JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 44, No. 3, March 2021

Random Finite Set Theory and Centralized Control of Large Collaborative Swarms

Bryce Doerr* and Richard Linares†
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
Pingping Zhu‡
Marshall University, Huntington, West Virginia 25755
and
Silvia Ferrari§
Cornell University, Ithaca, New York 14853

https://doi.org/10.2514/1.G004861

Controlling large swarms of robotic agents presents many challenges, including, but not limited to, computational complexity due to a large number of agents, uncertainty in the functionality of each agent in the swarm, and uncertainty in the swarm's configuration. This work generalizes the swarm state using random finite set (RFS) theory and solves a centralized control problem with a quasi-Newton optimization through the use of model predictive control (MPC) to overcome the aforementioned challenges. This work uses the RFS formulation to control the distribution of agents assuming an unknown or unspecified number of agents. Computationally efficient solutions are also obtained via the MPC version of the iterative linear quadratic regulator (ILQR), a variant of differential dynamic programming. Information divergence is used to define the distance between the swarm RFS and the desired swarm configuration through the use of the modified L_2^2 distance. Simulation results using MPC and ILQR show that the swarm intensity converges to the desired intensity. Additionally, the RFS control formulation is shown to be very flexible in terms of the number of agents in the swarm and configuration of the desired Gaussian mixtures. Lastly, the ILQR and the Gaussian mixture probability hypothesis density filter are used in conjunction to solve a spacecraft relative motion problem with imperfect information to show the viability of centralized RFS control for this real-world scenario.

I. Introduction

CONTROL of large collaborative networks or swarms is currently an emerging area for controls development. Typically, a swarm network comprises tiny robots with limited actuators that perform specific tasks in some collective configuration. For example, the swarm can use its collective effort to grasp or move in a changing environment that can offer more flexibility and redundancy to meet a goal compared with the abilities of a single agent [1]. Specifically in space applications, swarm control of satellites and rovers can be used for the exploration of asteroids and other celestial bodies of interest [2] or areas of assembly and construction on-orbit, including constructing space observatories and space habitats [3]. Swarms involving UAVs have proven to be widely useful in military applications such as search-and-rescue missions, communication relaying, border patrol, surveillance, and mapping of hostile territory [4]. From these engineering applications, the use of collaborative swarms is an attractive option to meet objectives that require flexibility and redundancy in a changing environment.

For collaborative swarms, several control techniques have been implemented to date. With centralized control, one agent in the swarm computes the overall swarm control and manages the control execution for individual agents, allowing it to oversee the other agents' system processes [5]. Unfortunately, centralized control suffers from two main problems. As the number of agents in the swarm increases, the computational workload becomes more expensive [6]. This is especially true when the swarm agents are low cost and are located in an unknown environment [7]. For example, formation control was applied experimentally to 1024 low-cost Kilobot robots, which took 12 h to converge to a specific formation [7]. Additionally, centralized control is not robust against individual agent failures [8]. With a thousand low-cost agents present in a swarm, communication, actuation, and sensing are performed with less reliability. Thus, control formulation must be found that considers the computational performance for control of low-cost agents as well as flexibility during uncertainty.

Random finite set (RFS) formalism provides a generalization of the state space for multi-agent systems that can be used for control [9–11]. It is used to solve a stochastic trans-dimensional problem, where the dimension of the state space is an unknown a priori (unknown number of agents). RFS allows for the probability density function over a collection of state spaces to be defined, which provides a potential hypothesis for the true number of agents. Then a Bayesian estimation problem is formulated, and approximate solutions are used through the Gaussian mixture probability hypothesis density (GM-PHD) filter [12]. Other than the GM-PHD filter, many extensions around RFS theory have been made using estimation and simultaneous localization and mapping techniques including the cardinalized probability hypothesis density (CPHD) filter and the generalized labeled multi-Bernoulli (GLMB) filter [9,12–14].

By using RFS theory to model multi-agent systems, the time-varying number of agents and their states can be jointly estimated from measurement sets including data association uncertainty, clutter, and noise [9,12]. The agents and measurements are modeled as RFSs, and the probability hypothesis density (PHD) filter is used to propagate the estimate forward in time. The RFS model has been used previously with a potential model to describe the temporal evolution of the probabilistic description of a robotic swarm to promote coordination [10]. Other work has developed control for individual agents using the estimated RFS state in a centralized fashion [11]. As

Received 4 October 2019; revision received 30 August 2020; accepted for publication 1 September 2020; published online 24 November 2020.

*Postdoctoral Fellow, Department of Aeronautics and Astronautics; bdoerr@mit.edu. Member AIAA.

†Charles Stark Draper Assistant Professor, Department of Aeronautics and Astronautics; linaresr@mit.edu. Senior Member AIAA.

‡Assistant Professor, Department of Computer Science and Electrical Engineering; zhup@marshall.edu. Member AIAA.

§John Brancaccio Professor, Sibley School of Mechanical and Aerospace Engineering; ferrari@cornell.edu. Senior Member AIAA.

an introduction to RFSs, the GM-PHD filter is explored for application to RFS control.

Other models were developed to represent the behavior for swarm agents in space and time including probabilistic swarm guidance and distributed optimal control, which has developed efficient decision making for swarm control [15–19]. Probabilistic swarm guidance has been used to enable swarms to converge to target distributions through distributed control [15]. Distributed control is defined as the reformulation of the control problem as a set of interdependent subproblems and solving these subproblems [20]. Probabilistic swarm guidance solves issues that involve a large number of agents, also identified as "computationally complex," by controlling the swarm density distribution of the agents [15]. The distributed optimal control method is a method that controls multi-agent systems by modeling the agents as Gaussian mixtures and using an integral cost function that is optimized to the advection equation [16–18]. The control laws themselves are determined using potential functions that attract the agent distributions to the desired state and repel the distribution from obstacles [21]. By minimizing the objective function based on distributions using the necessary conditions of optimality, the optimal control law is found using the potential function. The distributed optimal control method was also expanded to use the Kullback–Leibler divergence metric using distributions in the objective function for the use of path planning [19]. This provides a discovery to a whole class of divergence measures of distributions that can provide converging optimal control solutions to multi-agent systems.

Decentralized control has also been implemented in regard to swarm control. Decentralized model predictive control (MPC) was applied to swarms of low-cost spacecraft with limited capabilities for swarm reconfiguration [22]. The benefit of this solution is that it decentralizes the computation and communication required for the swarm system. In [23], they used decentralized planning and sequential convex programming to control swarms. Using sequential convex programming in combination with MPC in real time provided robustness as the agents converged to designated targets. The same authors also used sequential convex programming to do target assignment (mapping of agents to targets) and trajectory generation for varying swarm sizes through time [24].

The objective of this paper is the formulation of the swarm estimation and control problems using RFS theory. The main contributions of this paper are as follows:

1) The generalization of the state representation using RFS theory for the control of large collaborative swarms under unknown number of agents.

2) The proposal of new distributional-based distances for the control cost function.

3) The "closing-the-loop" between RFS control and the PHD filter.

4) The application of multi-agent estimation and control using RFSs for formation flying of varying number of large collaborative swarms with the inclusion of process and measurement noise.

The first contribution is accomplished by representing the swarm state with an RFS, where RFSs are a collection of agent states, with no ordering between individual agents, that can randomly change through time [9]. For contribution two, several key divergence metrics are considered as control cost functions to drive the overall swarm behavior to a desired configuration. The third contribution is shown in Fig. 1. In Fig. 1, the first moment of the RFS models the current RFS swarm configuration, ν, and the desired RFS swarm configuration is defined by its first moment, ν_{des}. The first moment (or intensity discussed later on) contains information on the number of agents and their states. The PHD filter is used to process measurements from an unknown number of agents with defined spawn (Γ), birth (B), and death (D) rates, and the distributional distance-based cost "closes-the-loop" for RFS swarm control. Note that the number of agents is not a controlled quantity and in fact this work assumes that the number of agents is unknown and estimated under the RFS formulation. For the last contribution, convergent control solutions through MPC and differential dynamic programming (DDP) are found from GM-PHD filter estimates to advance control methods for swarm applications (e.g., Clohessy–Wiltshire relative motion), which offers

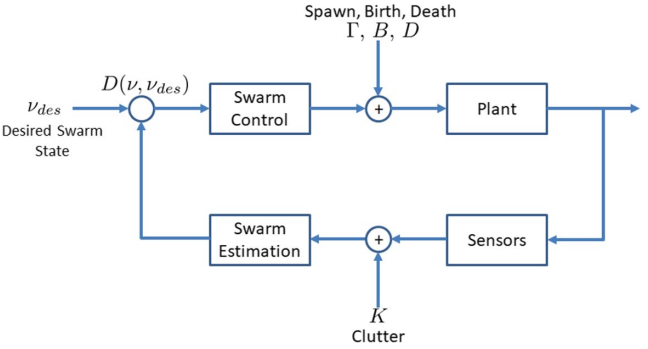


Fig. 1 A block diagram of the RFS control and estimation architecture in a closed loop.

improvements in computational efficiency and flexibility to uncertainty. Although the topology underlying the work is centralized, the formulation allows for a statistical model of the swarm and provides improvements in computational efficiency and flexibility (similar to the distributed optimal control methods [15–19]) when compared with traditional centralized methods [7]. Although not presented in this paper, the formulation in general can naturally be extended to decentralized control, which is expected to provide decentralized communication and computational efficiency.

The paper is organized as follows. Section II introduces preliminaries of the RFS theory relevant to this work. Section III presents the RFS-based control problem formulation, which is a central contribution of this work. Section IV proposes new distributional distance-based cost functions to form the RFS control problem. Section V discusses the dynamic models that will demonstrate the methods and solutions discussed in the previous section. Section VI discusses relevant simulation results involving the RFS control framework. Section VII provides limitations to the RFS control work presented. Lastly, in Sec. VIII, concluding remarks are provided.

II. Preliminaries

The swarm control approach developed in this work makes use of RFS theory. The motivation for the use of RFS theory stems from its application to multi-agent tracking [9,12]. We begin by presenting the single-agent and multi-agent tracking problems. Additionally, the multi-agent RFS-based tracking problem is used in this work to "close-the-loop" for control of large collaborative swarms.

The discrete dynamics and measurement model for a time-varying, single-agent system between discrete time steps k and $k+1$ is given by

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\epsilon}_k \tag{1a}$$

$$\boldsymbol{z}_k = H_k \boldsymbol{x}_k + D_k \boldsymbol{u}_k + \boldsymbol{\sigma}_k \tag{1b}$$

where $\boldsymbol{x}_k$ is the agent state, A_k is the system matrix, B_k is the control input matrix, $\boldsymbol{u}_k$ is the control input, $\boldsymbol{z}_k$ is the measurement vector, H_k is the observation matrix, and D_k is the feed-forward matrix. The process noise $\boldsymbol{\epsilon}_k$ and measurement noise $\boldsymbol{\sigma}_k$ are zero mean Gaussian noise with variances $\Sigma_{\boldsymbol{\epsilon}}$ and $\Sigma_{\boldsymbol{\sigma}}$, respectively. Note that a single agent, $\boldsymbol{x}_k$, is produced from a space $\mathcal{X} \subseteq \mathbb{R}^{d_x}$, where d_x is the agent state vector size. Similarly, the single agent's control input $\boldsymbol{u}_k$ is produced from a space $\mathcal{U} \subseteq \boldsymbol{R}^{d_u}$, where d_u is the agent's control vector size, and $\boldsymbol{z}_k$ is produced from a space $\mathcal{Z} \subseteq \mathbb{R}^{d_z}$, where d_z is the measurement vector size.

A. Single-Agent Filtering

To estimate the dynamics for a single-agent system, it is usually assumed that the state space follows a Markov process with a transition density,

$$f_{k|k-1}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{2}$$

which is the probability density for the single-agent system to move through its dynamics from $k-1$ to k. For generality, the dynamic system is partially observed as a likelihood function given by

$$g_k(\boldsymbol{z}_k|\boldsymbol{x}_k) \tag{3}$$

where the likelihood function is a probability density of observing the system by obtaining measurements, $\boldsymbol{z}_k$. By using the observation information from $\boldsymbol{z}_{1:k} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k)$, the posterior density estimate at a time k is determined using the Bayesian recursion given by

$$p_{k|k-1}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \int f_{k|k-1}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p_{k-1}(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})\, \mathrm{d}\boldsymbol{x}_{k-1} \tag{4a}$$

$$p_k(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{g_k(\boldsymbol{z}_k|\boldsymbol{x}_k) p_{k|k-1}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{\int g_k(\boldsymbol{z}_k|\boldsymbol{x}_k) p_{k|k-1}(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})\, \mathrm{d}\boldsymbol{x}_k} \tag{4b}$$

The posterior density contains the measurement update, and the estimate for this single-agent system can be found using a minimum mean-squared error method.

B. RFS Formulation

For the multi-agent tracking problem, a Bayesian recursion through an RFS formulation with discrete-time dynamics is considered [12]. This theory addresses the decentralized estimation problem for each agent in the formation. An ith agent in the swarm at time step k has the challenge of estimating its state configuration ($\boldsymbol{x}_k^i \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$) and designing a control policy to achieve that state. In this work, it is assumed that each agent within the swarm is identical, and using unique identifiers on each agent is unnecessary. Using this theory, the RFS models the uncertainty (i.e., the number of agents and their spatial states) by an RFS [12]. The agents in the field may die, survive, and move into the next state through dynamics, or appear by spawning or birthing. The unknown number of agents in the field is denoted by $N_{\text{total}}(k)$ and may be randomly varying at each time step by the union of the birth ($\Gamma_k: \varnothing \to \{\boldsymbol{x}_k^i, \boldsymbol{x}_k^{i+1}, \ldots, \boldsymbol{x}_k^{i+N_{\text{birth}(k)}}\}$), spawn ($B_{k|:k-1}(\boldsymbol{x}_{k-1}^i): \boldsymbol{x}_{k-1}^i \to \{\boldsymbol{x}_k^i, \boldsymbol{x}_k^{i+1}, \ldots, \boldsymbol{x}_k^{i+N_{\text{spawn}(k)}}\}$), and surviving ($S_{k|k-1}(\boldsymbol{x}_{k-1}^i): \boldsymbol{x}_{k-1}^i \to \boldsymbol{x}_k^i$) agents. Death is denoted by $D_k(\boldsymbol{x}_{k-1}^i)$: $\boldsymbol{x}_{k-1}^i \to \varnothing$. The number of births, $N_{\text{birth}(k)}$, and the number of spawns, $N_{\text{spawn}(k)}$, are unknown quantities that vary at each time step. The RFS, X_k, that describes the births, spawns, deaths, and surviving agents is given by

$$X_k = \left[\bigcup_{\boldsymbol{x}_{k-1}^i \in X_{k-1}} S_{k|k-1}(\boldsymbol{x}_{k-1}^i)\right] \cup \left[\bigcup_{\boldsymbol{x}_{k-1}^i \in X_{k-1}} B_{k|k-1}(\boldsymbol{x}_{k-1}^i)\right] \cup \Gamma_k \tag{5}$$

$X_k = \{\boldsymbol{x}_k^1, \boldsymbol{x}_k^2, \ldots, \boldsymbol{x}_k^{N_{\text{total}}(k)}\}$ denotes a realization of the RFS distribution for agents. The individual RFSs in Eq. (5) are assumed to be independent from each other. For example, any births that occur at any time step are independent from any surviving agents. At any time k, the RFS probability density function can be written as

$$p(X_k = \{\boldsymbol{x}_k^1, \boldsymbol{x}_k^2, \ldots, \boldsymbol{x}_k^n\}) = p(|X_k| = n)\, p(\{\boldsymbol{x}_k^1, \boldsymbol{x}_k^2, \ldots, \boldsymbol{x}_k^n\} || X_k| = n) \tag{6}$$

For a generalized observation process, the agents are either detected ($\Theta_k(\boldsymbol{x}_k^i): \boldsymbol{x}_k^i \to \boldsymbol{z}_k^i$) or not detected ($F_k(\boldsymbol{x}_k^i): \boldsymbol{x}_k^i \to \varnothing$). Clutter or false alarms ($K_k: \varnothing \to \{\boldsymbol{z}_k^1, \boldsymbol{z}_k^2, \ldots, \boldsymbol{z}_k^{N_{\text{clutter}}}\}$), defined as measurements that do not belong to any agents, are also present in the set of observations. For a time step k, note that $\boldsymbol{z}_k^i$ is the ith measurement obtained from a space $\mathcal{Z} \subseteq \mathbb{R}^{d_z}$. Therefore, RFS of measurements is described by

$$Z_k = K_k \cup \left[\bigcup_{\boldsymbol{x}_k^i \in X_k} \Theta_k(\boldsymbol{x}_k^i)\right] \tag{7}$$

where the origins of each measurement are not known and unique identifiers are not necessary. Again, the individual RFSs in Eq. (7) are independent of each other, so measurements and clutter are obtained independently from each other. With X_k and Z_k defined over sets of agents' states and measurements, a Bayesian recursion for multistate estimation can be applied.

On a similar note, the control sequence is also defined by an RFS in the form $U_k = \{\boldsymbol{u}_k^1, \boldsymbol{u}_k^2, \ldots, \boldsymbol{u}_k^{N_{\text{total}}(k)}\}$; and an RFS probability density given by

$$p(U_k = \{\boldsymbol{u}_k^1, \boldsymbol{u}_k^2, \ldots, \boldsymbol{u}_k^n\}) = p(|U_k| = n)\, p(\{\boldsymbol{u}_k^1, \boldsymbol{u}_k^2, \ldots, \boldsymbol{u}_k^n\} || U_k| = n) \tag{8}$$

because the realization of agents on the field to be controlled are varying with time k. Similarly, note that $\boldsymbol{u}_k^i$ is the ith agent's control input obtained from a space $\mathcal{U} \subseteq \mathbb{R}^{d_u}$.

The RFS formulation of describing multi-agent states and observations can be described very similarly to Eqs. (2) and (3) for single agent estimation, but the RFS states (X_k) and observations (Z_k) are used instead. To determine the multi-agent posterior density, a multi-agent Bayes recursion is used given by

$$p_{k|k-1}(X_k|Z_{1:k-1}) = \int f_{k|k-1}(X_k|X_{k-1}) p_{k-1}(X_{k-1}|Z_{1:k-1}) \mu_s(dX_{k-1}) \tag{9a}$$

$$p_k(X_k|Z_{1:k}) = \frac{g_k(Z_k|X_k) p_{k|k-1}(X_k|Z_{1:k-1})}{\int g_k(Z_k|X_k) p_{k|k-1}(X_k|Z_{1:k-1}) \mu_s(dX_k)} \tag{9b}$$

where μ_s is a reference measure on a collection of all finite subsets of state space [12]. From Eq. (9b), the integration occurs over all possible locations of agents residing in the state space as well as their number, which becomes a set integral. The recursion thus contains uncertainty in the agent number and location brought by detection uncertainty and measurement noise, respectively. Measurements are not a direct function of the individual agents due to explicitly incorporating clutter into the formulation. Therefore, measurement to agent assignment is not explicitly required in the formulation. By computing the set integral about all possible number of agents and their states, the recursion can become intractable, but solutions have been found for a small number of agents using sequential Monte Carlo [25]. Fortunately, a PHD filter approximation provides computational tractability for larger numbers of agents.

C. Probability Hypothesis Density Filter

Instead of propagating the multi-agent posterior density through a multi-agent Bayes recursion, the PHD filter propagates the posterior intensity function. The nonnegative intensity function $v(\boldsymbol{\xi})$ is a first-order statistical moment of the RFS state that represents the probability of finding an agent, represented by a generalized state variable $\boldsymbol{\xi} \in \mathcal{X}$, in a region of state space $\mathcal{S} \subseteq \mathcal{X}$. The estimated number of agents in the region $\mathcal{S}$ is the integral of the intensity function given by

$$\mathbb{E}(|X \cap \mathcal{S}|) = \int_{\mathcal{S}} v(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} \tag{10}$$

where the expectation represents an RFS X intersecting a region $\mathcal{S}$. This gives the total mass or the number of estimated agents for RFS X in a region $\mathcal{S}$. The local maximum in intensity $v(\boldsymbol{\xi})$ shows the highest concentration of expected number of agents, which can be used to determine an estimate for the agents in X at a time step.

To further interpret the intensity function, consider a one-dimensional example with four agents located with a mean and covariance of $\boldsymbol{m} = \{1, 4, 7, 11\}$ and $P^i = 1: i = 1, \ldots, 4$, respectively. This is a

realization of X. Assuming that the intensity function corresponds to a Gaussian mixture representation given by

$$\nu(\boldsymbol{\xi}) = \sum_{i=1}^{N_{\text{total}}} w^{(i)} \mathcal{N}(\boldsymbol{\xi}; \boldsymbol{m}^i, P^i) \tag{11}$$

where each weight $w^{(i)} = 1$, $\nu(\boldsymbol{\xi})$ can be plotted against the generalized state $\boldsymbol{\xi}$ given by Fig. 2. Note that this example is a specific case in which the number of agents is equal to the number of Gaussian mixtures by assuming $w^{(i)} = 1$, but in general, this is not the case. This is further discussed in the RFS Control Problem Formulation section. The agent locations ($X = \{1, 4, 7, 11\}$) are located at individual maxima of $\nu(\boldsymbol{\xi})$. The integral of $\nu(\boldsymbol{\xi})$ in Eq. (10) is

$$\begin{aligned}\mathbb{E}(|X \cap S|) = \int_S v(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} &= \int \mathcal{N}(\boldsymbol{\xi}; 1, 1)\, \mathrm{d}\boldsymbol{\xi} + \int \mathcal{N}(\boldsymbol{\xi}; 4, 1)\, \mathrm{d}\boldsymbol{\xi} \\ &\quad + \int \mathcal{N}(\boldsymbol{\xi}; 7, 1)\, \mathrm{d}\boldsymbol{\xi} + \int \mathcal{N}(\boldsymbol{\xi}; 11, 1)\, \mathrm{d}\boldsymbol{\xi} \\ &= 1 \quad + \quad 1 \quad + \quad 1 \quad + \quad 1 = 4\end{aligned} \tag{12}$$

which is the total mass or the total number of estimated agents of RFS X in this state space $\mathcal{S}$. It is also noted that the intensity function is not a probability density because the integral over $\boldsymbol{\xi}$ does not generally sum up into one. By estimating a potentially large, single intensity function, $\nu(\boldsymbol{\xi})$, an estimate of the number of agents and their states can be obtained.

An RFS that is fully characterized by their intensity is the Poisson RFS. By assuming that the RFS X is Poisson of the form $p(|X| = n)$ and $p(\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n\} | |X\} = n)$, approximate solutions can be determined by the PHD filter [9,12]. Propagation of the PHD can be determined if the agents are assumed to be independent and identical with the cardinality of the agent set that is Poisson distributed [12]. Clutter and birth RFSs are assumed to be Poisson RFSs. It is noted that the assumptions made by the PHD filter are strong assumptions for swarming robotics. However, this is a good starting point for an initial proof-of-concept study. For a time step k, the PHD recursion for a general intensity function $v_k(\boldsymbol{\xi})$ from the previous generalized state $\boldsymbol{\zeta} \in \mathcal{X}$ is given by

$$\bar{v}_k(\boldsymbol{\xi}) = b(\boldsymbol{\xi}) + \int p_s(\boldsymbol{\zeta}) f(\boldsymbol{\xi}|\boldsymbol{\zeta}) v(\boldsymbol{\zeta})\, \mathrm{d}\boldsymbol{\zeta} + \int \beta(\boldsymbol{\xi}|\boldsymbol{\zeta}) v(\boldsymbol{\zeta})\, \mathrm{d}\boldsymbol{\zeta} \tag{13}$$

where $b(\boldsymbol{\xi})$, $p_s(\boldsymbol{\zeta})$, and $\beta(\boldsymbol{\xi}|\boldsymbol{\zeta})$ are the agents' birth, survival, and spawn intensity, and $f(\boldsymbol{\xi}|\boldsymbol{\zeta})$ is the target motion model [12]. The bar on $\bar{v}_k(\boldsymbol{\xi})$ denotes that the PHD has been time-updated. For the measurement update, the equation is given by

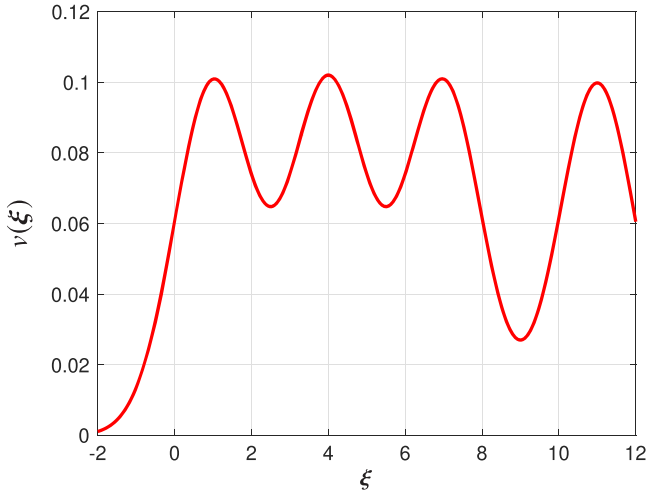


Fig. 2 The intensity for a 1D, 4-agent problem using Eq. (11).

$$v_k(\boldsymbol{\xi}) = (1 - p_d(\boldsymbol{\xi}))\bar{v}_k(\boldsymbol{\xi}) + \sum_{z \in Z_k} \frac{p_d(\boldsymbol{\xi}) g(z_k|\boldsymbol{\xi}) \bar{v}_k(\boldsymbol{\xi})}{c(z) + \int p_d(\boldsymbol{\zeta}) g(z_k|\boldsymbol{\zeta}) \bar{v}_k(\boldsymbol{\zeta})\, \mathrm{d}\boldsymbol{\zeta}} \tag{14}$$

where $p_d(\boldsymbol{\xi})$, $g(z_k|\boldsymbol{\xi})$, and $c(z)$ are the probability of detection, likelihood function, and clutter model of the sensor, respectively [12]. By using this recursion, the swarm probabilistic description can be updated. The recursion itself avoids computations that arise from the unknown relation between agents and its measurements, and that the posterior intensity is a function of the generalized state space. Unfortunately, Eqs. (13) and (14) do not contain a closed-form solution and the numerical integration suffers from higher computational time as the state increases due to an increasing number of agents.

III. Random Finite Set Control Problem Formulation

With the introduction of RFS theory from multi-agent tracking applications, a natural extension of RFS theory to the swarm control problem is appealing. We begin the discussion on how the swarm is modeled using the RFS intensity function and present the RFS control problem with the objective based on this model.

The PHD filter recursion given by Eqs. (13) and (14), as mentioned before, can be intractable as the state space increases. Fortunately, a closed-form solution exists if it is assumed that the survival and detection probabilities are state independent (i.e., $p_s(\boldsymbol{\xi}) = p_s$ and $p_d(\boldsymbol{\xi}) = p_d$), and the intensities of the birth and spawn RFSs are Gaussian mixtures initially presented in Eq. (11).

With the Gaussian mixture assumption, the current and desired intensities are defined as

$$\bar{\nu}(\boldsymbol{\xi}, k) \triangleq \sum_{i=1}^{N_f} w_f^{(i)} \mathcal{N}(\boldsymbol{\xi}; \boldsymbol{m}_f^i, P_f^i) = \nu_b(\boldsymbol{\xi}, k) + \nu_{p_s}(\boldsymbol{\xi}, k) + \nu_\beta(\boldsymbol{\xi}, k) \tag{15}$$

$$\nu_{\text{des}}(\boldsymbol{\xi}, k) \triangleq g(\boldsymbol{\xi}) \triangleq \sum_{i=1}^{N_g} w_g^{(i)} \mathcal{N}(\boldsymbol{\xi}; \boldsymbol{m}_g^i, P_g^i) \tag{16}$$

where $w^{(i)}$ are the weights and $\mathcal{N}(\boldsymbol{\xi}; \boldsymbol{m}^i, P^i)$ is the probability density function of an ith multivariate Gaussian distribution with a mean and covariance corresponding to the peaks and spread of the intensity, respectively. The terms N_f and N_g are the total number of multivariate Gaussian distributions in the current and desired intensities, respectively. It is assumed that the desired Gaussian mixture intensity, $\nu_{\text{des}}(\boldsymbol{\xi}, k)$, is known. Equation (15) includes the summation of the individual birth ($\nu_b(\boldsymbol{\xi}, k)$), spawn ($\nu_\beta$), and survival ($\nu_{p_s}(\boldsymbol{\xi}, k)$) Gaussian mixture intensities, which simplify to another Gaussian mixture. Note that closed-form solutions using Gaussian mixtures exist for cases without the state-independent assumption. Additionally, $\sum_{N_f}^{i=1} w_f^{(i)} = N_{\text{total}}(k)$ and $\sum_{i=1}^{N_g} w_g^{(i)} = \bar{N}_{\text{total}}(k)$, where $\bar{N}_{\text{total}}(k)$ is the desired number of agents. The current and desired intensity functions $\nu(\boldsymbol{\xi}, k)$ and $\nu_{\text{des}}(\boldsymbol{\xi}, k)$ are in terms of the agents' state. The swarm intensity function can be propagated through updates on the mean and covariance of the Gaussian mixtures as given by

$$\boldsymbol{m}_{f,k+1}^i = A_k \boldsymbol{m}_{f,k}^i + B_k \boldsymbol{u}_{f,k}^i \tag{17}$$

$$P_{f,k+1}^i = A_k P_{f,k}^i A_k^T + \Sigma_\epsilon \tag{18}$$

The agents' states $\boldsymbol{x}$ are incorporated in the mean and covariance of the Gaussian mixture intensity. Then given the Gaussian mixture intensities assumption, a control variable is calculated for each component $\boldsymbol{u}_{f,k}^i$. Additionally, each Gaussian mixture component may represent many agents because the intensity function integrates to the total number of agents, and the number of agents on the field is found using $\sum_{i=1}^{N_f} w_f^{(i)} = N_{\text{total}}(k)$. So control is directly applied to the Gaussian mixture, which may represent single or multiple agents. The weights

can be scaled as the number of agents in the swarm change while not affecting the optimal solution found for swarm control. This detail is emphasized in the Results section. Note that although linear dynamics are used, the dynamics can be modeled as a nonlinear function of the state.

The measurement update is also closed form given by the intensity

$$\nu_k(\boldsymbol{\xi},k) = f(\boldsymbol{\xi}) = (1-p_d(\boldsymbol{\xi}))\bar{\nu}_k(\boldsymbol{\xi}) + \sum_{z\in Z_k}\sum_{j=1}^{N_f} w_k^{(j)}\mathcal{N}\left(\boldsymbol{\xi}; \boldsymbol{m}_{k|k}^{(j)}(z), P_{k|k}^{(j)}\right) \tag{19}$$

where

$$w_k^{(j)} = \frac{p_d(\boldsymbol{\xi}) w_f^{(j)} q^{(j)}(z)}{K(z) + p_d(\boldsymbol{\xi}) \sum_{l=1}^{N_f} w_f^{(l)} q^{(l)}(z)} \tag{20a}$$

$$\boldsymbol{m}_{k|k}^{(j)}(z) = \boldsymbol{m}_f^{(j)} + K^{(j)}\left(z - H_k \boldsymbol{m}_f^{(j)}\right) \tag{20b}$$

$$P_{k|k}^{(j)} = \left(I - K^{(j)} H_k\right) P_f^i \tag{20c}$$

$$K^{(j)} = P_f^i H_k^T \left(H_k P_f^i H_k^T + \Sigma_{\boldsymbol{\sigma}}\right)^{-1} \tag{20d}$$

$$q_k^{(j)}(z) = \mathcal{N}\left(z; H_k \boldsymbol{m}_f^{(j)}, \Sigma_{\boldsymbol{\sigma}} + H_k P_f^i H_k^T\right) \tag{20e}$$

which follow closely to the Kalman filter measurement update equations. Using RFS theory, it is assumed that the individual swarm agents form a Gaussian mixture intensity function in which the means and covariances of the Gaussian mixture are propagated and controlled. An optimal control problem is defined by minimizing the swarm control effort and distance from the desired swarm formation. The objective function for this optimal control problem is defined as

$$J(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T) = \sum_{k=1}^{T} \boldsymbol{u}_k^T R \boldsymbol{u}_k + D(\nu(\boldsymbol{\xi},k), \nu_{\text{des}}(\boldsymbol{\xi},k)) \tag{21}$$

where $\nu_{\text{des}}(\boldsymbol{\xi},k)$ is the desired formation, R is the positive definite control weight matrix, and the $\boldsymbol{u}_k$ are the control effort for the Gaussian mixture intensities shown in Eq. (17). Both $\nu(\boldsymbol{\xi},k)$ and $\nu_{\text{des}}(\boldsymbol{\xi},k)$ are defined over the complete state space that include position and velocity parameters. The distance between Gaussian mixtures, $D(\cdot,\cdot)$, has several closed-form solutions, and it has been used previously to define an objective function for path planning of multi-agent systems [19].

The key feature of the RFS control problem is that it can allow for a unified representation for swarming systems. This unified representation is achieved by minimizing the RFS objective function, Eq. (21), about the swarm intensity statistics given by Eqs. (17) and (18). Thus, it can handle multifidelity swarm localization and control between the current and desired spatial distributions. The swarm is treated probabilistically, and the bulk motion is modeled, which allows the theory to handle large numbers of indistinguishable units with unknown swarm size. This reduces the dimensionality of the state while enabling complex behavior. Naturally, the RFS control problem is formulated to enable complex decision making through RFS theory.

Two different scenarios can be applied by "closing-the-loop" between RFS control and the PHD filter shown in Fig. 1. Specifically, a single observer can be used to estimate the entire state of the swarm by collecting measurements of each agent in the field. This provides a centralized approach to obtaining estimates and controlling the swarm through RFSs. The other option is to run a local PHD observer on each agent to estimate the state of the swarm. In this case, the observer is limited to an agent's field of view, but it is able to make localized or decentralized control decisions using RFS. For this work, centralized RFS control is explored by using the complete topology obtained from the PHD filter.

IV. Distributional Distance-Based Cost

The control objective for the RFS formulation of agents with an unknown distance between the intensities is provided by Eq. (21). The distance metric can be defined using several closed-form solutions for Gaussian mixtures. Then, the corresponding optimal control problem is formulated using several closed-form methods discussed in the next section.

A. Cauchy–Schwarz Divergence

The Cauchy–Schwarz divergence is based on the Cauchy–Schwarz inequality for inner products of RFS, and it is defined for two RFS with intensities f and g given by

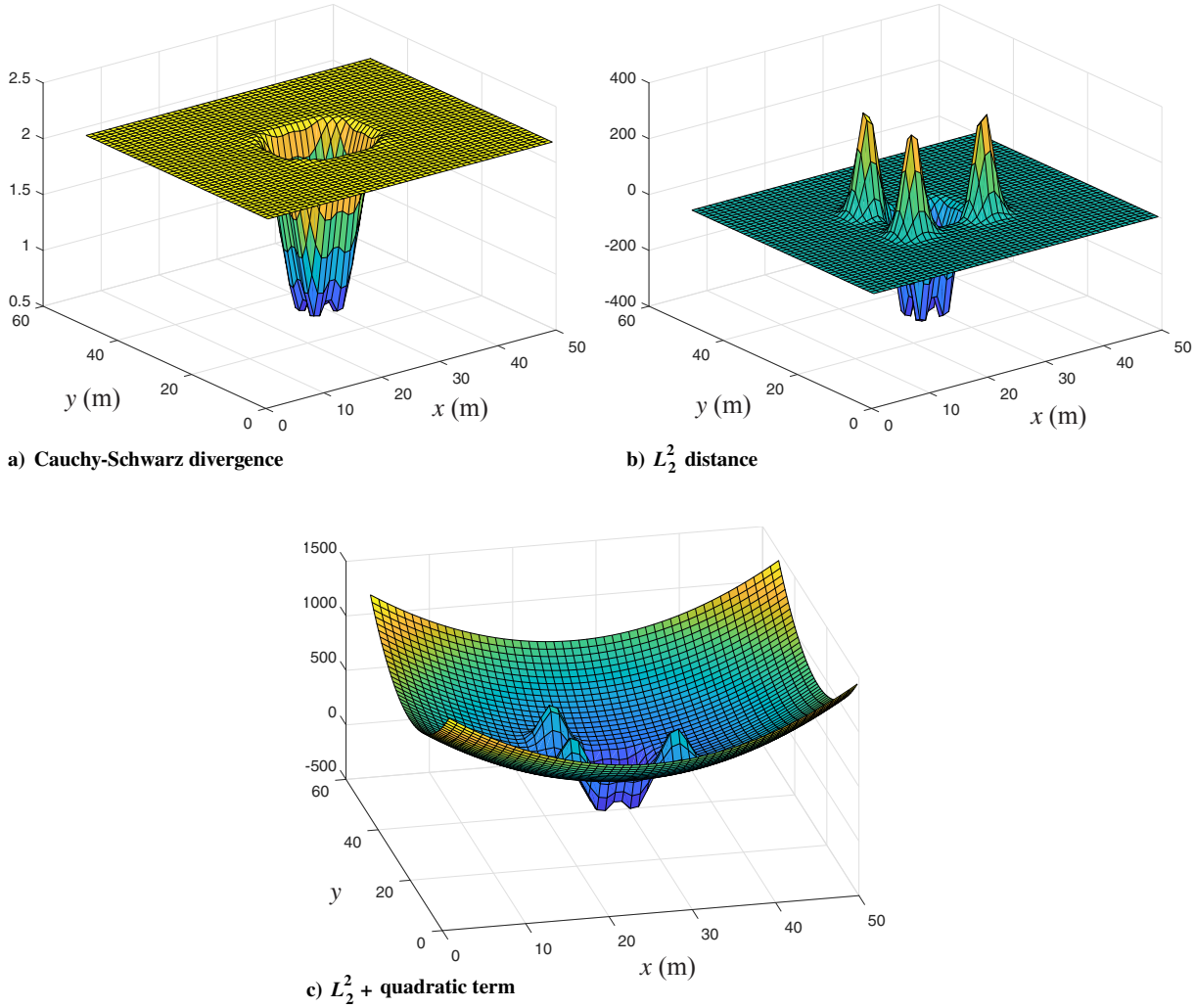
$$D_{CS}(f,g) = -\ell_n\left(\frac{\langle f,g\rangle}{\|f\|\|g\|}\right) \tag{22}$$

where $\langle r(\boldsymbol{\xi}), t(\boldsymbol{\xi})\rangle \triangleq \int r(\boldsymbol{\xi}) t(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi}$ is the L_2^2 inner product over the generalized RFS intensities $r(\boldsymbol{\xi})$ and $t(\boldsymbol{\xi})$ [26]. The argument of the logarithm is nonnegative because probability densities are nonnegative, and it does not exceed one by the Cauchy–Schwarz inequality. The Cauchy–Schwarz divergence can be interpreted as an approximation to the Kullback–Leibler divergence but has a closed-form expression for Gaussian mixtures [26]. This is useful for calculating the distance between two-point processes represented by intensity functions. By substituting the intensities from Eqs. (19) and (16) for f and g, respectively, the Cauchy–Schwarz divergence between two Poisson point processes with Gaussian mixture intensities, $D_{CS}(f,g)$, is simplified to

$$\begin{aligned} D_{CS}(f,g) = {} & \frac{1}{2}\ell_n\left(\sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_f^{(j)} w_f^{(i)} \mathcal{N}\left(\boldsymbol{m}_f^j; \boldsymbol{m}_f^i, P_f^i + P_f^j\right)\right) \\ & + \frac{1}{2}\ell_n\left(\sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_g^{(j)} w_g^{(i)} \mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_g^i, P_g^i + P_g^j\right)\right) \\ & - \ell_n\left(\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_f^i, P_g^i + P_f^j\right)\right) \end{aligned} \tag{23}$$

Note that in the control formulation used, only $\nu(\boldsymbol{\xi},k)$ is assumed to depend on the control $\boldsymbol{u}$. Therefore, the term that depends only on $\nu_{\text{des}}(\boldsymbol{\xi},k)$ is omitted from the objective function since $\nu_{\text{des}}(\boldsymbol{\xi},k)$ does not depend on $\boldsymbol{u}$.

Figure 3a shows the surface plot using the Cauchy–Schwarz divergence for four Gaussian mixtures in the swarm at an initial time instance that designates the distributional distance-based cost of the objective function. The four Gaussian mixtures start with initial conditions of $(\pm 3, \pm 3)$ in a square grid. The desired intensity is set as $(\pm 1, \pm 1)$ in a square grid. From the surface plot, each initial intensity has hills and the desired intensity has valleys. The goal is to minimize the objective function; thus, an optimization method (e.g., the quasi-Newton method) determines a control solution that minimizes the objective. Because the desired intensity in Fig. 3a is located at a minimum in the objective surface plot, the optimization method finds a control input to move toward that point. The opposite occurs with the hills (current intensity). The minimization finds a control solution that moves away from the hills, and thus gives individual current Gaussian mixtures collision avoidance attributes. Therefore in the minimization of the objective function, each Gaussian mixture will repel each other while moving toward the desired Gaussian mixtures through time. Although the Cauchy–Schwarz divergence has a repelling effect, collision avoidance is not guaranteed, but the distance does encourage collision-reducing trajectory solutions. If the initial intensity is too large compared with the desired intensity, it will take longer for the four Gaussian mixtures to converge to the desired values or diverge due to the optimization



a) Cauchy-Schwarz divergence

b) L_2^2 distance

c) L_2^2 + quadratic term

Fig. 3 a–c) Surface plots with the corresponding distributional distance-based costs. The current and desired intensity are initialized at (±3, ±3) and (±1, ±1), respectively.

getting stuck in local minima (the flat plane). Also, the repelling effect due to the hills are relatively small. Thus, the Cauchy–Schwarz divergence may not be the fastest converging solution for the objective function minimization.

B. L_2^2 Distance

Alternatively, the distance between two Poisson point processes with Gaussian mixture intensities can be determined by using the L_2^2 distance between the intensities. The L_2^2 distance is given by

$$D_{L_2^2}(f,g) = \int (f-g)^2 \,\mathrm{d}\boldsymbol{\xi} = \|f-g\|^2 \tag{24}$$

where the close-form solution for Gaussian mixture intensities is simplified to

$$\begin{aligned} D_{L_2^2}(f,g) = &\sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_f^{(j)} w_f^{(i)} \mathcal{N}\left(\boldsymbol{m}_f^j; \boldsymbol{m}_f^i, P_f^i + P_f^j\right) \\ &+ \sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_g^{(j)} w_g^{(i)} \mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_g^i, P_g^i + P_g^j\right) \\ &- 2\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_f^i, P_g^i + P_f^j\right) \end{aligned} \tag{25}$$

The L_2^2 distance is stationary (i.e., gradients are zero) when intensities f and g are equal. That is, the cost is minimum when the target g is reached from any intensity f.

The L_2^2 distance follows the property of the Bregman divergence, which has an additional property of convexity [27]. The distance, given by

$$D_F(f,g) = F(f) - F(g) - \langle \nabla F(g), f-g \rangle \tag{26}$$

is convex if $F(\cdot)$ is strictly convex and continuously differentiable on a closed convex set [27]. Strictly convex functions are listed in [27]. For this work, the squared Euclidean distance $F(f) = f^2$ was used to generate the Bregman divergence given by

$$D_F(f,g) = \langle f,f \rangle + \langle g,g \rangle - 2\langle f,g \rangle \tag{27}$$

which is in the same exact form of Eq. (25). Figure 3b shows the surface plot using the L_2^2 distance for a 4 Gaussian mixture swarm for the same example as the Cauchy–Schwarz divergence. The initial intensity has more defined hills compared with the Cauchy–Schwarz divergence. Thus, the initial Gaussian mixtures have a stronger repelling effect upon one another. Also, the desired Gaussian mixtures have large valleys that create a large attraction effect for each initial Gaussian mixture to move to. Thus, the optimization solution will be faster in the L_2^2 distance case. Unfortunately, the L_2^2 distance suffers from a similar issue to the Cauchy–Schwarz divergence. If the initial conditions increase farther away from the desired intensity, the optimization may take much longer or get stuck in local minima due to a flat surface away from the desired intensity.

C. L_2^2 Distance with Quadratic Term

The issue of convergence remains for the L_2^2 distance when the initial states are farther away from the desired intensity. To achieve faster convergence, an additional term is added to the L_2^2 distance to shape the gradient descent through a quadratic term as given by

$$D_{L_2^2\text{mod}}(f,g) = D_{L_2^2}(f,g) - \alpha \sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)}\, \ell_n\left(\mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_f^i, P_g^i + P_f^j\right)\right) \tag{28}$$

where α is a fixed or changing parameter. Unfortunately, adding the quadratic term to the L_2^2 distance does not make the objective function stationary at $f = g$. To alleviate this issue, the α parameter is included with the quadratic term to relax the contribution of the gradient to the L_2^2 stationary point. By substituting Eq. (25) into Eq. (28), the equation becomes

$$\begin{aligned} D_{L_2^2\text{mod}}(f,g) = {} & \sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_f^{(j)} w_f^{(i)} \mathcal{N}\left(\boldsymbol{m}_f^j; \boldsymbol{m}_f^i, P_f^i + P_f^j\right) \\ & + \sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_g^{(j)} w_g^{(i)} \mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_g^i, P_g^i + P_g^j\right) \\ & - 2\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)} \mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_f^i, P_g^i + P_f^j\right) \\ & - \alpha \sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_g^{(j)} w_f^{(i)}\, \ell_n\left(\mathcal{N}\left(\boldsymbol{m}_g^j; \boldsymbol{m}_f^i, P_g^i + P_f^j\right)\right) \end{aligned} \tag{29}$$

Note that this term is referred as quadratic, although it may be more appropriate to call it quadratic-like. Figure 3c shows the surface plot using Eq. (29) for the same 4 Gaussian mixture swarm used in the Cauchy–Schwarz divergence. Compared with the L_2^2 distance, the initial and desired intensities provide the hills and valleys necessary to obtain convergence. However, as the initial intensity move outward, the surface map decreases in a quadratic fashion instead of staying flat. This prevents the optimization from converging to a local minima. Instead, the additional quadratic term allows convergence to the desired intensity (global minima). Thus, the optimization can occur at any point to reach convergence.

Traditional LQR-based solutions are not applicable to the minimization of the objective function, Eq. (29), because the L_2^2 terms are nonquadratic [28]. The minimization of the objective function in discrete time is

$$\begin{aligned} & \min_{\boldsymbol{u}_k, k=1,\ldots,T} J(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_T) \\ & = \sum_{k=1}^{T} \boldsymbol{u}_k^T R \boldsymbol{u}_k + \sum_{j=1}^{N_f}\sum_{i=1}^{N_f} w_{f,k}^{(j)} w_{f,k}^{(i)} \mathcal{N}\left(\boldsymbol{m}_{f,k}^j; \boldsymbol{m}_{f,k}^i, P_{f,k}^i + P_{f,k}^j\right) \\ & + \sum_{j=1}^{N_g}\sum_{i=1}^{N_g} w_{g,k}^{(j)} w_{g,k}^{(i)} \mathcal{N}\left(\boldsymbol{m}_{g,k}^j; \boldsymbol{m}_{g,k}^i, P_{g,k}^i + P_{g,k}^j\right) \\ & - 2\sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_{g,k}^{(j)} w_{f,k}^{(i)} \mathcal{N}\left(\boldsymbol{m}_{g,k}^j; \boldsymbol{m}_{f,k}^i, P_{g,k}^i + P_{f,k}^j\right) \\ & - \alpha \sum_{j=1}^{N_g}\sum_{i=1}^{N_f} w_{g,k}^{(j)} w_{f,k}^{(i)}\, \ell_n\left(\mathcal{N}\left(\boldsymbol{m}_{g,k}^j; \boldsymbol{m}_{f,k}^i, P_{g,k}^i + P_{f,k}^j\right)\right) \end{aligned} \tag{30}$$

$$\begin{aligned} \text{Subject to: } & \boldsymbol{m}_{f,k+1}^i = A_k \boldsymbol{m}_{f,k}^i + B_k \boldsymbol{u}_{f,k}^i, \\ & P_{f,k+1}^i = A_k P_{f,k}^i A_k^T + \Sigma_{\boldsymbol{e}} \end{aligned} \tag{31}$$

where $\boldsymbol{u}_k = [(\boldsymbol{u}_{f,k}^1)^T, \ldots, (\boldsymbol{u}_{f,k}^{N_f})^T]^T$ is the collection of all control variables. Therefore, control solutions are found by either using DDP, where the objective function is quadratized by taking a Taylor series approximation about a nominal trajectory, or using optimization techniques (e.g., the quasi-Newton method), where the nonquadratic objective function is used directly to find an optimal control solution. The number of agents is not a controlled quantity and can be normalized out of the objective function [this can be shown by examining Eq. (30)]. In this work we do not desire to control the number of agents but rather control individual agents to track the desired concentration or distribution of agents. Under the proposed formulation, the distribution of agents is indeed a controllable quantity using the control inputs defined in Eq. (30).

To obtain efficient control solutions for large time horizons, MPC is used directly with the control techniques. Quasi-Newton handles the nonlinearities in the objective function, and it provides an initial basis in comparing the time-history responses for RFS control using different distributional distance-based costs. Next, RFS control is extended to MPC with DDP, which approximates (quadratizes) the objective function for value iteration to provide quick and reliable convergence to locally optimal control solutions. DDP and MPC are discussed in the Appendix. The RFS control solution is demonstrated on spacecraft swarm relative motion simulation with and without perfect information combining the GM-PHD filter and DDP (formed as iterative linear quadratic regulator [ILQR]) in a closed-loop fashion given in Fig. 1. In a single loop, the RFS control is determined from optimizing the objective containing the distributional distance-based cost between the estimate and the desired intensity; the swarm dynamics are updated with new spawn, birth, or death of agents in the field; and measurements including clutter are incorporated into the overall system before a GM-PHD filter estimate is determined for control again. From this RFS-based architecture, the ability to determine an estimate of the cardinality and states of the swarm that is used directly for control using ILQR is realized.

As previously discussed for this work, the topology underlying the RFS control is complete and uses the complete graph in a centralized manner. Thus, the computational load for control of the swarm is centralized. It is computationally feasible to perform centralized control from a single agent, although a separate ground station may be necessary to perform difficult control computations.

V. Dynamic Models

To show viability of optimal swarm control via RFS, an acceleration model and a relative motion model, both linear systems, are used to describe rover and satellite dynamics, respectively. The dynamic equations of individual agents are used here to describe the dynamics of the Gaussian mixture components (means) given by the control objective Eqs. (30) and (31). Because linear dynamics are used, the DDP term can be expressed as ILQR.

A. Acceleration Model

On a 2D plane, the linear time-invariant system of each agent can be described by the continuous state and control matrices

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{32}$$

and a state vector $\boldsymbol{x} = [x, y, \dot{x}, \dot{y}]^T$. Both x and y are defined to be the 2D positions of the agent, respectively. The A_c and B_c matrices are discretized along a fixed time interval using a zero-order hold assumption for the control (i.e., control is held constant over the time interval). This results in discretized A and B matrices for the state-space equation

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k \tag{33}$$

B. Relative Motion Using Clohessy–Wiltshire Equations

For a spacecraft in low Earth orbit, the relative dynamics of each spacecraft (agent), to a chief spacecraft in circular orbit, is given by the Clohessy–Wiltshire equations [29]

$$\ddot{x} = 3n^2x + 2n\dot{y} + a_x \tag{34a}$$

$$\ddot{y} = -2n\dot{x} + a_y \tag{34b}$$

$$\ddot{z} = -n^2z + a_z \tag{34c}$$

where x, y, and z are the relative positions in the orbital local-vertical local-horizontal (LVLH) frame and a_x, a_y, and a_z are the accelerations in each axis, respectively. The variable n is defined as the orbital frequency given by

$$n = \sqrt{\frac{\mu}{a^3}} \tag{35}$$

where μ is the standard gravitational parameter and a is the radius of the circular orbit. The continuous state-space representation is given by

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix}, \qquad B_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{36}$$

with a state vector $\boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$ and a control input $\boldsymbol{u} = [a_x, a_y, a_z]^T$. These equations are discretized similarly to the acceleration model discussed previously.

VI. Results

The first goal is to generalize RFSs for control of large collaborative swarms to form and test behaviors of several different RFS-based distance measures for control. Using the acceleration model, which is discretized from Eq. (32) to Eq. (33), a 4 Gaussian mixture swarm on a 2D plane is initialized in a square grid where the mixtures 1, 2, 3, and 4 are defined counterclockwise starting on the first quadrant. With the 4 Gaussian mixture swarm, three different test cases are implemented to bring the intensity to the target trajectories and to test the distributional distance-based costs and control theory involved from the RFS formulation. The first test case compares the L_2^2 distance with varying initial conditions in a square grid with the L_2^2 plus quadratic distance with four desired Gaussian mixtures located at $(\pm 1, \pm 1)$ using quasi-Newton MPC. An L_2^2 plus quadratic distance comparison is also done using ILQR. The last two cases present the quasi-Newton MPC and ILQR control using the L_2^2 distance with a quadratic term and varying desired Gaussian mixtures. For case 2, three target destinations are located at $(\pm 1, 1)$ and $(-1, -1)$. Lastly, in case 3, five target destinations are located at $(\pm 1, \pm 1)$ and $(0, 0)$.

The second goal is to form and apply multi-agent estimation and control of large collaborative swarms in the presence of unknown number of agents, clutter, and noise using RFS theory. This is applied directly to the satellite relative motion problem using the Clohessy–Wiltshire equations. Specifically, the L_2^2 plus quadratic distance is used for spacecraft formation flight. A 77 Gaussian mixture swarm is initialized uniformly random between -1 and 1 on a 2D plane. Assuming that the spacecraft swarm is at lower Earth orbit, the goal for the spacecraft is to track a rotating star pattern moving counterclockwise at an orbital frequency of n.

For these two separate examples, it should be emphasized that each Gaussian mixture component may represent many agents because the intensity function integrates to the total number of agents. For the initial example, many agents are represented by each Gaussian mixture, and the number of agents on the field must be calculated using $\sum_{i=1}^{N_f} w_f^{(i)} = N_{\text{total}}(k)$. So individual agents follow the control law that is applied to their specific Gaussian mixture. As mentioned before, the weights of the desired RFS do not affect the optimal solution, and therefore, for simplicity, this initial goal uses weights of $w_k = 1$ for all k. For the second example, which incorporates the Clohessy–Wiltshire relative motion model, the perfect and imperfect information scenarios assume that the number of agents is equal to the number of Gaussian mixtures (i.e., $w_f^{(i)} = 1$ for all k) and the number of agents is estimated from the GM-PHD filter at each time step k, respectively. Thus, the individual agents are controlled directly.

A. Acceleration Model

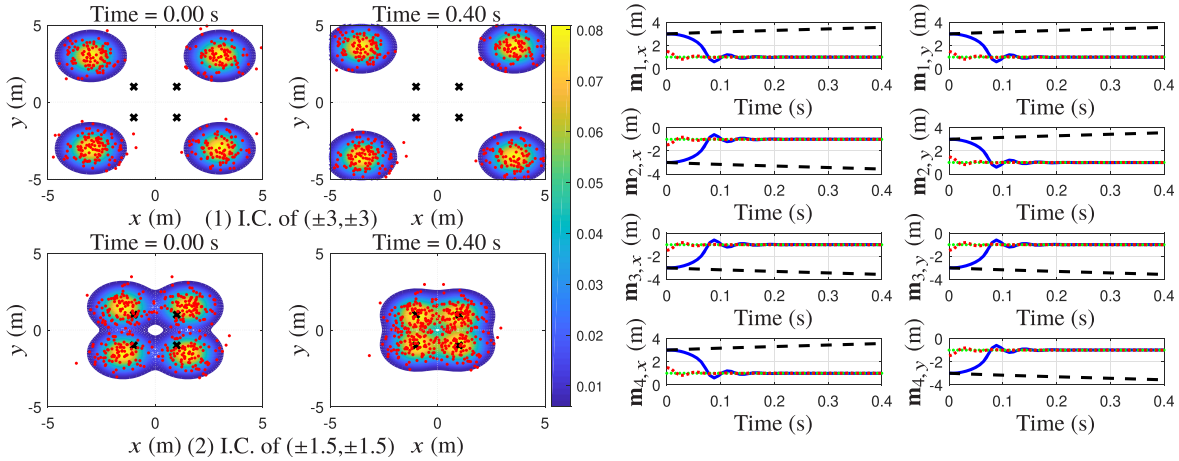
1. Case 1: L_2^2 Versus L_2^2+Quadratic Term, Four Desired Gaussian Mixtures

For case 1, four swarm Gaussian mixtures are controlled to move toward the desired intensity at initial conditions farther away (square grid at $(\pm 3, \pm 3)$) and closer to (square grid at $(\pm 1.5, \pm 1.5)$) the desired intensity as shown by mean responses given by the black-dashed and red-dotted lines in Fig. 4b, respectively. From the trajectory snapshots given by Fig. 4a1, initial conditions that are far from the desired intensity do not have a converging control solution. From the surface visualization in Fig. 3b, the general plane is flat in areas away from the desired targets and the current agents' states. Therefore, optimization using quasi-Newton MPC is more difficult in these flat areas and may not converge to a solution. If the current intensity is initialized much closer to the desired intensity as shown in Fig. 4a2, the flatness in the general plane is minimal, and the optimization step in quasi-Newton MPC converges to a solution. By using the L_2^2 distance, converging control solutions can only be found for initial conditions and target destinations that are close. For the L_2^2 plus quadratic distance, four swarm Gaussian mixtures move toward the four desired Gaussian mixtures given by the blue-solid lines (mean responses) in Fig. 4b. Figure 4c shows the trajectory snapshots and final states of each of the swarm Gaussian mixtures during the simulation. The target destinations are plotted as black x's. The red dots are the individual swarm agents that form the Gaussian mixture intensities. From the figure, all four mixtures converge to the desired mixtures in approximately 0.17 s and approximately 0.03 of steady-state error between the mixtures' position to the desired intensity. In comparison to only the L_2^2 distance, Fig. 4b shows that for small distances between the initial state and the desired intensity, the L_2^2 distance is sufficient for state convergence, but as the distance increases, the L_2^2 distance diverges away. By adding the quadratic term to L_2^2, the optimization step can directly determine the minimum for the control solution shown in Fig. 3c. Therefore, the desired intensity attracts the current swarm intensity at distances that fail for only L_2^2 distance given by Fig. 4b.

The L_2^2 plus quadratic distance is also extended to ILQR. Figure 5a shows the trajectory snapshots and final states of the simulation. All four Gaussian mixtures converge to the desired intensity in approximately 0.03 s and approximately 0.01 of steady-state error as shown in Fig. 5b. In this figure, the x responses, y responses, and the desired intensity are given by blue, green, and red lines, respectively. The entire simulation horizon is used to provide the prediction horizon for the ILQR trajectory. Even with a quadratic approximation of the objective function, ILQR is able to find control solutions that follow the L_2^2 plus quadratic characteristics that are presented using quasi-Newton MPC.

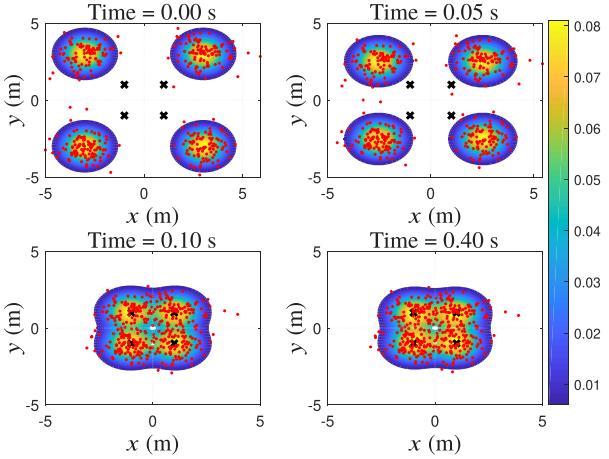
2. Case 2: Three Desired Gaussian Mixtures

Case 2 illustrates the effect of three desired Gaussian mixtures on the final trajectories of the four swarm Gaussian mixtures using quasi-Newton MPC and ILQR. Using quasi-Newton MPC, the current swarm intensity converges as given by the position time history in Fig. 6b. The trajectories for mixture 1 and mixture 3 reach their target, but mixtures 2 and 4 reach the third target with approximately 0.42 and 0.50 of steady-state error with 0.20 and 0.16 s of settling time, respectively. From Fig. 6a, it can be visually shown where the



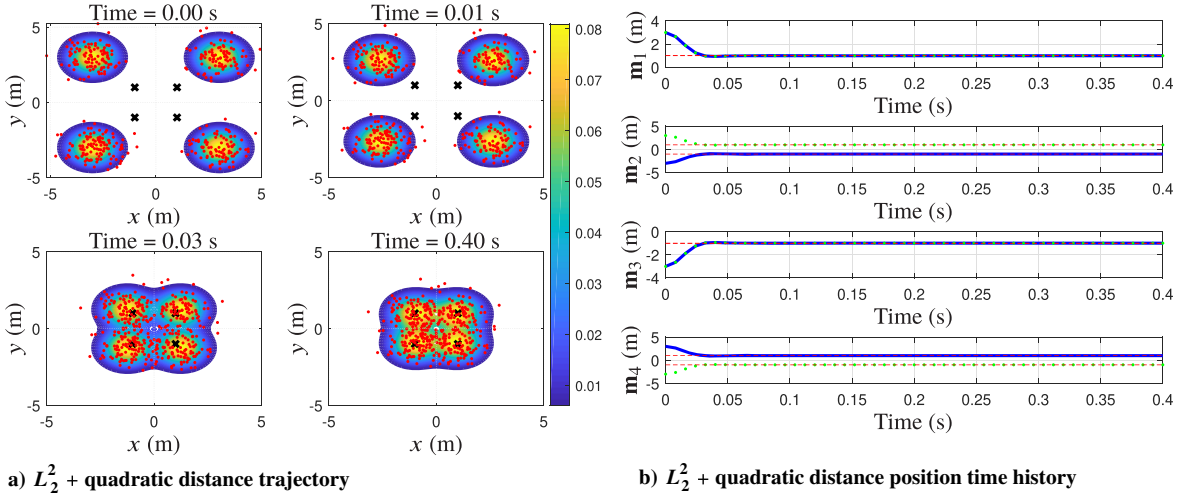
a) L_2^2 distance trajectory

b) L_2^2, L_2^2 + quadratic distance position time history



c) L_2^2 + quadratic distance trajectory

Fig. 4 Case 1: a, c) controlled trajectories using the acceleration model and quasi-Newton MPC; b) intensity mean responses from the trajectories.

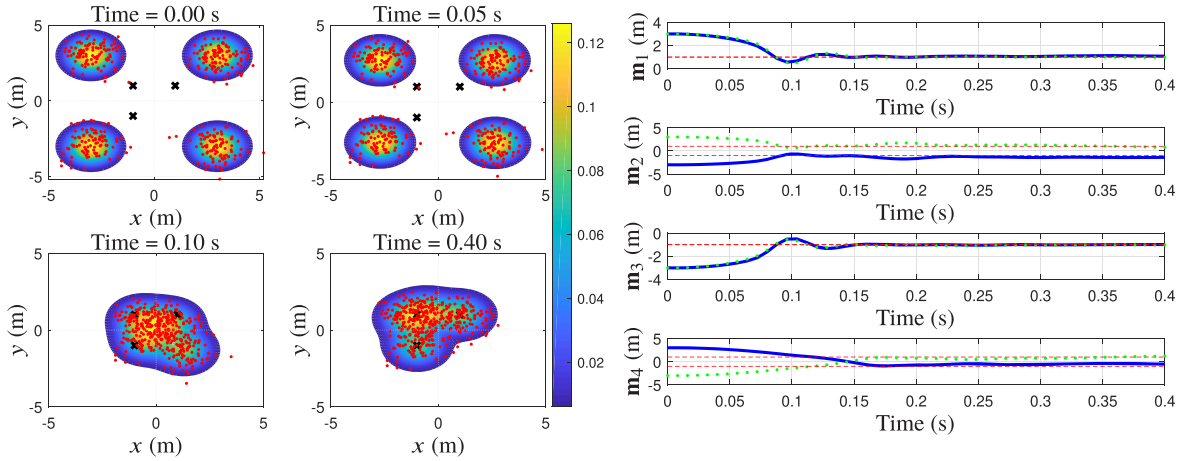


a) L_2^2 + quadratic distance trajectory

b) L_2^2 + quadratic distance position time history

Fig. 5 Case 1: 4 Gaussian mixture swarm controlled to four desired Gaussian mixtures via ILQR. a) Trajectories for the swarm; b) position time history.

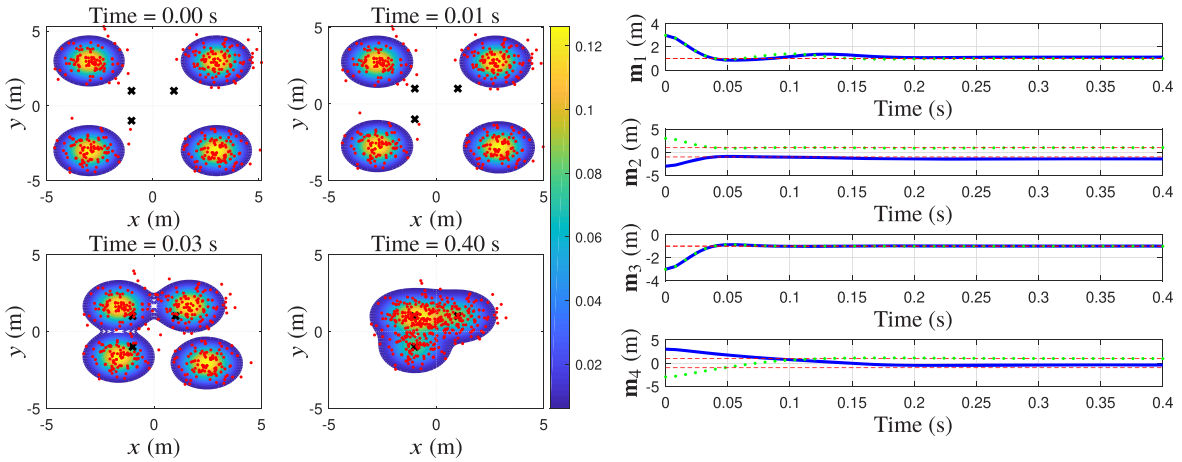
swarm intensity is located relative to the desired intensity at each time step. The results obtained follow directly from the RFS control theory using the L_2^2 plus quadratic distance term. By using this L_2^2 with a quadratic term in the objective function, the current intensity will attract toward the desired intensity while repulsing away from each other. This can be seen in the surface map shown in Fig. 3c, where the hills are areas of repulsion and valleys are areas of attraction. Thus, for quasi-Newton MPC, mixtures 2 and 4 are attracted to the same target, but they stay away from each other. This case is also extended to ILQR. Figures 7a and 7b show the trajectory snapshots and time history of the same swarm using ILQR. As discussed previously, due to the approximation of the objective function, the mixtures 2 and 4 converged in 0.03 and 0.15 s with approximately 0.01 and 0.42 of steady-state error. By comparing the time histories in Figs. 7b and 6b, the fourth intensity using ILQR follows very similarly to the MPC method. Therefore, there is a degree of accuracy in the approximation



a) L_2^2 + quadratic distance trajectory

b) L_2^2 + quadratic distance position time history

Fig. 6 Case 2: 4 Gaussian mixture swarm controlled to three desired Gaussian mixtures via quasi-Newton MPC. a) Trajectories for the swarm; b) position time history.



a) L_2^2 + quadratic distance trajectory

b) L_2^2 + quadratic distance position time history

Fig. 7 Case 2: 4 Gaussian mixture swarm controlled to three desired Gaussian mixtures via ILQR. a) Trajectories for the swarm; b) position time history.

of the objective function to minimize for ILQR that allows the attraction of individual mixtures to the desired intensity while repulsing away from each other.

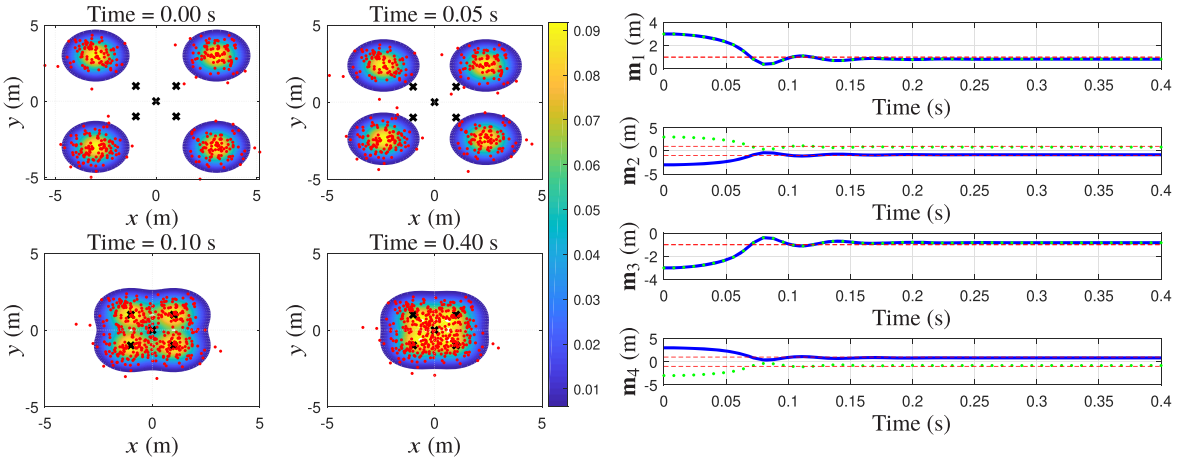
3. *Case 3: Five Desired Gaussian Mixtures*

Case 3 shows the effect of five desired Gaussian mixtures with the four swarm Gaussian mixtures using quasi-Newton MPC and ILQR. Figure 8b shows the time histories for all the mixtures using quasi-Newton MPC. The trajectory snapshots of the Gaussian mixtures are visually shown relative to the desired Gaussian mixtures in Fig. 8a. From Fig. 8b, the intensity converges in 0.19 s with a steady state error of approximately 0.17, which follows the theory as expected. Because the swarm Gaussian mixtures are far from each other, the effects of repulsion are minimal. Also, the mixtures are attracted to the four desired Gaussian mixtures that make up a square, but they are also attracted to the desired Gaussian mixture at the origin. This is due to the minimization of the objective function that has both an L_2^2 and a quadratic term where the individual mixtures will attract toward the desired intensity. Because there is an additional desired Gaussian mixture at the origin, all four swarm Gaussian mixtures are affected by the origin as they are moving toward the four square desired Gaussian mixtures. Thus, compared with case 1 with only four desired Gaussian mixtures, the swarm intensity, in this case, will have a steady-state error due to the attraction to the additional desired Gaussian mixture. ILQR is also used to show how five desired Gaussian mixtures affect the quadratization of the L_2^2 plus quadratic objective function. Figures 9a and 9b show the trajectory snapshots and time history, respectively. The swarm converges in 0.03 s and 0.12 of steady-state error. This steady-state error shows the attraction of the desired Gaussian mixture at the origin, which follows directly from results from the L_2^2 plus quadratic distance given by Fig. 3c.

B. Clohessy–Wiltshire Relative Motion

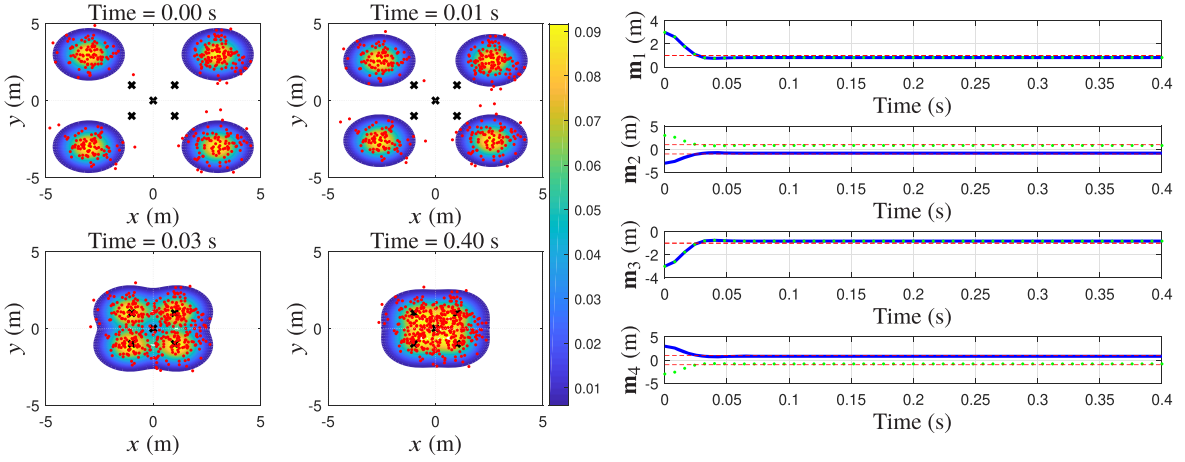
1. *Relative Motion with Perfect Information*

For the spacecraft relative motion, 77 Gaussian mixtures are initialized (birthed) from uniformly random initial conditions between −1 and 1 m from the chief satellite in a circular orbit. This is similar to the setup in [30] and follows Fig. 1 without the Swarm Estimation block because it is assumed that the state information received throughout the simulation is perfect. Additionally, no agent dies or spawns during the simulation. The goal is to control the spacecraft into a moving star-shaped pattern. Both the spacecraft and the rotating star pattern have an orbital frequency of $n = 0.00110678$ rad/s. Figure 10a shows the trajectory snapshots of the spacecraft (contours) and the desire Gaussian mixtures (black x's) using ILQR and the L_2^2 plus quadratic distance. The Gaussian mixtures, represented by each contour, can be safely assumed to contain a single agent. As time progresses, agents converge quickly into the formation and maintains the formation for the simulation time of 40 min. A few agents lag behind the desired targets due to the repelling effect from their proximity to other agents in the swarm. Figure 10b shows the acceleration for five agents to maintain the star



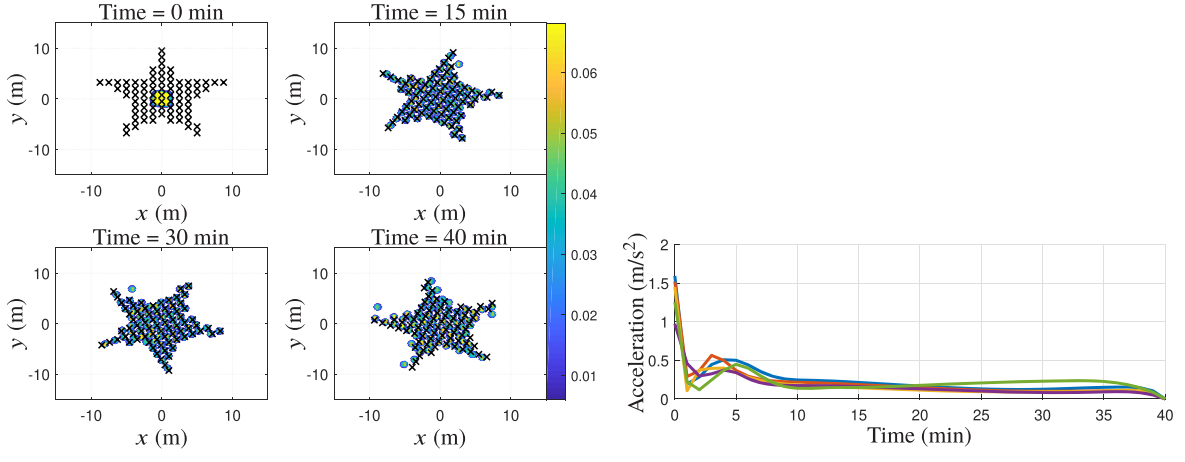
a) L_2^2 + quadratic distance trajectory b) L_2^2 + quadratic distance position time history

Fig. 8 Case 3: 4 Gaussian mixture swarm controlled to five desired Gaussian mixtures via quasi-Newton MPC. a) Trajectories for the swarm; b) position time history.



a) L_2^2 + quadratic distance trajectory b) L_2^2 + quadratic distance position time history

Fig. 9 Case 3: 4 Gaussian mixture swarm controlled to five desired Gaussian mixtures via ILQR. a) Trajectories for the swarm; b) position time history.



a) Clohessy-Wiltshire trajectory snapshots b) Control input

Fig. 10 Seventy-seven Gaussian mixture spacecraft swarm controlled to a rotating star target via ILQR with perfect information. a) Trajectories; b) acceleration for five agents.

formation. From these results, control using RFS can be expanded to physical spacecraft systems and can be used for moving targets.

2. *Relative Motion with Imperfect Information*

Next, the imperfect information (i.e., process, measurement, and clutter noise) is included in the simulation. To control with imperfect information, the GM-PHD filter is used in the Swarm Estimation block in Fig. 1 with the RFS control method. The GM-PHD filter determines the estimates of the intensities that are used for RFS control. The problem was altered to be more complicated by including differing birth and death times for the agents. With the addition of imperfect information and the added complication of changing number of agents, using the GM-PHD filter provides accurate estimates of the agents through time, which allows for RFS control in the loop. Figure 11a shows the cardinality or number of agents in the swarm through time. The solid line is the true number of agents, and the dotted line shows the estimate at each time step. At each time step, the agent estimates are fed through the RFS control using ILQR to obtain a control input for each agent. Then, the estimates are controlled and fed back to the GM-PHD filter at the next time step. Figure 11c shows the snapshots of the controlled agents (black circles) and targets (green stars) at each time step. Figure 11b shows the time history for the true agents (solid lines), estimated agents (black dots), and overall measurements (gray x's). From Fig. 11a, as the true agents die or birth initially, estimates of the occurrence are accurate. As the number of agents increases, the estimates become less accurate. This is because the GM-PHD filter only uses the first-order statistical moment to propagate the cardinality information of agents [31]. The cardinality distribution is unknown, and it is approximated as a Poisson distribution. For a Poisson distribution, the mean and covariance are equal. Therefore, if there are a larger number of agents in the field, the corresponding covariance of the cardinality distribution is also higher. Although the estimates are less accurate at high cardinality, the individual agents are controlled successfully into a star pattern in the presence of imperfect information. This is shown directly in Figs. 11c and 11b. As agents die or birth, the corresponding control dies or births with it, and due to the L_2^2 plus quadratic distance, agents are flexible to move into different parts of the formation.

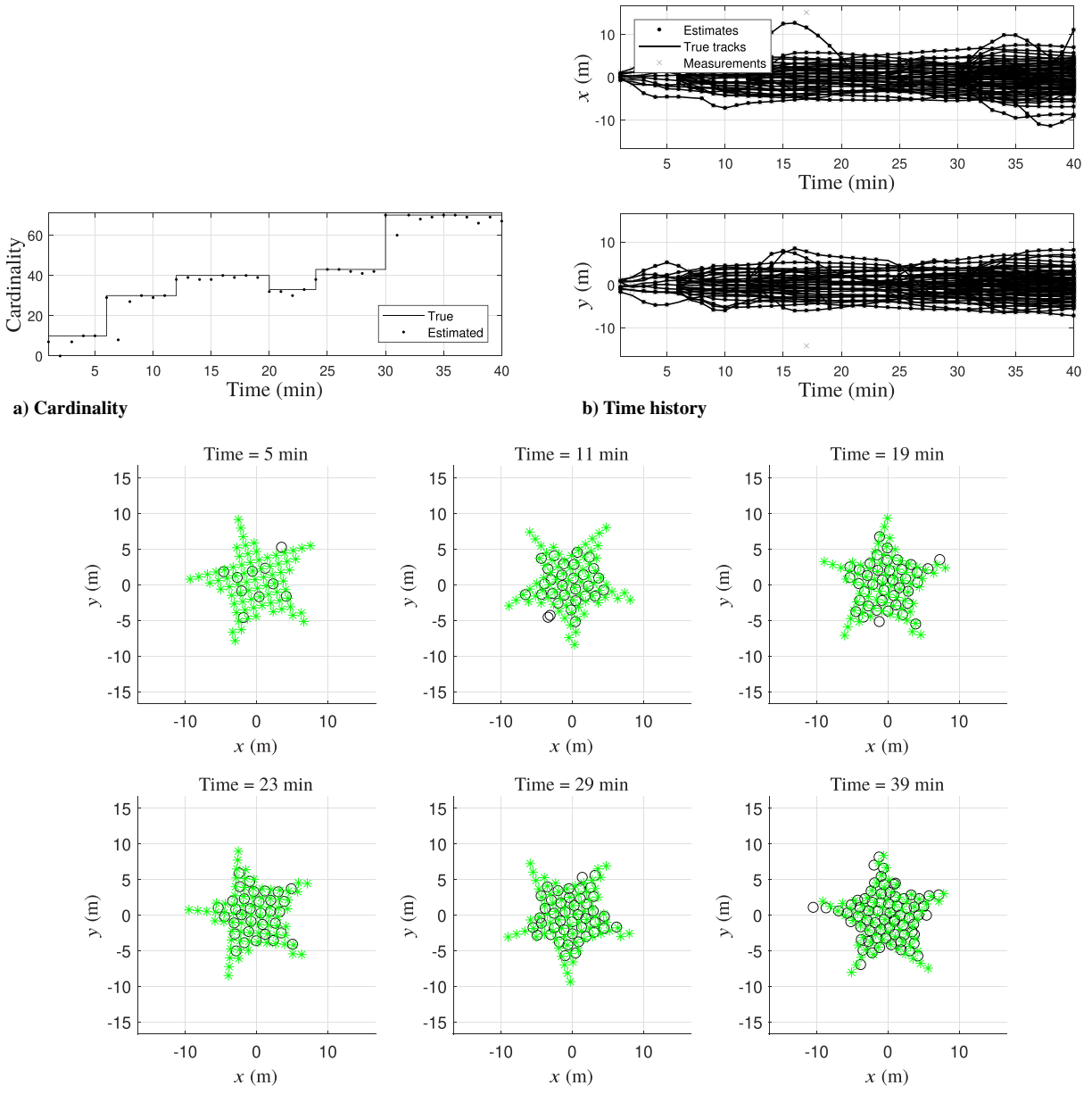


Fig. 11 c) The seventy-seven-agent spacecraft swarm controlled via ILQR to a rotating star target with imperfect information. a, b) Plot cardinality and the tracks, respectively.

VII. Limitations

RFS control for large collaborative swarms provides control solutions that are adaptive to varying swarm size (number of agents) and desired targets in the presence of process, measurement, and cardinality uncertainty, but several limitations currently exist. First, agents are assumed to be identical and unlabeled. Thus, this formulation does not provide control to a specific agent in the field. Secondly, control optimality of the solutions shown in this work is demonstrated through empirical results, but this work lacks theoretical proofs of robustness and optimally. Thirdly, RFS control was applied using a complete topology (centralized control); therefore, computational inefficiencies do exist. But the authors believe that the RFS-based method can be generalized and extended to decentralized applications. Lastly, no strict collision avoidance methods are applied for this problem. Although repelling behavior between agents exists in the objective function, no strict collision avoidance constraints are applied. Thus, this provides many areas for future work.

VIII. Conclusions

The objective of this paper is to formulate the multi-agent estimation and control background for swarming formations using the GM-PHD filter and either quasi-Newton MPC or ILQR from RFS theory. The RFS formulation is used to control the concentration of agents to match a desired distribution. By setting up the problem using information divergence to define the distance between the swarm RFS and the desired target configuration, an optimal control problem is found that tracks a linear system with a nonquadratic objection function through the use of quasi-Newton MPC and ILQR. The results show that the approach can be adaptive to varying swarm size and desired targets. Lastly, control using RFSs is also applied to the spacecraft relative motion problem by "closing-the-loop" with the GM-PHD filter. With the inclusion of process and measurement noise and uncertainty in the large number of agents in the field, a converging control solution is found obtained from estimates from the GM-PHD filter. These examples show the benefit of control using RFS by overcoming the curse of dimensionality.

Appendix A: Differential Dynamic Programming

Finite-horizon LQR control is first discussed to provide the necessary background for DDP discussed afterward.

A.1. LQR Finite-Horizon Optimal Control Problem

The linear quadratic regulator problem is defined by a discrete time-varying system given by

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\epsilon}_k \tag{A1}$$

where $\boldsymbol{\epsilon}_k$ is Brownian process noise. For the finite horizon N, the total cost is calculated from an initial state $\boldsymbol{x}_0$ and using the control sequence $U = [\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{N-1}]$ applied to the dynamics given by

$$J(\boldsymbol{x}_0, U) = \sum_{k=0}^{N-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k) + l_f(\boldsymbol{x}_N) \tag{A2}$$

where $l(\boldsymbol{u}_k, \boldsymbol{u}_k)$ is the running cost and $l_f(\boldsymbol{x}_N)$ is the terminal cost. The LQR costs are quadratic given by

$$l(\boldsymbol{x}_k, \boldsymbol{u}_k) = \frac{1}{2}\begin{bmatrix} 1 \\ \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}^T \begin{bmatrix} 0 & \boldsymbol{q}_k^T & \boldsymbol{r}_k^T \\ \boldsymbol{q}_k & Q_k & P_k \\ \boldsymbol{r}_k & P_k & R_k \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix},$$

$$l_f(\boldsymbol{x}_N) = \frac{1}{2}\boldsymbol{x}_N^T Q_N \boldsymbol{x}_N + \boldsymbol{x}_N^T \boldsymbol{q}_N \tag{A3}$$

where $\boldsymbol{q}_k$, $\boldsymbol{r}_k$, Q_k, R_k, and P_k are the running weights (coefficients), and Q_N and $\boldsymbol{q}_N$ are the terminal weights. The weight matrices Q_k and R_k are positive definite and the block matrix

$$\begin{bmatrix} Q_k & P_k \\ P_k & R_k \end{bmatrix}$$

is positive semidefinite [32]. The costs are substituted into Eq. (A2), and due to the symmetry in the weight matrices, the total cost is simplified to

$$J(\boldsymbol{x}_0, U) = \sum_{k=0}^{N-1} x_k^T \boldsymbol{q}_k + u_k^T \boldsymbol{r}_k + \frac{1}{2}x_k^T Q_k x_k + \frac{1}{2}u_k^T R_k u_k + u_k^T P_k x_k + \frac{1}{2}x_N^T Q_N x_N + x_N^T \boldsymbol{q}_N \tag{A4}$$

The optimal control solution is based on minimizing the cost function in terms of the control sequence, which is given by

$$U^*(\boldsymbol{x}_0) = \arg\min_U J(\boldsymbol{x}_0, U) \tag{A5}$$

To solve for the optimal control solution given by Eq. (A5), a value iteration method is used. Value iteration is a method that determines the optimal cost-to-go (value) starting at the final time step and moving backward in time minimizing the control sequence. Similar to Eq. (A2) and (A5), the cost-to-go and optimal cost-to-go are defined as

$$J(\boldsymbol{x}_k, U_k) = \sum_k^{N-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k) + l_f(\boldsymbol{x}_N) \tag{A6a}$$

$$V(\boldsymbol{x}_k) = \min_{U_k} J(\boldsymbol{x}_k, U_k) \tag{A6b}$$

where $U_k = [\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_{N-1}]$. Instead, the cost starts from time step k instead of $k = 0$. At a time step k, the optimal cost-to-go function is a quadratic function given by

$$V(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T S_k \boldsymbol{x}_k + \boldsymbol{x}_k^T \boldsymbol{s}_k + c_k \tag{A7}$$

where S_k, $\boldsymbol{s}_k$, and c_k are computed backward in time using the value iteration method. First, the final conditions $S_N = Q_N$, $\boldsymbol{s}_N = \boldsymbol{q}_N$, and $c_N = c$ are set. This reduces the minimization of the entire control sequence to just a minimization over a control input at a time step, which is the principle of optimality [33]. To find the optimal cost-to-go, the Riccati equations are used to propagate the final conditions backward in time given by

$$S_k = A_k^T S_{k+1} A_k + Q_k - (B_k^T S_{k+1} A_k + P_k^T)^T (B_k^T S_{k+1} B_k + R_k)^{-1} \times (B_k^T S_{k+1} A_k + P_k^T) \tag{A8a}$$

$$\boldsymbol{s}_k = \boldsymbol{q}_k + A_k^T \boldsymbol{s}_{k+1} + A_k^T S_{k+1} \boldsymbol{g}_k - (B_k^T S_{k+1} A_k + P_k^T)^T \times (B_k^T S_{k+1} B_k + R_k)^{-1} (B_k^T S_{k+1} \boldsymbol{g}_k + B_k^T \boldsymbol{s}_{k+1} + \boldsymbol{r}_k) \tag{A8b}$$

$$c_k = \boldsymbol{g}_k^T S_{k+1} \boldsymbol{g}_k + 2\boldsymbol{s}_{k+1}^T \boldsymbol{g}_k + c_{k+1} - (B_k^T S_{k+1} \boldsymbol{g}_k + B_k^T \boldsymbol{s}_{k+1} + \boldsymbol{r}_k)^T \times (B_k^T S_{k+1} B_k + R_k)^{-1} (B_k^T S_{k+1} \boldsymbol{g}_k + B_k^T \boldsymbol{s}_{k+1} + \boldsymbol{r}_k) \tag{A8c}$$

Using the Riccati solution, the optimal control policy is in the affine form

$$\boldsymbol{u}_k(\boldsymbol{x}_k) = K_k \boldsymbol{x}_k + \boldsymbol{l}_k \tag{A9}$$

where the controller K_k and controller offset is given by

$$K_k = -(R_k + B_k^T S_{k+1} B_k)^{-1} (B_k^T S_{k+1} A_k + P_k^T) \tag{A10a}$$

$$\boldsymbol{l}_k = -(R_k + B_k^T S_{k+1} B_k)^{-1} (B_k^T S_{k+1} \boldsymbol{g}_k + B_k^T \boldsymbol{s}_{k+1} + \boldsymbol{r}_k) \tag{A10b}$$

This optimal solution to the LQR problem works for linear dynamics and quadratic cost functions, but unfortunately, the objective function specified for the swarm problem is nonquadratic. Fortunately, DDP can be used for nonlinear dynamics and nonquadratic local cost functions.

A.2. Differential Dynamic Programming Problem

The DDP approach to solving nonlinear and nonquadratic equations uses a similar process as the LQR solution, but a second-order approximation of the dynamics and objective function are obtained for value iteration and the solution is iterated to increasingly get better approximations of the optimal trajectory of the system. Note that if linear dynamics are used, the ILQR formulation is obtained [28,34]. Because the results are produced by a linear system, both the DDP and ILQR terms can be used interchangeably. The following discussion on DDP follows closely to that of Tassa et al. [34,35]. The general nonlinear discrete-time dynamics is given by

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{A11}$$

where the state at the next time step, $\boldsymbol{x}_{k+1}$, is a function of the current state $\boldsymbol{x}_k$ and control input $\boldsymbol{u}_k$. The cost function is in the form of Eq. (A2), but the costs are nonquadratic. The solution to the optimal control problem is Eq. (A5). Similarly, the cost-to-go and the optimal cost-to-go function are defined by Eq. (A6a) and Eq. (A.6b), respectively. Given the terminal condition $V(\boldsymbol{x}_N) = l_f(\boldsymbol{x}_N)$, the optimal cost-to-go obtained from the principle of optimality is

$$V(\boldsymbol{x}_k) = \min_{\boldsymbol{u}_k}(l(\boldsymbol{x}_k, \boldsymbol{u}_k) + V(\boldsymbol{x}_{k+1})) \tag{A12}$$

which minimizes over the control at a time step and solved through time by a backward pass (value iteration).

a. Backward Pass

The first step in the backward pass (value iteration) is to determine a value function that is quadratic. The argument in Eq. (A12) is taken as a function of small perturbations around the state ($\delta\boldsymbol{x}_k$) and control input ($\delta\boldsymbol{u}_k$), and it is quadratized through a second-order Taylor series expansion given by

$$\begin{aligned} Q(\delta\boldsymbol{x}, \delta\boldsymbol{u}) &= l(\boldsymbol{x}_k + \delta\boldsymbol{x}_k, \boldsymbol{u}_k + \delta\boldsymbol{u})_k - l(\boldsymbol{x}, \boldsymbol{u}) + V(\boldsymbol{x}_{k+1} + \delta\boldsymbol{x}_{k+1}) \\ &\quad - V(\boldsymbol{x}_{k+1}), \\ &\approx \frac{1}{2}\begin{bmatrix} 1 \\ \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\boldsymbol{x}_k \\ \delta\boldsymbol{u}_k \end{bmatrix} \end{aligned} \tag{A13}$$

where Q_x, Q_u, Q_{xx}, Q_{xu}, and Q_{uu} are the running coefficients (weights) of the quadratized value function at a certain time step. Note, in the standard formulation, the time step k is dropped for these equations. Any primes denote the next time step. The equations for the running weights are given by

$$Q_x = l_x + f_x^T V_x' \tag{A14a}$$

$$Q_u = l_u + f_u^T V_x' \tag{A14b}$$

$$Q_{xx} = l_{xx} + f_x^T V_{xx}' f_x + V_x' f_{xx} \tag{A14c}$$

$$Q_{uu} = l_{uu} + f_u^T V_{xx}' f_u + V_x' f_{uu} \tag{A14d}$$

$$Q_{ux} = l_{ux} + f_u^T V_{xx}' f_x + V_x' f_{ux} \tag{A14e}$$

where l_x, l_u, l_{xx}, l_{uu}, and l_{ux} are the gradients and Hessians of the cost function; f_x, f_u, f_{xx}, f_{uu}, and f_{ux} are the gradients and Hessians of the nonlinear dynamics; and V_x' and V_{xx}' are the gradient and Hessian of the value function. For the ILQR formulation, the gradients and Hessians for an LTV system (which is the model used for RFS control in Sec. V) are trivial, but for the DDP formulation, the gradients and Hessians for the nonlinear dynamics must be computed. By using this quadratic approximation, the minimum in terms of $\delta\boldsymbol{u}$ is found using

$$\delta\boldsymbol{u} = \arg\min_{\delta\boldsymbol{u}} Q(\delta\boldsymbol{x}, \delta\boldsymbol{u}) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x}) \tag{A15}$$

which provides local feedback and feed-forward gains of

$$K = -Q_{uu}^{-1} Q_{ux} \tag{A16a}$$

$$\boldsymbol{k} = -Q_{uu}^{-1} Q_u \tag{A16b}$$

respectively. The locally optimal controller is substituted back into Eq. (A13) to get the optimal value given by

$$\Delta V = -\frac{1}{2}\boldsymbol{k}^T Q_{uu}\boldsymbol{k} \tag{A17a}$$

$$V_x = Q_x - K^T Q_{uu}\boldsymbol{k} \tag{A17b}$$

$$V_{xx} = Q_{xx} - K^T Q_{uu} K \tag{A17c}$$

so the value can be propagated backward in time to find new locally optimal solutions to the value function.

b. Forward Pass

By continually computing the quadratic approximations in Eq. (A14), local controller in Eq. (A16), and the new values in Eq. (A.17) backward in time from the terminal condition $V(\boldsymbol{x}_N) = l_f(\boldsymbol{x}_N)$, the updated trajectory can be found through a forward pass given by

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \tag{A18a}$$

$$\hat{\boldsymbol{u}}_k = \boldsymbol{u}_k + \boldsymbol{k}_k + K_k(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k) \tag{A18b}$$

$$\hat{\boldsymbol{x}}_{k+1} = f(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k) \tag{A18c}$$

where $\hat{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{u}}_k$ consists of the state and control input at a time step of the new trajectory ($\hat{\boldsymbol{X}}, \hat{\boldsymbol{U}}$). This composes one iteration of DDP. If the cost of the new trajectory, ($\hat{\boldsymbol{X}}, \hat{\boldsymbol{U}}$), is less than the cost of the old trajectory, ($\boldsymbol{X}, \boldsymbol{U}$), then $\boldsymbol{X} = \hat{X}$ and $\boldsymbol{U} = \hat{\boldsymbol{U}}$ are set, and the algorithm is ran again until a convergence threshold is met between the old and new costs.

c. Regularization via the Levenberg–Marquardt Heuristic

If the cost of the new trajectory is greater than the cost of the old trajectory, the iteration has not provided a better solution. To circumvent this issue, the Hessian is regularized. This is called the Levenberg–Marquardt heuristic. The control sequence that is calculated in DDP is computed like a Newton optimization that uses second-order information (curvature information) on top of the first-order information (gradient information) [36]. By including second-order curvature to the update, optimization can occur faster, but this relies on the fact that the Hessian is positive definite and an accurate quadratic model. If the control update is not improving (for a nonpositive definite Hessian and inaccurate quadratic model), the Levenberg–Marquardt heuristic uses less curvature information and more on the gradient information. This regularization is added to the Hessian of the control cost given by

$$\tilde{Q}_{uu} = Q_{uu} + \mu I_m \tag{A19}$$

where $\tilde{Q}_{uu}$ is the regularized control cost Hessian, μ is the Levenberg–Marquardt parameter, and I_m is the identity matrix that is the size of the control input vector [37]. This allows for the increase or decrease of curvature information in the optimization by adding a quadratic

cost around the current control input. Unfortunately, adding this regularization term can have different effects at different time steps using the same control perturbation based on a changing f_u in the linearized dynamics. By increasing $\mu \to \infty$, the $\boldsymbol{k}$ and K gains become very small due to the $\tilde{Q}_{uu}^{-1}$ term. Therefore, the regularization term is improved by penalizing the states instead of the control inputs, which are given by

$$\tilde{Q}_{uu} = l_{uu} + f_u^T(V'_{xx} + \mu I_n)f_u + V'_x f_{uu} \tag{A20a}$$

$$\tilde{Q}_{ux} = l_{ux} + f_u^T(V'_{xx} + \mu I_n)f_x + V'_x f_{ux} \tag{A20b}$$

$$K = -\tilde{Q}_{uu}^{-1}\tilde{Q}_{ux} \tag{A20c}$$

$$\boldsymbol{k} = -\tilde{Q}_{uu}^{-1}Q_u \tag{A20d}$$

where I_n is the identity matrix that is the size of the state vector. The μ parameter is placed on the state instead of the control input. For this method, the regularization term is directly incorporated with f_u, and the feedback gains $\boldsymbol{k}$ and K do not disappear as $\mu \to \infty$. Instead, the new $\boldsymbol{k}$ and K values bring the new trajectory closer to the old one. For the implementation of the μ term, three requirements should be followed. If reaching the minimum is accurate, the μ should become zero in order to obtain faster convergence due to the second-order optimization term. If a nonpositive definite $\tilde{Q}_{uu}$ is found, the backward pass should be restarted with a larger μ. The last requirement is that when a $\mu > 0$ is needed, the smallest μ should be used that allows the $\tilde{Q}_{uu}$ to be positive definite. Therefore, more of the second-order information can be used to provide faster convergence than gradient descent. The specific algorithm is found in [35].

Equation (A17) must also be modified based on regularization added in Eq. (A20a) [28]. Equation (A17) was originally derived using Eqs. (A13) and (A15), but using the new regularized terms in Eq. (A20a) creates error. Therefore, the modified values at a time step k are

$$\Delta V = \frac{1}{2}\boldsymbol{k}^T Q_{uu}\boldsymbol{k} + \boldsymbol{k}^T Q_u \tag{A21a}$$

$$V_x = Q_x + K^T Q_{uu}\boldsymbol{k} + K^T Q_u + Q_{ux}^T\boldsymbol{k} \tag{A21b}$$

$$V_{xx} = Q_{xx} + K^T Q_{uu}K + K^T Q_{ux} + Q_{ux}^T K \tag{A21c}$$

The regularization terms create a faster and more accurate solution to the backward pass of the DDP solution.

d. Forward Pass Line Search

Regularization of the forward pass can improve convergence and performance of the DDP algorithm. For linear time-varying systems, one iteration provides a minimal solution after one iteration. This is not the case for general nonlinear systems. Because nonlinear systems are approximated by a Taylor series expansion, there may be regions in the new DDP trajectory that are not valid about the nonlinear model. This may lead to divergence and have a larger cost function than the old trajectory. To fix this issue, a backtracking line-search parameter is introduced in the control update equation given by

$$\hat{\boldsymbol{u}}_k = \boldsymbol{u}_k + \alpha\boldsymbol{k}_k + K_k(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k) \tag{A22}$$

where α is set to $\alpha = 1$ at the start of the forward pass. Then the expected cost reduction is considered using

$$\Delta J(\alpha) = \alpha\sum_{k=0}^{N-1}\boldsymbol{k}(k)^T Q_u(k) + \frac{\alpha^2}{2}\sum_{k=0}^{N-1}\boldsymbol{k}^T(k)Q_{uu}(k)\boldsymbol{k}(k) \tag{A23}$$

A ratio z is determined using the actual and expected cost reduction given by

$$z = \frac{(J(\boldsymbol{x}_0, U) - J(\hat{\boldsymbol{x}}_0, \hat{U}))}{\Delta J(\alpha)} \tag{A24}$$

where $J(\boldsymbol{x}_0, U)$ and $J(\hat{\boldsymbol{x}}_0, \hat{U})$ are the old and new cost, respectively. The control update is accepted if the condition

$$0 < c_1 < z \tag{A25}$$

is met where c_1 is a parameter set by the user. The c_1 is usually set close to zero. If the condition is not met, the forward pass is restarted with a smaller α value, which means that the new trajectory strayed farther than the system's region of validity. By using the α line search parameter, convergence can be achieved for nonlinear systems by iteratively deceasing α to obtain a cost reduction.

e. DDP Summary

A DDP iteration can be summarized in four steps. First, an initial rollout of the nonlinear dynamics given by Eq. (A11) is integrated over time for a given control sequence U. If there is no good initialization of the control sequence, the control sequence can be set to $U = 0$. After the initial rollout, the derivatives of the cost function and nonlinear dynamics used in Eq. (A14) are found. The derivatives are used in the third step, which is to determine local control solutions using a backward pass. Using the terminal condition, $V(\boldsymbol{x}_N) = l_f(\boldsymbol{x}_N)$, local control solutions are found by iterating Eq. (A14), (A20), and (A21) backward at each time step. When a nonpositive definite $\tilde{Q}_{uu}$ is found, increase the regularization parameter μ and restart the backward pass. Once a local optimal policy is found, α is set to $\alpha = 1$, and Eqs. (A18c) and (A23) are propagated forward in time. If the integration diverged or cost reduction condition in Eq. (A25) was not met, the forward pass is restarted with a smaller α.

Appendix B: Receding Horizon Control using the RFS Formulation

An optimal solution, $\boldsymbol{u}$, can also be obtained in a real-time computational sense by minimizing the objective, Eq. (29), by reducing the finite horizon to a computational manageable prediction horizon using MPC or receding horizon control [24]. Conceptually, at a time k, the knowledge of the system model is used to derive a sequence $\boldsymbol{u}(k|k), \boldsymbol{u}(k+1|k), \boldsymbol{u}(k+2|k), \ldots, \boldsymbol{u}(k+T_p|k)$, where T_p is the finite prediction horizon from the current state $\boldsymbol{x}(k)$ [38]. With the input sequence, the state is moved forward in time by the control horizon, T_c; usually one time step. Then the same strategy is repeated for time $k+1$. The finite prediction horizon T_p can be chosen to be either small or large. As T_p increases, the degrees of freedom in the optimization increase, which can slow down the algorithm considerably, even though more of the future reference trajectory would be useful to bring the output closer to the reference. With a smaller T_p, the computation time will be faster, but the optimization may be more suboptimal. Thus, the swarm may not converge to the desired configuration.

For the RFS control formulation, a $\boldsymbol{u}$ that controls the swarm intensities through their statistics (mean and covariance) is found by minimizing the objective as given by Eqs. (30) and (31). This can be done by using MPC via the quasi-Newton method or DDP. DDP is able to determine an optimal solution for nonlinear equations of motion and a nonquadratic cost function through an iterative process of finding the solution involving second-order approximations of the dynamics and the objective function. The dynamic systems used in the results are linear; thus, DDP can be formed as its variant, ILQR. For the quasi-Newton method, the optimal control input $\boldsymbol{u}$ is found using MATLAB's fminunc solver [39]. Note that MPC via DDP and the quasi-Newton method are both closed-loop control methods in terms of the statistics (mean and covariance) of the system.

Acknowledgments

The authors wish to acknowledge support by the NASA under Contract Number NNX16CP45P issued through the NASA Small Business Technology Transfer (STTR) Program by the Jet Propulsion Laboratory (JPL) and led by Amir Rahmani at JPL. This work was also supported in part by Office of Naval Research (ONR) Code 321 and in part by National Science Foundation (NSF) Algorithms for Threat Detection (ATD) Grant 1738010. The authors wish to acknowledge useful conversations related to satellite technologies with Chuck Hisamoto, Vaughn Weirens, and Suneel Sheikh of ASTER Labs, Inc. The authors wish to acknowledge Andrew Akerson, a graduate student at the California Institute of Technology, who enabled useful notation for the mathematical theory used in the paper. Lastly, the authors wish to acknowledge Piyush M. Mehta, assistant professor at West Virginia University, for providing useful conversations in structuring and optimizing code. This paper is an extension of the paper published in the Proceedings of 2018 American Control Conference: Doerr B, Linares R, "Control of Large Swarms via Random Finite Set Theory," 2018 Annual American Control Conference (ACC), Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 2904–2909.

References

[1] Kube, C. R., and Zhang, H., "Collective Robotics: From Social Insects to Robots," *Adaptive Behavior*, Vol. 2, No. 2, 1993, pp. 189–218. https://doi.org/10.1177/105971239300200204

[2] Vassev, E., Hinchey, M., and Paquet, J., "Towards an ASSL Specification Model for NASA Swarm-Based Exploration Missions," *Proceedings of the 2008 ACM Symposium on Applied Computing—SAC 08*, ACM Press, New York, 2008, pp. 1652–1657. https://doi.org/10.1145/1363686.1364079

[3] Izzo, D., Pettazzi, L., and Ayre, M., "Mission Concept for Autonomous on Orbit Assembly of a Large Reflector in Space," *56th International Astronautical Congress*, Vol. 5, 2005, pp. D1–D4.

[4] Ryan, A., Zennaro, M., Howell, A., Sengupta, R., and Hedrick, J., "An Overview of Emerging Results in Cooperative UAV Control," *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601)*, Inst. of Electrical and Electronics Engineers, New York, 2004, pp. 602–607. https://doi.org/10.1109/cdc.2004.1428700

[5] Sommerville, I., *Software Engineering GE.*, Pearson Australia, 2016.

[6] Bakule, L., "Decentralized Control: An Overview," *Annual Reviews in Control*, Vol. 32, No. 1, 2008, pp. 87–98.

[7] Rubenstein, M., Cornejo, A., and Nagpal, R., "Programmable Self-Assembly in a Thousand-Robot Swarm," *Science*, Vol. 345, No. 6198, 2014, pp. 795–799. https://doi.org/10.1126/science.1254295

[8] Mondada, F., Gambardella, L., Floreano, D., Nolfi, S., Deneubourg, J., and Dorigo, M., "The Cooperation of Swarm-Bots—Physical Interactions in Collective Robotics," *IEEE Robotics & Automation Magazine*, Vol. 12, No. 2, 2005, pp. 21–28. https://doi.org/10.1109/mra.2005.1458313

[9] Mahler, R., "Multitarget Bayes Filtering via First-Order Multitarget Moments," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 39, No. 4, 2003, pp. 1152–1178. https://doi.org/10.1109/taes.2003.1261119

[10] Pace, M., Birattari, M., and Dorigo, M., "The Swarm/Potential Model: Modeling Robotics Swarms with Measure-Valued Recursions Associated to Random Finite Sets," *IEEE Transactions on Robotics* (submitted for publication).

[11] Doerr, B., and Linares, R., "Control of Large Swarms via Random Finite Set Theory," *2018 Annual American Control Conference (ACC)*, Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 2904–2909. https://doi.org/10.23919/acc.2018.8430968

[12] Vo, B.-N., and Ma, W.-K., "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, Vol. 54, No. 11, 2006, pp. 4091–4104. https://doi.org/10.1109/tsp.2006.881190

[13] Vo, B., Vo, B., and Cantoni, A., "The Cardinalized Probability Hypothesis Density Filter for Linear Gaussian Multi-Target Models," *2006 40th Annual Conference on Information Sciences and Systems*, Inst. of Electrical and Electronics Engineers, New York, 2006, pp. 681–686. https://doi.org/10.1109/ciss.2006.286554

[14] Vo, B.-T., and Vo, B.-N., "Labeled Random Finite Sets and Multi-Object Conjugate Priors," *IEEE Transactions on Signal Processing*, Vol. 61, No. 13, 2013, pp. 3460–3475. https://doi.org/10.1109/tsp.2013.2259822

[15] Bandyopadhyay, S., Chung, S.-J., and Hadaegh, F. Y., "Probabilistic Swarm Guidance Using Optimal Transport," *2014 IEEE Conference on Control Applications (CCA)*, Inst. of Electrical and Electronics Engineers, New York, 2014, pp. 498–505. https://doi.org/10.1109/cca.2014.6981395

[16] Foderaro, G., Ferrari, S., and Wettergren, T. A., "Distributed Optimal Control for Multi-Agent Trajectory Optimization," *Automatica*, Vol. 50, No. 1, 2014, pp. 149–154. https://doi.org/10.1016/j.automatica.2013.09.014

[17] Rudd, K., Foderaro, G., and Ferrari, S., "A Generalized Reduced Gradient Method for the Optimal Control of Multiscale Dynamical Systems," *52nd IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, New York, 2013. https://doi.org/10.1109/cdc.2013.6760478

[18] Foderaro, G., Zhu, P., Wei, H., Wettergren, T. A., and Ferrari, S., "Distributed Optimal Control of Sensor Networks for Dynamic Target Tracking," *IEEE Transactions on Control of Network Systems*, Vol. 5, No. 1, 2018, pp. 142–153. https://doi.org/10.1109/tcns.2016.2583070

[19] Ferrari, S., Foderaro, G., Zhu, P., and Wettergren, T. A., "Distributed Optimal Control of Multiscale Dynamical Systems: A Tutorial," *IEEE Control Systems*, Vol. 36, No. 2, 2016, pp. 102–116. https://doi.org/10.1109/mcs.2015.2512034

[20] Lunze, J., *Feedback Control of Large-Scale Systems*, Prentice Hall, Upper Saddle River, NJ, 1992.

[21] Reif, J. H., and Wang, H., "Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots," *Robotics and Autonomous Systems*, Vol. 27, No. 3, 1999, pp. 171–194. https://doi.org/10.1016/s0921-8890(99)00004-4

[22] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Decentralized Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, Nov. 2014, pp. 1725–1740. https://doi.org/10.2514/1.g000218

[23] Morgan, D., Chung, S.-J., and Hadaegh, F., "Spacecraft Swarm Guidance Using a Sequence of Decentralized Convex Optimizations," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2012-4583, 2012. https://doi.org/10.2514/6.2012-4583

[24] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Swarm Assignment and Trajectory Optimization Using Variable-Swarm, Distributed Auction Assignment and Model Predictive Control," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2015-0599, 2015. https://doi.org/10.2514/6.2015-0599

[25] Ma, W.-K., Vo, B.-N., Singh, S., and Baddeley, A., "Tracking an Unknown Time-Varying Number of Speakers Using TDOA Measurements: A Random Finite Set Approach," *IEEE Transactions on Signal Processing*, Vol. 54, No. 9, 2006, pp. 3291–3304. https://doi.org/10.1109/tsp.2006.877658

[26] Hoang, H. G., Vo, B.-N., Vo, B. T., and Mahler, R., "The Cauchy-Schwarz Divergence for Poisson Point Processes," *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, Inst. of Electrical and Electronics Engineers, New York, 2014, pp. 4475–4485. https://doi.org/10.1109/ssp.2014.6884620

[27] Banerjee, A., Merugu, S., Dhillon, I., and Ghosh, J., "Clustering with Bregman Divergences," *Proceedings of the 2004 SIAM International Conference on Data Mining*, Soc. for Industrial and Applied Mathematics, Philadelphia, PA, 2004, pp. 1705–1749. https://doi.org/10.1137/1.9781611972740.22

[28] Todorov, E., and Li, W., "A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems," *Proceedings of the 2005, American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, 2005, pp. 300–306. https://doi.org/10.1109/acc.2005.1469949

[29] Curtis, H. D., *Orbital Mechanics for Engineering Students*, Butterworth-Heinemann, Oxford, 2013.

[30] Eren, U., Demirer, N., and Açkmeşe, B., "Density-Based Feedback Control for Earth Orbiting Swarms via Velocity Fields," *IFAC-PapersOnLine*, Vol. 51, No. 12, 2018, pp. 44–49. https://doi.org/10.1016/j.ifacol.2018.07.086

[31] Mahler, R., "PHD Filters of Higher Order in Target Number," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 43, No. 99, 2007, pp. 1523–1543. https://doi.org/10.1109/taes.2007.4407475

[32] Inaba, M., and Corke, P. (eds.), *Robotics Research*, Springer, Berlin, 2016. https://doi.org/10.1007/978-3-319-28872-7

[33] Bellman, R., "The Theory of Dynamic Programming," *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 38, No. 8, 1952, p. 716.

[34] Tassa, Y., Mansard, N., and Todorov, E., "Control-Limited Differential Dynamic Programming," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2014, pp. 1168–1175. https://doi.org/10.1109/icra.2014.6907001

[35] Tassa, Y., Erez, T., and Todorov, E., "Synthesis and Stabilization of Complex Behaviors Through ONLINE Trajectory Optimization," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Inst. of Electrical and Electronics Engineers, New York, 2012, pp. 4906–4913. https://doi.org/10.1109/iros.2012.6386025

[36] Liao, L.-Z., and Shoemaker, C. A., "Advantages of Differential Dynamic Programming over Newton's Method for Discrete-Time Optimal Control Problems," Tech. Rept., Cornell Univ., New York, 1992.

[37] Liao, L.-Z., and Shoemaker, C., "Convergence in Unconstrained Discrete-Time Differential Dynamic Programming," *IEEE Transactions on Automatic Control*, Vol. 36, No. 6, 1991, pp. 692–706. https://doi.org/10.1109/9.86943

[38] Findeisen, R., and Allgöwer, F., "An Introduction to Nonlinear MODEL Predictive Control," *21st Benelux Meeting on Systems and Control*, Vol. 11, Technische Univ. Eindhoven Veldhoven, Eindhoven, The Netherlands, 2002, pp. 119–141.

[39] Fletcher, R., *Practical Methods of Optimization: Vol. 1 Unconstrained Optimization*, Wiley, New York, 1980.