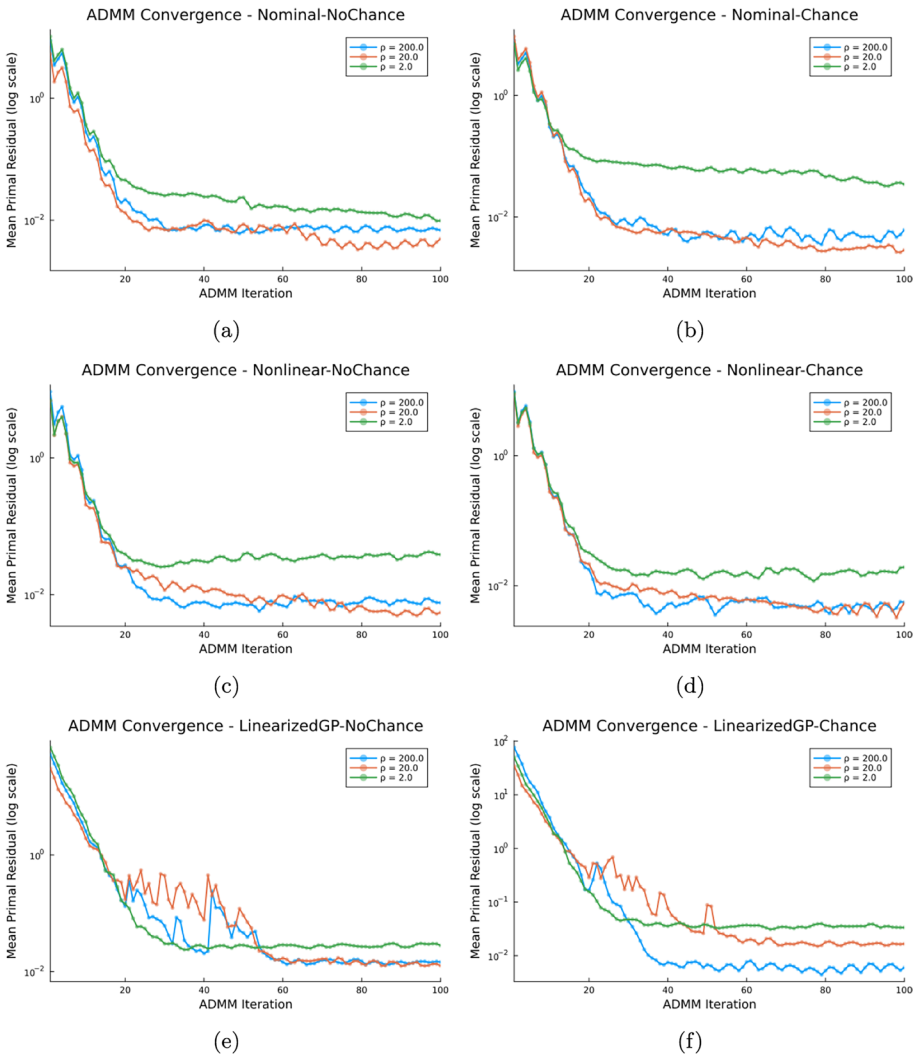

Fig. 4 ADMM convergence results for different hyperparameter values ρ, showing that algorithms converge over a wide range of parameter settings

Figure 6 shows the total path lengths traveled by each agent under the different methods, averaged over 10 runs. Each color represents one agent. The dashed line shows the naive straight trajectory length, giving the minimal possible path, but subject to potential collisions. The actual paths are longer than the minimum due to the noisy model, and due to the extra distance covered to ensure collision avoidance. The nominal model with a deterministic collision-avoidance constraint has the shortest average path length; however, these paths are not safe. For the nonlinear and linearized GP methods, we see that the average path length is shorter when the chance constraint for collision is used. This is due to safer trajectories, reducing the need for dramatic corrections when agents get close.

In summary, the simulations showed that the Linearized GP is effective to capture the residual dynamics. Using the GP also makes it possible to include the chance constraint, which leads to safer navigation. When uncertainty is considered, agents keep a larger

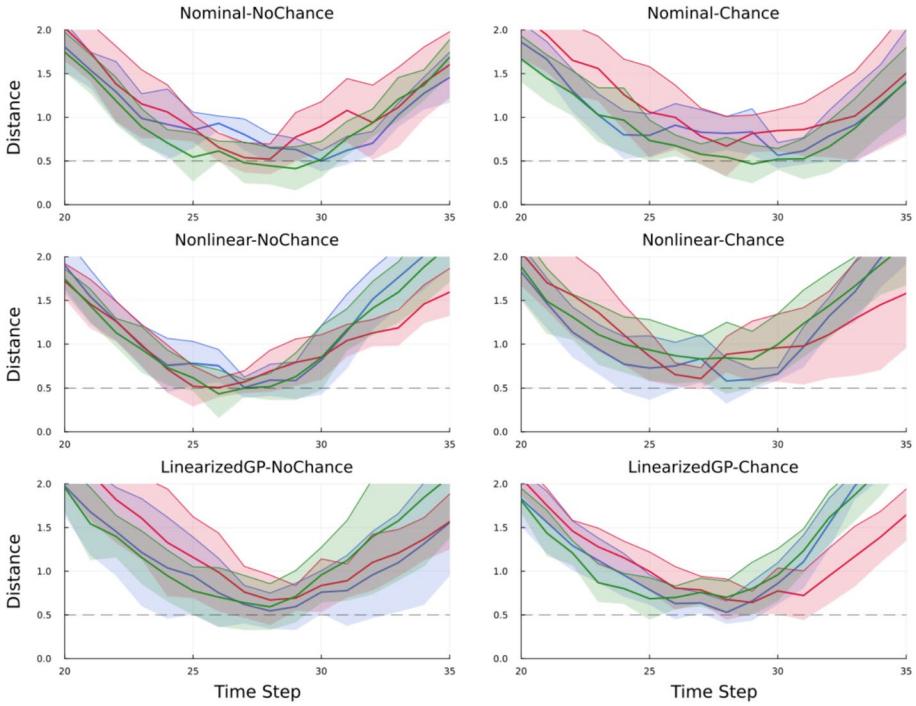

Fig. 5 Inter-agent distances for each pair of agents, for each of the considered methods. The dashed line represents the safe distance r_{safe}

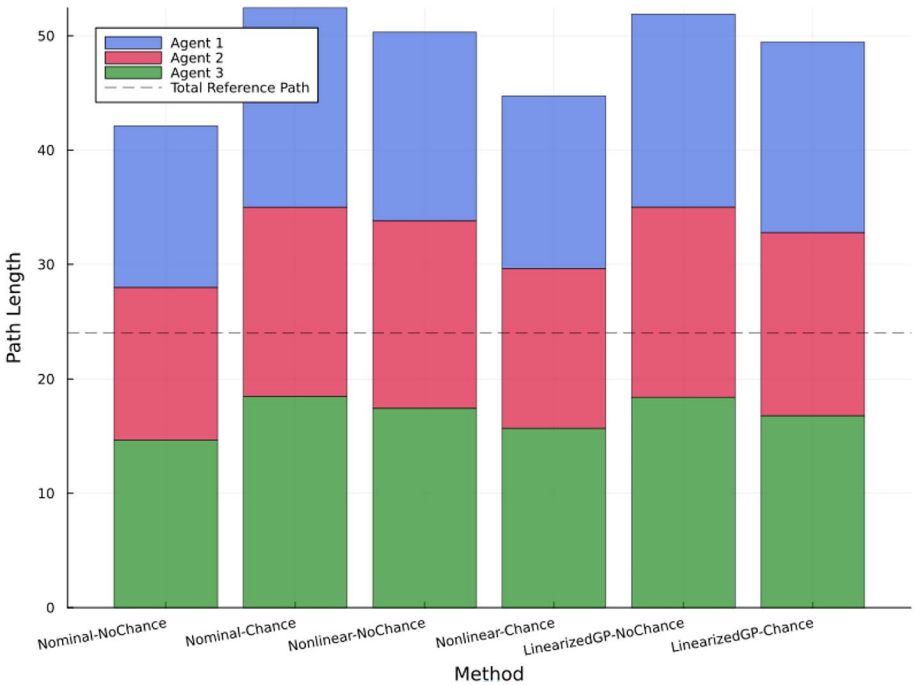

Fig. 6 Total path lengths traveled by each agent, averaged over 10 runs. The dashed line represents the straight trajectory length (subject to collisions)

distance from each other and avoid collisions more reliably. Path lengths are also shorter with the proposed method, because the agents follow smoother and safer trajectories, so they do not need sudden corrections. The deterministic methods may look efficient in distance, but they do not guarantee safety. The algorithm consistently achieves consensus for a range of parameters ρ, showing it handles tuning well.

Each agent solves an MPC cost function at time t, which now includes a GP-linearization step and an ADMM update. The local quadractic program with linearized GP constraints has a computational cost of $\mathcal{O}(D_x^3 T^3)$ for each iteration. Communication per ADMM iteration consists of exchanging the $2D_x T$–dimensional variables $(\mathbf{y}_{kj,k}^t)^i$ and $(\mathbf{y}_{kj,j}^t)^i$ with each neighbor. With $|\mathcal{N}_k|$ neighbors, that is $2D_x T|\mathcal{N}_k|$ numbers sent and received each iteration. Because each agent's computation is local and each edge only carries coupling variables, overall complexity scales roughly linearly in the number of edges $|\mathcal{E}|$.

4.3 Discussion

It is worth noting that integrating GP modeling and ADMM coordination requires careful parameter tuning (e.g., the risk level ϵ), which in turn offers substantial benefits. For agile agents like drones, standard MPC often yields overly conservative trajectories or violates safety constraints under unmodeled wind disturbances. Our proposed framework justifies its complexity by explicitly balancing path efficiency with probabilistic safety. Nevertheless, our approach has some limitations. First, probabilistic safety guarantees are approximate; because collision avoidance relies on a linearized distance surrogate, large linearization errors could degrade actual safety margins. Second, state covariance propagation over the prediction horizon is currently simplified and requires a more rigorous formulation for practical scenarios. Third, the offline-trained GP cannot be readily adapted to non-stationary disturbances in real time. Finally, extending this framework to higher-order kinematics or larger state spaces poses scalability challenges [32]. Higher-dimensional systems require more training data (N) to model residual dynamics accurately [33] Since GP evaluation scales poorly with N, repeated computations within the real-time MPC loop can become a bottleneck [34]. Furthermore, introducing complex dynamics may compromise the recursive feasibility of the local MPC solvers and hinder ADMM convergence.

5 Conclusion

In this paper, we presented a fully decentralized Gaussian Process–augmented Model Predictive Control (GP-MPC) framework for navigation with collision avoidance for multi-agent systems. Each agent maintains its own locally trained GP model to capture residual dynamics and quantify predictive uncertainty, enabling adaptive tightening of collision-avoidance margins without sharing full covariance information. Through a distributed ADMM scheme, agents exchange only minimal trajectory snippets with their immediate neighbors, eliminating the need for a central coordinator and significantly reducing both computational and communication burdens.

Simulation studies demonstrate that our linearized GP-MPC with chance constraints consistently outperforms both nominal and fully nonlinear MPC baselines, achieving faster convergence to targets while respecting a prescribed risk level. The incorporation

of uncertainty estimates leads to smoother trajectories and shorter average path lengths under probabilistic safety guarantees. Convergence analysis further confirms that the method reliably reaches consensus across a wide range of algorithmic parameters, underscoring its robustness to tuning.

The fully decentralized architecture ensures scalability as the number of agents grows, and yet each controller's computational load remains constant and communication is confined to local neighborhoods. Future work will focus on deriving formal probabilistic bounds for the chance-constraint approximation to strengthen safety guarantees, integrating online GP updates or sparse approximations for enhanced adaptability in dynamic environments, and extending the framework to heterogeneous agent teams navigating more complex obstacle fields to broaden its applicability to real-world robotic swarms and autonomous vehicle fleets. In our future work, we aim to study the scalability of the proposed solutions for more complex real-world scenarios and higher-order kinematics. To handle non-stationary disturbances, future extensions will explore online GP learning. Subsequent work will involve a formal characterization of covariance propagation over the prediction horizon, as well as an investigation into bounding the errors introduced by linearizing the chance constraints.

Author Contributions
EHJR conceived the study, developed the methodology, and wrote the main manuscript text. AJV contributed to the analysis, interpretation, and structure of the paper. RTR gave inputs to the concept and development of the methodology. All authors reviewed, edited, and approved the final manuscript.

Funding
Not applicable.

Data Availability
No datasets were generated or analyzed during the current study.

Declarations

Data and materials availability
Not applicable.

Code availability
Code available at https://github.com/asil-lab/EHJR-GPMPC-chance

Conflict of interest
Not applicable.

Conflict of interest
The authors declare no conflict of interest.

Received: 9 September 2025 Accepted: 13 February 2026
Published online: 11 March 2026

References

1. D.Q. Mayne, J.B. Rawlings, C.V. Rao, P.O.M. Scokaert, Constrained model predictive control: stability and optimality. Automatica **36**, 789–814 (2000)
2. L. Ferranti, L. Lyons, R.R. Negenborn, T. Keviczky, J. Alonso-Mora, Distributed nonlinear trajectory optimization for multi-robot motion planning. IEEE Trans. Control Syst. Technol. **31**(2), 809–824 (2022)
3. E. H. Riemens, R. T. Rajan, Distributed navigation with dynamic obstacles, in ICASSP 2025–2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5, IEEE, (2025)
4. J. Kocijan, A. Girard, B. Banko, R. Murray-Smith, Dynamic systems identification with Gaussian processes. Math. Comput. Model. Dyn. Syst. **11**, 411–424 (2005)
5. M. P. Deisenroth, C. E. Rasmussen, PILCO: A Model-Based and Data-Efficient Approach to Policy Search
6. L. Hewing, J. Kabzan, M.N. Zeilinger, Cautious Model Predictive Control Using Gaussian Process Regression. IEEE Trans. Control Syst. Technol. **28**, 2736–2743 (2020)

7. S. Kamthe, M. Deisenroth, Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control. in Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, pp. 1701–1710, PMLR, (2018)
8. F. Rey, Z. Pan, A. Hauswirth, J. Lygeros, Fully Decentralized ADMM for Coordination and Collision Avoidance, in 2018 European Control Conference (ECC), pp. 825–830, (2018)
9. F. Augugliaro, A. P. Schoellig, R. D'Andrea, Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1917–1922, Oct. (2012)
10. W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, D. Rus, Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention, in 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1928–1935, May (2017)
11. C. E. Luis, M. Vukosavljev, A. P. Schoellig, Angela P. Schoellig, Online Trajectory Generation with Distributed Model Predictive Control for Multi-Robot Motion Planning, (2019)
12. D. Zhou, Zijian Wang, Z. Wang, Zijian Wang, S. Bandyopadhyay, M. Schwager, Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells, 2, pp. 1047–1054, (2017)
13. H. Zhu, B. Brito, J. Alonso-Mora, Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware Voronoi cells. Auton. Robot. **46**, 401–420 (2022)
14. R. Hegde, D. Panagou, Multi-agent motion planning and coordination in polygonal environments using vector fields and model predictive control, in 2016 European Control Conference (ECC), pp. 1856–1861, (2016)
15. B. Alrifaee, M. G. Mamaghani, D. Abel, Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles, in 2014 IEEE International Symposium on Intelligent Control (ISIC), pp. 1583–1588, (2014)
16. D. Lyons, J.-P. Calliess, U. D. Hanebeck, Chance constrained model predictive control for multi-agent systems with coupling constraints, in 2012 American Control Conference (ACC), pp. 1223–1230, (2012)
17. X. Zhang, J. Ma, Z. Cheng, S. Huang, S.S. Ge, T.H. Lee, Trajectory generation by chance-constrained nonlinear MPC With probabilistic prediction. IEEE Trans. Cybernet. **51**, 3616–3629 (2021)
18. V.-A. Le, T. X. Nghiem, Gaussian Process Based Distributed Model Predictive Control for Multi-agent Systems using Sequential Convex Programming and ADMM, in 2020 IEEE Conference on Control Technology and Applications (CCTA), pp. 31–36, (2020)
19. S. Anderson, K. Byl, J. P. Hespanha, Experiment Design with Gaussian Process Regression with Applications to Chance-Constrained Control, in 2023 62nd IEEE Conference on Decision and Control (CDC), pp. 3931–3938, (2023)
20. A. Saviolo, J. Frey, A. Rathod, M. Diehl, G. Loianno, Active learning of discrete-time dynamics for uncertainty-aware model predictive control. IEEE Trans. Rob. **40**, 1273–1291 (2024)
21. H. Wang, J. Wang, Model Predictive Tracking Control for Three-wheeled Mobile Robots Combined with Gaussian Process Regression, in 2024 36th Chinese Control and Decision Conference (CCDC), pp. 4628–4633 (2024)
22. Y. Dang, Y. Huang, X. Shen, D. Zhu, Z. Chu, Incremental sparse Gaussian process-based model predictive control for trajectory tracking of unmanned underwater vehicles. IEEE Robot. Automat. Lett. **10**, 2327–2334 (2025)
23. A. Lederer, Z. Yang, J. Jiao, S. Hirche, Cooperative Control of Uncertain Multiagent Systems via Distributed Gaussian Processes. IEEE Trans. Autom. Control **68**, 3091–3098 (2023)
24. C. Williams and C. Rasmussen, Gaussian Processes for Regression, in Advances in Neural Information Processing Systems, vol. 8, MIT Press, (1995)
25. T. X. Nghiem, Linearized Gaussian Processes for Fast Data-driven Model Predictive Control, in 2019 American Control Conference (ACC), pp. 1629–1634, (2019). ISSN: 2378–5861
26. H. Zhu, J. Alonso-Mora, Chance-constrained collision avoidance for MAVs in dynamic environments. IEEE Robot. Automat. Lett. **4**, 776–783 (2019)
27. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. MAL **3**, 1–122 (2011)
28. Torrente, G., Kaufmann, E., Föhn, P., Scaramuzza, D. (2021) Data-Driven MPC for Quadrotors. IEEE Robot. Autom. Lett. **6**(2), 3769–3776. https://doi.org/10.1109/LRA.2021.3061307
29. Haninger, K., Hegeler, C., Peternel, L. (2022) Model predictive control with gaussian processes for flexible multi-modal physical human robot interaction. in 2022 International Conference on Robotics and Automation (ICRA), pp. 6948–6955. https://doi.org/10.1109/ICRA46639.2022.9811590
30. D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization. Math. Program. **45**, 503–528 (1989)
31. N. Gafur, G. Kanagalingam, A. Wagner, M. Ruskowski, Dynamic Collision and Deadlock Avoidance for Multiple Robotic Manipulators, IEEE Access, vol. 10, pp. 55766–55781, (2022). Conference Name: IEEE Access
32. Mishra, A., Rajan, R.T. (2025) Estimation of relative kinematic parameters of an anchorless network. IEEE Trans. Sig. Inf. Process Netw. **11**, 831-844. https://doi.org/10.1109/TSIPN.2025.3557585
33. Mishra, A., Rajan, R.T. (2026) Domain-aware Gaussian process state-space models. Sig. Process. **238**, 110003. https://doi.org/10.1016/j.sigpro.2025.110003
34. Balc, A.E., Rajan, R.T. (2026) Multiple model recursive gaussian process for robust target tracking. IEEE Open J. Sig. Process. **7**, 23–31. https://doi.org/10.1109/OJSP.2025.3646127

Publisher's Note