Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Decentralized Safe Multi-agent Stochastic Optimal Control using Deep FBSDEs and ADMM

Marcus A. Pereira*†, Augustinos D. Saravanos*†, Oswin So‡ and Evangelos A. Theodorou*
*Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA
‡College of Computing, Georgia Institute of Technology, Atlanta, GA
{mpereira30, asaravanos, oswinso, evangelos.theodorou}@gatech.edu
†These authors contributed equally

***Abstract*—In this work, we propose a novel safe and scalable decentralized solution for multi-agent control in the presence of stochastic disturbances. Safety is mathematically encoded using stochastic control barrier functions and safe controls are computed by solving quadratic programs. Decentralization is achieved by augmenting to each agent's optimization variables, copy variables, for its neighbors. This allows us to decouple the centralized multi-agent optimization problem. However, to ensure safety, neighboring agents must agree on *what is safe for both of us*, creating a need for consensus. To enable safe consensus solutions, we incorporate an ADMM-based approach. Specifically, we propose a Merged Consensus ADMM-OSQP implicit neural network layer, that solves a mini-batch of both, local quadratic programs as well as the overall consensus problem, as a single optimization problem. This layer is embedded within a Deep Forward-Backward Stochastic Differential Equations (FBSDEs) network architecture at every time step, to facilitate end-to-end differentiable, safe and decentralized stochastic optimal control. The efficacy of the proposed approach is demonstrated on several challenging multi-robot tasks in simulation. By imposing collision avoidance constraints, the safe operation of all agents is ensured during the entire training process. We also demonstrate superior scalability in terms of computational and memory savings as compared to a centralized approach.**

I. Introduction

A vast variety of robotics applications such as coverage control [15], flocking of UAVs [32], multi-robot navigation [2], etc., falls into the class of multi-agent control problems. Such settings usually include a team of autonomous agents which are required to cooperate in order to accomplish a common goal. Effective frameworks for addressing these problems should be able to control the agents in an *optimal* manner, while ensuring their *safety under uncertainty*. In addition, it is of paramount importance that such methods are *scalable* to large-scale systems in terms of computational efficiency, memory usage and communication requirements.

Deep reinforcement learning has enjoyed significant fame over the past few years [30, 40, 41, 28], although being restricted to simulation settings where safety is not a primary concern. Therefore, very recently, the focus has shifted to the area of safe reinforcement learning [46, 12], where safety is required during the entire training process.

Control Barrier Functions (CBFs) [3, 4], have been popularly used in the robotics community to design controllers that guarantee invariance of user-defined safe sets. These are generally combined with off-the-shelf Quadratic Programming (QP) solvers to deliver real-time safe control solutions. However, most work has focused primarily on deterministic systems. Very recent works employing, so called, Stochastic CBFs (SCBFs) [14, 36, 38, 49] aim to bridge the gap, but lack scalability to large scale systems.

The confluence of Deep Learning and traditional optimization methods for *intra-layer optimization* has been of special interest lately. In particular, recent works embed QPs [1, 5, 7], root-finding methods [8] and even non-convex solvers [6, 20, 21] into the forward-pass of deep neural network layers and are popularly referred to as *implicit* neural network layers (other examples are [10, 27]). In most of these methods, the backward-pass is efficiently computed by invoking the implicit function theorem rather backpropagating through the unrolled graph of the forward-pass.

A general approach for continuous-time Stochastic Optimal Control (SOC) relying on Forward-Backward Stochastic Differential Equations (FBSDEs) was recently combined with deep learning [23] to solve high-dimensional Hamilton-Jacobi-Bellman PDEs (HJB-PDEs). Most noteworthy being deep FBSDEs [33, 34, 47, 11], a scalable framework for SOC problems that, at its core, leverages the function approximation capabilities of deep recurrent neural networks (specifically Long Short-Term Memory networks or LSTMs) to learn the gradient of the value-function, which can then be used to compute optimal control policies.

Distributed optimization-based frameworks have been gaining significant attention for tackling multi-agent control problems recently. A suitable method for deriving such algorithms is the Alternating Direction Method of Multipliers (ADMM) [9]. In particular, several ADMM-based methods such as [13, 22, 44], have been proposed, yielding elegant decentralized solutions for multi-agent control. Furthermore, recent works employing ADMM in a stochastic setting [37, 26], have shown to be capable of successfully encompassing both the safety under uncertainty and scalability desired attributes.

There has been very little work trying to put together all these ingredients into one framework. The Safe Deep FBSDEs framework [35] is one recent approach that partly addresses this need, the missing component being decentralization. More specifically, it uses an instantiation of implicit layers to solve embedded SCBF-based QPs combined with deep learning. Additionally, it utilizes a SCBF that tries to gaurantee safety

with a probability of 1. The method was tested on low-dimensional systems in simulation and the resulting policies appeared conservative. Inspired by Safe Deep FBSDEs with the aim to overcome its limitations in order to scale to large multi-agent SOC problems, we propose a decentralized approach to safety. To this end, the specific contributions of our work are,

1) A safe end-to-end differentiable framework that uses a novel SCBF formulation that is practically more meaningful and allows for safe, yet non-conservative policies,
2) A *fully decentralized* ADMM-based algorithm for large scale QPs embedded into an implicit neural network layer with drastic improvements in memory efficiency and training time as compared to a centralized approach,
3) A safe reinforcement framework, i.e., one that ensures safety not only at test time, but also while training the policy, and,
4) Extensive testing on several multi-robot challenging tasks in simulation, demonstrating the capabilities of the proposed framework.

II. Notation

Here, we introduce the notation followed throughout this paper. Non-bold symbols are used for scalars $a \in \mathbb{R}$, and bold lowercase and uppercase symbols for vectors $\mathbf{a} \in \mathbb{R}^n$ and matrices $\mathbf{A} \in \mathbb{R}^{n\times m}$, respectively. With $\mathbf{A}[i,j]$, we refer to the element of the i-th row and j-th column of $\mathbf{A}$. In addition, with $\mathbf{a} = [\mathbf{a}_1; \dots; \mathbf{a}_N]$, we denote the vertical concatenation of vectors $\mathbf{a}_1, \dots, \mathbf{a}_N$. The ℓ_2-norm of a vector $\mathbf{a} \in \mathbb{R}^n$ is defined as $\|\mathbf{a}\|_2 = \sqrt{\sum_{i=1}^n a_i^2}$ where $\mathbf{a} = [a_1; \dots; a_N]$. Moreover, the trace of a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is denoted with $\mathrm{tr}(\mathbf{A})$. We also define as $\mathrm{diag}(a_1, \dots, a_N) \in \mathbb{R}^{N\times N}$ the diagonal matrix with diagonal elements scalars $a_1, \dots, a_N$, and as $\mathrm{bdiag}(\mathbf{A}_1, \dots, \mathbf{A}_N)$ the block diagonal matrix constructed by the matrices $\mathbf{A}_1, \dots, \mathbf{A}_N$. Given a set $\mathcal{N}$, its cardinality is denoted by $|\mathcal{N}|$. With $[\![a, b]\!]$, we denote the integer interval $[a, b] \cap \mathbb{Z}$. Finally, given a convex set $\mathcal{C}$, $\Pi_{\mathcal{C}}(\mathbf{x})$ denotes the projection of a vector $\mathbf{x}$ onto the set and $\mathcal{I}_c(\mathbf{x})$ denotes the set indicator function such that $\mathcal{I}_c(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{C}$ and $\mathcal{I}_c(\mathbf{x}) = +\infty$ otherwise.

III. Problem Formulation

The framework developed in this paper can be applied to various problems, however, we focus on a multi-robot setting so as to establish a notion of distance as well as to provide context for constraints commonly encountered in this setting.

Consider a collection of N agents. The stochastic, nonlinear and control-affine dynamics for any agent $i \in [\![1, N]\!]$ are given by the following stochastic differential equation,

$$\mathrm{d}\mathbf{x}_i = \big(\mathbf{f}_i(\mathbf{x}_i) + \mathbf{G}_i(\mathbf{x}_i)\mathbf{u}_i\big)\mathrm{d}t + \mathbf{\Sigma}_i(\mathbf{x}_i)\mathrm{d}\mathbf{w}_i \tag{1}$$

where $\mathbf{x}_i = \mathbf{x}_i(t) \in \mathbb{R}^{n_i}$, $\mathbf{u}_i = \mathbf{u}_i(t) \in \mathbb{R}^{m_i}$, $\mathbf{w}_i = \mathbf{w}_i(t) \in \mathbb{R}^{p_i}$ are the state, control and standard Brownian-motion vectors, respectively, and $\mathbf{f}_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$, $\mathbf{G}_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i\times m_i}$ and $\mathbf{\Sigma}_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i\times p_i}$ denote the drift vector, actuation matrix and diffusion matrix, respectively. In order to state the centralized problem, we construct the global state, control and Brownian-motion vectors, $\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_N]$, $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \dots; \mathbf{u}_N]$, and $\mathbf{w} = [\mathbf{w}_1; \mathbf{w}_2; \dots; \mathbf{w}_N]$ by concatenation. Thus, the global stochastic dynamics are provided as follows,

$$\mathrm{d}\mathbf{x} = \big(\mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u}\big)\mathrm{d}t + \mathbf{\Sigma}(\mathbf{x})\mathrm{d}\mathbf{w} \tag{2}$$

with $\mathbf{f} = [\mathbf{f}_1; \mathbf{f}_2; \dots; \mathbf{f}_N]$, $\mathbf{G} = \mathrm{bdiag}(\mathbf{G}_1, \dots, \mathbf{G}_N)$ and $\mathbf{\Sigma} = \mathrm{bdiag}(\mathbf{\Sigma}_1, \dots, \mathbf{\Sigma}_N)$.

Stochastic Optimal Control (SOC) aims to minimize an expected cost subject to (2) given by,

$$\begin{aligned}\mathcal{J}(\mathbf{x},\, \mathbf{u},\, t_0) &= \sum_{i=1}^N J_i(\mathbf{x}_i,\, \mathbf{u}_i,\, t_0) \\ &= \sum_{i=1}^N \mathbb{E}\left[\phi_i\big(\mathbf{x}_i(T)\big) + \int_{t_0}^T \left(c_i\big(\mathbf{x}_i\big) + \frac{1}{2}\mathbf{u}_i^{\mathrm{T}}\mathbf{R}_i\mathbf{u}_i\right)\mathrm{d}t\right]\end{aligned} \tag{3}$$

where for any agent i, $\phi_i(\cdot)$ is the terminal state-cost function and the running cost comprises of a purely state-dependent term $c_i(\cdot)$, and a quadratic control-cost term with weighting coefficients given by the matrix $\mathbf{R}_i \in \mathbb{R}^{m_i\times m_i}$.

To solve the SOC problem using dynamic programming, we define the value function as $V(\mathbf{x},\, t) = \inf_{\mathbf{u}} \mathcal{J}(\mathbf{x},\, \mathbf{u},\, t)$, i.e., the optimal *cost-to-go* from state $\mathbf{x}$ at time step t. Next, using Ito's formula [39, Chapter 4], one can derive the Hamilton-Jacobi-Bellman Partial Differential Equation (HJB-PDE),

$$\begin{aligned}&\frac{\partial V}{\partial t} + \inf_{\mathbf{u}} \mathcal{H} = 0,\; V(\mathbf{x},\, T) = \sum_{i=1}^N \phi_i\big(\mathbf{x}_i(T)\big) \\ &\text{where, } \mathcal{H} = \frac{1}{2}\,\mathrm{tr}\left(\frac{\partial^2 V}{\partial \mathbf{x}^2}\mathbf{\Sigma}\mathbf{\Sigma}^{\mathrm{T}}\right) + \frac{\partial V}{\partial \mathbf{x}}^{\mathrm{T}}\Big(\mathbf{f} + \mathbf{G}\mathbf{u}\Big) \\ &\qquad\qquad + \sum_{i=1}^N \left(c_i + \frac{1}{2}\mathbf{u}_i^{\mathrm{T}}\mathbf{R}_i\mathbf{u}_i\right)\end{aligned} \tag{4}$$

which is a backward semilinear PDE and $\mathcal{H}$ is referred to as the Hamiltonian. Owing to the construction of the global state and control by concatenation, the Hamiltonian minimization can be split into a sum of decoupled minimizations of the Hamiltonians associated with each agent i (i.e., $\inf_{\mathbf{u}} \mathcal{H} = \sum_{i=1}^N \inf_{\mathbf{u}_i} \mathcal{H}_i$) in the absence of inter-agent safety constraints. However, when the latter are considered, this decoupling is no longer valid. In this work, we adopt a probabilistic approach to safety by imposing such constraints using Stochastic Control Barrier Functions (SCBFs). We assume that a safe set defined by $\mathcal{S} = \{\mathbf{x} : h(\mathbf{x}) \geq 0\}$ is known, where $h(\mathbf{x})$ is task-dependent and usually hand-designed and the goal is to stay within the safe set (with a high probability) for the entire time horizon. In the context of multi-robot systems, these SCBFs can encode obstacle and inter-agent collision avoidance constraints.

There are two types of SCBFs currently in literature – *almost-sure* (type-I) and those which allow for violations (type-II). The type-I SCBFs ensure that the system remains inside the safe set with a probability of 1 [14] for all time $t \in$

$[0,\, T]$. These were successfully utilized within a recent deep-learning based SOC framework [35], but simulations suggest that the framework leads to conservative policies preventing agents from getting close to each other, thus, reducing the flexibility of the trained policy. This restriction can prohibit application to systems with a large number of agents. On the other hand, type-II SCBFs allow tuning the probability of failure based on one's risk appetite and therefore have a higher practical appeal. We hypothesize that policies trained using Type-II SCBFs would result in less conservative and therefore more scalable policies. These have been employed in recent works [36, 49] and are based on derivation of Lyapunov functions for finite time stability of stochastic systems [25, Chapter 3]. Restating the result from [36, Proposition 1] here for convenience of the reader: *suppose there exists a twice differentiable function $B(\mathbf{x})$, that satisfies the following inequalities,*

$$B(\mathbf{x}) \geq 0 \quad \forall\, \mathbf{x} \in \mathbb{R}^{Nn_i} \tag{5}$$

$$B(\mathbf{x}) \geq 1 \quad \forall\, \mathbf{x} \in (\mathbb{R}^{Nn_i} \backslash \mathcal{S}) \tag{6}$$

$$\frac{\partial B}{\partial \mathbf{x}}^{\mathrm{T}} \Big(\mathbf{f} + \mathbf{G}\mathbf{u}\Big) + \frac{1}{2} tr\Big(\frac{\partial^2 B}{\partial \mathbf{x}^2} \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\Big) \leq -\alpha B(\mathbf{x}) + \beta, \tag{7}$$

where (7) is satisfied $\forall\, t \in [0,\, T]$ and $\forall\, \mathbf{x} \in \mathbb{R}^{Nn_i}$ for some $\alpha \geq 0$ and $\beta \geq 0$. Based on the chosen values of α and β, one can compute bounds on the probability of failure. We refer the reader to Section XVI of the supplementary material for additional details.

Inequalities (5) and (6) can be satisfied by choosing $B(\mathbf{x}) = e^{-\gamma h(\mathbf{x})}$ so that when the system exits the safe set $\mathcal{S}$, then $B(\mathbf{x}) > 1$ because $h(\mathbf{x}) < 0$. To satisfy (7), we impose it as a hard constraint on the Hamiltonian minimization in (4). Since the objective $\mathcal{H}(\mathbf{u})$ is quadratic and the constraint (7) is linear in $\mathbf{u}$, a safe optimal control can be obtained by solving the resulting Quadratic Program (QP) at every time step. Similar to the work in [35], we have the following HJB-PDE,

$$\frac{\partial V}{\partial t} + \inf_{\mathbf{u} \in \mathbf{u}_{\text{safe}}} \left\{ \sum_{i=1}^{N} \left(\frac{1}{2} \mathbf{u}_i^{\mathrm{T}} \mathbf{R}_i \mathbf{u}_i + \frac{\partial V}{\partial \mathbf{x}_i}^{\mathrm{T}} \mathbf{G}_i \mathbf{u}_i \right) \right\} + \sum_{i=1}^{N} \left(c_i + \frac{\partial V}{\partial \mathbf{x}_i}^{\mathrm{T}} \mathbf{f}_i + \frac{1}{2} \operatorname{tr}\left(\frac{\partial^2 V}{\partial \mathbf{x}_i^2} \boldsymbol{\Sigma}_i \boldsymbol{\Sigma}_i^{\mathrm{T}} \right) \right) = 0 \tag{8}$$

where, $\mathbf{u}_{\text{safe}} = \Bigg\{ \mathbf{u} \Bigg| \frac{\partial B_k}{\partial \bar{\mathbf{x}}_k}^{\mathrm{T}} \left(\bar{\mathbf{f}}_k + \bar{\mathbf{G}}_k \bar{\mathbf{u}}_k \right) + \frac{1}{2} \operatorname{tr}\left(\frac{\partial^2 B_k}{\partial \bar{\mathbf{x}}_k^2} \bar{\boldsymbol{\Sigma}}_k \bar{\boldsymbol{\Sigma}}_k^{\mathrm{T}} \right) \leq -\alpha B_k(\bar{\mathbf{x}}_k) + \beta,\, \forall\, k \in [\![1, N_{\text{ineq}}]\!] \Bigg\}$

where each B_k is either a pairwise safety constraint between two agents or between an agent and an obstacle. The vectors $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{u}}_k$ are obtained by concatenating states and controls of the two agents corresponding to B_k for inter-agent constraints or just the state and the control vectors of the ego agent for agent-obstacle constraints. Finally, $\bar{\mathbf{f}}_k$, $\bar{\mathbf{G}}_k$ and $\bar{\boldsymbol{\Sigma}}_k$ are constructed similar to $\bar{\mathbf{x}}_k$ and $N_{\text{ineq}} = \binom{N}{2} + NN_o$, is the total number of inequality constraints. The $\binom{N}{2}$ constraints account for all possible agent pairs for collision avoidance similar to work in [35, Section 4.3.2] and N_o is the number of obstacles. However, this clearly would not scale for large values of N. Further, due to the inter-agent constraints, the minimization of individual $\mathcal{H}_i$ can no longer be decoupled. Thus, the safe optimal control $\mathbf{u}^*$, has to be solved as one large QP, i.e, in a *centralized* manner.

IV. Centralized Solution Using Deep FBSDEs

Before we propose our decentralized solution to address scalability, we summarize the deep learning based solution to safe SOC problems adapted from recent work [35], which if applied directly to solve Equation (8) would lead to a centralized approach.

The unique solution of (8) is linked to that of a system of FBSDEs via the Nonlinear Feynman-Kac lemma (we refer the reader to Section XIV of the supplementary material for the derivation). Assuming that a solution $\mathbf{u}^*$ exists, the FBSDE system that solves (8) is given by,

$$\textbf{(FSDE)} \quad \mathrm{d}\mathbf{x} = (\mathbf{f} + \mathbf{G}\mathbf{u}^*)\mathrm{d}t + \boldsymbol{\Sigma}\mathrm{d}\mathbf{w},\, \mathbf{x}(0) = \mathbf{x}_0 \tag{9}$$

$$\textbf{(BSDE)} \begin{cases} \mathrm{d}V = -\left[\sum_{i=1}^{N} \left(c_i + \frac{1}{2} \mathbf{u}_i^{*\mathrm{T}} \mathbf{R}_i \mathbf{u}_i^* \right) \right] \mathrm{d}t + \frac{\partial V}{\partial \mathbf{x}}^{\mathrm{T}} \boldsymbol{\Sigma} \mathrm{d}\mathbf{w} \\ V\big(\mathbf{x}(T)\big) = \sum_{i=1}^{N} \phi_i\big(\mathbf{x}_i(T)\big) \end{cases} \tag{10}$$

where, $\mathbf{u}^* = \underset{\mathbf{u} \in \mathbf{u}_{\text{safe}}}{\arg\min}\, \mathcal{H}$.

Deep FBSDEs use deep neural networks to learn $\frac{\partial V}{\partial \mathbf{x}}(\mathbf{x},\, t;\, \theta)$, which can be used to compute optimal control policies $\mathbf{u}^*(\mathbf{x})$. Traditional methods [16, 19, 18, 17] to solve FBSDEs relied on back-propagating and approximating the conditional expectation of the value function, $\mathbb{E}\big[V(\mathbf{x},\, t)\big]$, using least squares. This approach is prone to numerical ill-conditioning issues depending on which area of the state-space the system visits, requires hand-picking basis functions and suffers from compounding least squares errors over time as $\mathbb{E}\big[V(\mathbf{x},\, t)\big]$ is back-propagated from $t = T$ to $t = t_0$. Deep FBSDEs circumvent the need to back-propagate $\mathbb{E}\big[V(\mathbf{x},\, t)\big]$ by instead parameterizing $\hat{V}\big(\mathbf{x}(0),\, 0\big)$ with trainable weights. Using this approximation of the initial condition, $V(\mathbf{x},\, t)$ can then be forward propagated similar to a forward SDE using (10). At the end of the time horizon, the predicted terminal value $\hat{V}\big(\mathbf{x}(T),\, T\big)$ that relies on the predictions of $\frac{\partial V}{\partial \mathbf{x}}(\mathbf{x},\, t;\, \theta)$ provided by a deep LSTM network, is compared to the true terminal value $V\big(\mathbf{x}(T),\, T\big)$ to construct a loss function. $V\big(\mathbf{x}(T),\, T\big)$ is evaluated using the given $\phi_i\big(\mathbf{x}_i(T)\big)$ and terminal states $\mathbf{x}(T)$ obtained by forward propagation of (9). This is then used to train the deep LSTM network using optimizers such as Adam [24]. Thus, deep FBSDEs is a self-supervised learning framework. Over iterations, as the loss is minimized, the network improves its predictions of $\hat{V}\big(\mathbf{x}(0),\, 0\big)$ and $\frac{\partial V}{\partial \mathbf{x}}(\mathbf{x},\, t;\, \theta)$.

For unconstrained problems, $\mathbf{u}_i^* = -\mathbf{R}_i^{-1} \mathbf{G}_i^{\mathrm{T}} \frac{\partial V}{\partial \mathbf{x}_i}$. However, when combined with SCBFs, $\mathbf{u}_i^*$ can only be computed

numerically. Similar to work in [35], safe optimal controls $\mathbf{u}_i^*$ can be computed in an end-to-end differentiable manner using an OptNet-like [5] implicit layer to solve the constrained QP (8) and to ensure efficient backpropagation for DNN training using the implicit function theorem.

V. Decentralized Approach

In this section, we propose a decentralized approach for addressing the Hamiltonian minimization in (8). To achieve this, we first reformulate the centralized problem in a distributed form. Subsequently, we propose a decentralized ADMM-based method combining elements from Consensus ADMM [9] and OSQP [42] for solving large-scale QPs, which is then employed for solving our problem.

A. Decentralized Problem

The key restriction in problem (8) which prevents us from directly solving it in a distributed manner is the coupling induced by the inter-agent constraints. To overcome this issue, let us introduce the *neighborhood* sets $\mathcal{N}_i,\ i \in [\![1, N]\!]$, which contain the indices of the neighboring agents of each agent i. For instance, in a 5-agent scenario where agents $2, 4, 5$ are neighbors of agent 1, its neighborhood set would be $\mathcal{N}_1 = \{2, 4, 5\}$. We also define the sets $\mathcal{P}_i = \{j : i \in \mathcal{N}_j\},\ i \in [\![1, N]\!]$, where each set $\mathcal{P}_i$ contains all the agents that have agent i as a neighbor.

Assumption 1. *Each agent $i \in [\![1, N]\!]$ is able to communicate with all agents $j \in \mathcal{N}_i \cup \mathcal{P}_i$, and vice versa.*

Next, we consider for each agent i, the copy control and state variables $\{\mathbf{u}_j^{(i)}\}_{j\in\mathcal{N}_i}$ and $\{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}$, respectively. Essentially, a copy variable $\mathbf{u}_j^{(i)}$ can be interpreted as *agent i deciding what is safe for its neighbor j from its own perspective*. Let us also define the augmented local variables containing the states and controls of all agents within agent i's neighborhood:

$$\tilde{\mathbf{u}}_i = \left[\mathbf{u}_i; \{\mathbf{u}_j^{(i)}\}_{j\in\mathcal{N}_i}\right],\ \tilde{\mathbf{x}}_i = \left[\mathbf{x}_i; \{\mathbf{x}_j^{(i)}\}_{j\in\mathcal{N}_i}\right],\ i \in [\![1, N]\!].$$

Nevertheless, the inclusion of the copy variables creates a requirement for enforcing a consensus between variables that correspond to the same agents. For this reason, we also introduce the global control variable $\mathbf{g} = [\mathbf{g}_1; \ldots; \mathbf{g}_N]$ and impose the following constraints between local and global variable components,

$$\mathbf{u}_j^{(i)} = \mathbf{g}_j,\ \forall j \in \mathcal{N}_i \cup \{i\},\ \forall i \in [\![1, N]\!]. \tag{11}$$

We can now formulate a *decentralized* form of the Hamiltonian minimization problem as,

$$\min \sum_{i=1}^{N} \mathcal{H}_i(\tilde{\mathbf{u}}_i) \tag{12a}$$

$$\text{s.t.}\quad \frac{\partial B^{i,k}}{\partial \tilde{\mathbf{x}}_{i,k}}\left(\tilde{\mathbf{f}}_{i,k} + \tilde{\mathbf{G}}_{i,k}\tilde{\mathbf{u}}_{i,k}\right) + \frac{1}{2}\operatorname{tr}\left(\frac{\partial^2 B^{i,k}}{\partial \tilde{\mathbf{x}}_{i,k}^2}\tilde{\mathbf{\Sigma}}_{i,k}\tilde{\mathbf{\Sigma}}_{i,k}^{\mathrm{T}}\right)$$
$$\leq -\alpha B^{i,k} + \beta,\ \forall k \in [\![1, N_{\text{ineq},i}]\!],\ \forall i \in [\![1, N]\!] \tag{12b}$$
$$\tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!] \tag{12c}$$

where the vectors $\tilde{\mathbf{x}}_{i,k}$, $\tilde{\mathbf{u}}_{i,k}$ are defined as $\tilde{\mathbf{x}}_{i,k} = [\mathbf{x}_i; \mathbf{x}_j^{(i)}]$, $\tilde{\mathbf{u}}_{i,k} = [\mathbf{u}_i; \mathbf{u}_j^{(i)}]$, if (12b) is an inter-agent constraint involving a specific neighboring agent $j \in \mathcal{N}_i$ and as $\tilde{\mathbf{x}}_{i,k} = \mathbf{x}_i$, $\tilde{\mathbf{u}}_{i,k} = \mathbf{u}_i$, if (12b) is an obstacle avoidance constraint. The functions $\tilde{\mathbf{f}}_{i,k}$, $\tilde{\mathbf{G}}_{i,k}$, $\tilde{\mathbf{\Sigma}}_{i,k}$ are defined accordingly in each case. Finally, $\tilde{\mathbf{g}}_i$ is defined as $\tilde{\mathbf{g}}_i = \left[\mathbf{g}_i; \{\mathbf{g}_j\}_{j\in\mathcal{N}_i}\right]$ and $N_{\text{ineq},i} = |\mathcal{N}_i| + N_o$.

Subsequently, by denoting the linear inequality constraints (12b) as $\mathbf{A}_i\tilde{\mathbf{u}}_i \leq \mathbf{d}_i$, we can rewrite problem (12) in a more compact form as,

$$\min \sum_{i=1}^{N} \mathcal{H}_i(\tilde{\mathbf{u}}_i) + \mathcal{I}_{\mathbf{A}_i\tilde{\mathbf{u}}_i \leq \mathbf{d}_i}(\mathbf{A}_i\tilde{\mathbf{u}}_i)$$
$$\text{s.t.}\quad \tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!]. \tag{13}$$

Problem (13) is now in a form where CADMM could be directly applied to solve it. This would yield a bilevel distributed optimization algorithm where at every ADMM iteration, each agent would first locally solve a QP, and then the local solutions would be used to perform the global and dual updates [9, Chapter 7]. These local QPs could be solved by well-known solvers such as OSQP [43], interior-point methods [31, 29, 5], etc.

B. Merged CADMM-OSQP Method

In this work, we exploit the fact that the inner QP program in CADMM could itself be solved using the ADMM-based solver OSQP. Therefore, we propose flattening the bilevel optimization framework with a novel Merged CADMM-OSQP method for solving QPs in a decentralized manner.

First, let us define $\tilde{\mathbf{R}}_i = \text{blkdiag}(\mathbf{R}_i, \{\mathbf{0}\}_{k\in[\![1,|\mathcal{N}_i|]\!]})$ and $\tilde{\mathbf{q}}_i = [\mathbf{q}_i; \{\mathbf{0}\}_{k\in[\![1,|\mathcal{N}_i|]\!]}]$ where $\mathbf{q}_i = \frac{\partial V}{\partial \mathbf{x}_i}^{\mathrm{T}}\mathbf{G}_i$. We can then reformulate (13) as,

$$\min \sum_{i=1}^{N} \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i + \mathcal{I}_{\tilde{\mathbf{z}}_i \leq \mathbf{d}_i}(\tilde{\mathbf{z}}_i)$$
$$\text{s.t.}\quad \mathbf{A}_i\tilde{\mathbf{u}}_i = \tilde{\mathbf{z}}_i,\ \tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!] \tag{14}$$

where each $\mathcal{H}_i(\tilde{\mathbf{u}}_i)$ is given by $\mathcal{H}_i(\tilde{\mathbf{u}}_i) = \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i$. Next, we introduce the auxiliary variables $\hat{\mathbf{z}}_i,\ i \in [\![1, N]\!]$ in a similar manner as in the OSQP derivation [43, Section 3] and transform (14) to,

$$\min \sum_{i=1}^{N} \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i + \mathcal{I}_{\mathbf{A}_i\tilde{\mathbf{u}}_i=\tilde{\mathbf{z}}_i}(\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i) + \mathcal{I}_{\tilde{\mathbf{z}}_i \leq \mathbf{d}_i}(\hat{\mathbf{z}}_i)$$
$$\text{s.t.}\quad \tilde{\mathbf{z}}_i = \hat{\mathbf{z}}_i,\ \tilde{\mathbf{u}}_i = \tilde{\mathbf{g}}_i,\ \forall i \in [\![1, N]\!]. \tag{15}$$

The Augmented Lagrangian (AL) of (15) yields,

$$\mathcal{L} = \sum_{i=1}^{N} \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i + \mathcal{I}_{\mathbf{A}_i\tilde{\mathbf{u}}_i=\tilde{\mathbf{z}}_i}(\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i) + \mathcal{I}_{\tilde{\mathbf{z}}_i \leq \mathbf{d}_i}(\hat{\mathbf{z}}_i)$$
$$+ \frac{\rho_1}{2}\left\|\tilde{\mathbf{z}}_i - \hat{\mathbf{z}}_i + \frac{\mathbf{y}_i}{\rho_1}\right\|_2^2 + \frac{\rho_2}{2}\left\|\tilde{\mathbf{u}}_i - \tilde{\mathbf{g}}_i + \frac{\boldsymbol{\zeta}_i}{\rho_2}\right\|_2^2 \tag{16}$$

where $\mathbf{y}_i$, $\boldsymbol{\zeta}_i$ are the dual variables for the corresponding equality constraints and $\rho_1, \rho_2 > 0$ are penalty parameters.

Therefore, it is possible to use ADMM in a manner such that the consensus and OSQP updates take place within the

same cycle of updates. In the following, the superscript $l \in [\![1, l_{\max}]\!]$ denotes the current ADMM iteration, with $l_{\max}$ being the maximum allowed iterations. The first block of updates on $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{z}}_i$ will be,

$$\begin{aligned}\{\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i\}^{l+1} = \underset{\tilde{\mathbf{u}}_i, \tilde{\mathbf{z}}_i}{\arg\min}\ & \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i \\ & + \frac{\rho_1}{2}\left\|\tilde{\mathbf{z}}_i - \hat{\mathbf{z}}_i^l + \frac{\mathbf{y}_i^l}{\rho_1}\right\|_2^2 + \frac{\rho_2}{2}\left\|\tilde{\mathbf{u}}_i - \tilde{\mathbf{g}}_i^l + \frac{\boldsymbol{\zeta}_i^l}{\rho_2}\right\|_2^2 \\ & \text{s.t. } \mathbf{A}_i\tilde{\mathbf{u}}_i = \tilde{\mathbf{z}}_i. \end{aligned} \tag{17}$$

The second block where $\hat{\mathbf{z}}_i$ and $\mathbf{g}$ are updated will consist of,

$$\hat{\mathbf{z}}_i^{l+1} = \Pi_{(-\infty, \mathbf{d}_i]}(\tilde{\mathbf{z}}_i^{l+1} + \frac{1}{\rho_1}\mathbf{y}_i^l) \tag{18a}$$

$$\mathbf{g}_i^{l+1} = \frac{1}{|\mathcal{P}_i| + 1}\sum_{j \in \mathcal{P}_i \cup \{i\}}\left(\mathbf{u}_i^{(j),l+1} + \frac{1}{\rho_2}\boldsymbol{\zeta}_i^{(j),l}\right) \tag{18b}$$

where $\boldsymbol{\zeta}_i^{(j)}$ is the part of the dual variable $\boldsymbol{\zeta}_i$ corresponding to the constraint $\mathbf{u}_j^{(i)} = \mathbf{g}_j$. Finally, the dual variables updates will be,

$$\mathbf{y}_i^{l+1} = \mathbf{y}_i^l + \rho_1(\tilde{\mathbf{z}}_i^{l+1} - \hat{\mathbf{z}}_i^{l+1}) \tag{19a}$$

$$\boldsymbol{\zeta}_i^{l+1} = \boldsymbol{\zeta}_i^l + \rho_2(\tilde{\mathbf{u}}_i^{l+1} - \tilde{\mathbf{g}}_i^{l+1}). \tag{19b}$$

Since updating each variable with subscript i only requires variables that have the same subscript, the updates (17), (18), (19) can be performed in parallel by each agent. Therefore, the algorithm can be executed in a fully decentralized manner. During every ADMM iteration, two communication steps are required. The first one takes place before the computation of (18b), where every agent $j \in \mathcal{P}_i$ must send its variables $\tilde{\mathbf{u}}_j^{l+1}$ and $\boldsymbol{\zeta}_j^l$ to each agent i. The second one is performed before (19b), where every agent $j \in \mathcal{N}_i$ must send $\mathbf{g}_j^{l+1}$ to agent i, so that the latter can construct the vector $\tilde{\mathbf{g}}_i^{l+1}$. Finally, one extra communication step is required before the first ADMM iteration between agent i and its neighbors $j \in \mathcal{N}_i$, to construct $\mathbf{A}_i$ and $\mathbf{d}_i$ for the inequality constraints which remain the same for all subsequent ADMM iterations.

It should be highlighted that the $N_{\text{ineq},i} = r + N_o$ constraints involved in each of the local subproblems, will be drastically fewer than the $N_{\text{ineq}} = \binom{N}{2} + NN_o$ constraints of the centralized problem for $r \ll N$.

VI. Implementation Details

We make the following assumption for our implementation,

Assumption 2. *All neighborhood sets within a single time step t, $\mathcal{N}_i(t)$, are of equal size r.*

This assumption ensures that all $N_{\text{ineq},i}$ are equal so that each local subproblem can be solved in a batched fashion, thereby allowing for efficient training on GPUs. Note that this is not a very restrictive assumption as we still allow the individual $\mathcal{N}_i(t)$ to change across time.

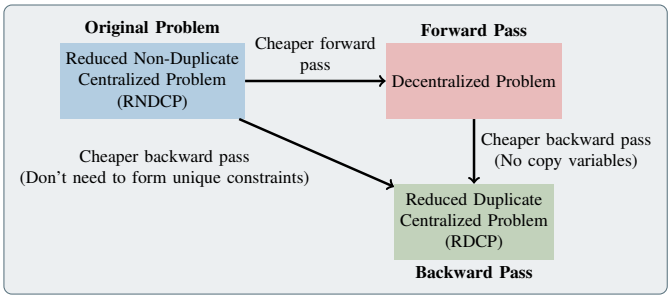


Fig. 1: Relationship between RNDCP, RDCP (22) and the Decentralized Problem (Section V).

A. Time Discretization

In order to use deep learning, we consider a Euler-Maruyama time discretization of (9) and (10) so that back-propagation through time can be performed on a finite number of time steps to train the deep network. This is similar to past deep FBSDE works [33, 34, 35] wherein using a finite time interval of Δt, the time horizon is divided into $\frac{T}{\Delta t}$ equal intervals of length Δt. The time-discretized equations are,

$$\mathbf{x}[\tau + 1] = \mathbf{x}[\tau] + (\mathbf{f} + \mathbf{G}\mathbf{u}^*)\Delta t + \boldsymbol{\Sigma}\sqrt{\Delta t}\,\boldsymbol{\epsilon} \tag{20}$$

$$\begin{aligned} V[\tau + 1] = V[\tau] - & \left[\sum_{i=1}^{N}\left(c_i + \frac{1}{2}\mathbf{u}_i^{*\mathrm{T}}\mathbf{R}_i\mathbf{u}_i^*\right)\right]\Delta t \\ & + \frac{\partial V}{\partial \mathbf{x}}^{\mathrm{T}}\boldsymbol{\Sigma}\sqrt{\Delta t}\,\boldsymbol{\epsilon} \end{aligned} \tag{21}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\tau \in [\![0, \frac{T}{\Delta t} - 1]\!]$.

B. Forward Pass

Similar to past work [33, 35], the forward pass involves propagating the discretized FBSDE and BSDE forward in time using (20) and (21). The primary distinction between [35] and our approach is a new implicit safe layer based on CADMM. This difference is clear by comparing the unrolled compute graph shown in Figure 2 with that of [35, Figure 1]. The additional difference from past works is the inclusion of extra fully-connected layers FC_c, FC_h and FC_V to allow for training from random initial conditions. These extra networks serve to initialize the initial cell-state and initial hidden-state of the LSTM layers and the initial value-function respectively.

C. Backward Pass

Our proposed decentralized Merged CADMM-OSQP solver is an instantiation of an implicit neural network layer. Hence, we utilize the implicit function theorem to compute the necessary gradients for the backward pass. Now, one approach could be to formulate the KKT matrix resulting from the KKT conditions that the optimal solution, $(\tilde{\mathbf{u}}_i^*, \mathbf{y}_i^*)\ \forall i \in [\![1, N]\!]$, must satisfy. This is precisely the solution of the decentralized problem in the forward pass. However, the resulting size of the KKT matrix could be very large depending on N and r because of the presence of control copy variables for each agent's neighbors used in the decentralized problem. Based on the insight that each $\tilde{\mathbf{u}}_i^*$ also satisfies the consensus constraint,

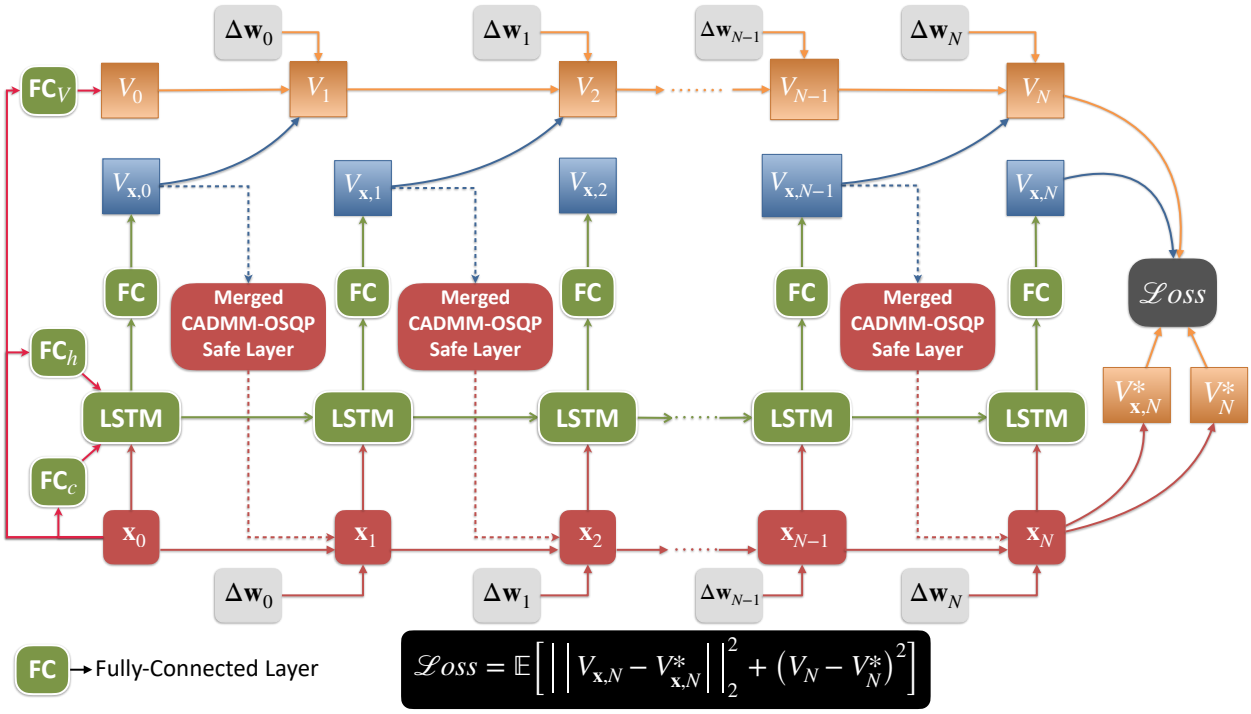


Fig. 2: Unrolled Deep FBSDE compute graph using the proposed Merged CADMM-OSQP implicit safe layer.

$\tilde{\mathbf{u}}_i^* = \tilde{\mathbf{g}}_i$, we can eliminate the copy variables and consider only the control variables of the ego agents of each neighborhood. However, if two agents i, j are mutual neighbors, i.e., $j \in \mathcal{N}_i$ and $i \in \mathcal{N}_j$, then the constraint involving i and j will appear twice in the constraint matrix (i.e., as duplicate row entries). We therefore refer to this problem as the *Reduced Duplicate Centralized Problem* (RDCP) where *reduced* indicates the reduction in the number of constraints from N_{ineq} to $N\bar{N}_{\text{ineq},i}$. This is formally stated as follows:

$$\min_{\mathbf{u}} \ \mathcal{H}(\mathbf{u}), \quad \text{s.t. } \mathbf{C}\mathbf{u} \leq \mathbf{d} \tag{22}$$

where $\mathbf{d} = [\mathbf{d}_1; \mathbf{d}_2; \ldots; \mathbf{d}_N]$ and $\mathbf{C}$ can be constructed from the neighborhood constraint matrices $\mathbf{A}_i$ (see supplementary material Section X). Using Equation (22) would result in a much smaller KKT matrix and thus a smaller system of equations to solve for the backward pass gradients. This is what is implied by *cheaper backward pass* on the arrow going from the decentralized problem to the RDCP in Figure 1.

In contrast to the RDCP is the *Reduced Non-Duplicate Centralized Problem* (RNDCP) wherein the objective stays the same as in Equation (22), however, the constraint matrix is now $\bar{\mathbf{C}}$ which has only the unique entries of the constraints forming $\mathbf{C}$. There are two reasons we choose the RDCP over the RNDCP for solving the backward pass:

1) Construction of the non-duplicate constraint matrix $\bar{\mathbf{C}}$ requires checking if mutual neighbors exist for every agent which does not scale for large N,
2) $\mathbf{C}$ can be easily constructed using the matrices $\mathbf{A}_i$ used in the forward pass of Merged CADMM-OSQP.

To justify the usage of (22), we first make the following assumption about the corresponding RNDCP:

Assumption 3. *The control cost matrix $\mathbf{R}$ is positive definite and LICQ holds for the reduced non-duplicate problem, i.e., the active constraints of the matrix $\bar{\mathbf{C}}$ are linearly independent.*

Constraint qualifications are necessary for optimal solutions to constrained optimization problems to satisfy the KKT conditions. LICQ is one of the most frequently used constraint qualification in optimization literature [31]. In our case, we additionally rely on LICQ to ensure that the Lagrange multipliers satisfying the KKT conditions are unique [45, Section 3]. Given that Assumption 3 holds, we can establish the connection between the unique Lagrange multipliers of the RNDCP and those of the RDCP by the following lemma,

Lemma 1. *Given that Assumption 3 holds, then it follows that,*

$$\lambda_{non\text{-}dup,l} = \sum_{j=1}^{\xi_l} \lambda_{dup,\zeta_{l,j}} \quad \forall l \in [\![1, \bar{N}_{ineq}]\!] \tag{23}$$

where $\boldsymbol{\lambda}_{dup}, \boldsymbol{\lambda}_{non\text{-}dup}$ denote the vectors of Lagrange multipliers for the duplicate and non-duplicate problems respectively.

In the following lemma, we derive the KKT system for the RDCP as well as expressions for the gradients of the loss function ℓ with respect to parameters of interest of the Merged CADMM-OSQP Safe Layer for any given every time step t.

Lemma 2 (QP Gradients)**.** *Let $\mathbf{u}^*$ and $\boldsymbol{\lambda}^*_{dup}$ denote the solutions to the reduced duplicate problem at any given time*

step t,

$$\min_{\mathbf{u}(t)} \quad \frac{1}{2}\mathbf{u}(t)^{\mathrm{T}}\mathbf{R}(t)\mathbf{u}(t) + \mathbf{q}(t)^{\mathrm{T}}\mathbf{u}(t) \tag{24}$$

$$s.t. \quad \mathbf{C}(t)\mathbf{u}(t) \leq \mathbf{d}(t) \tag{25}$$

and let ℓ denote the overall neural-network training loss function. Then, the gradients of ℓ with respect to the data matrices $\mathbf{R}, \mathbf{q}, \mathbf{C}, \mathbf{d}$ for the time step t have the form,

$$\nabla_{\mathbf{q}}\ell = \mathrm{d}\mathbf{u} \tag{26a}$$

$$\nabla_{\mathbf{d}}\ell = -\mathrm{diag}(\boldsymbol{\lambda}^*_{dup})\,\mathrm{d}\boldsymbol{\lambda}_{dup} \tag{26b}$$

$$\nabla_{\mathbf{R}}\ell = \frac{1}{2}(\mathrm{d}\mathbf{u}\,\mathbf{u}^{*\mathrm{T}} + \mathbf{u}^*\,\mathrm{d}\mathbf{u}^{\mathrm{T}}) \tag{26c}$$

$$\nabla_{\mathbf{C}}\ell = \boldsymbol{\lambda}^*_{dup}\,\mathrm{d}\mathbf{u}^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\lambda}^*_{dup})\,\mathrm{d}\boldsymbol{\lambda}_{non\text{-}dup}\mathbf{u}^{*\mathrm{T}} \tag{26d}$$

where $\mathrm{d}\mathbf{u}$ and $\mathrm{d}\boldsymbol{\lambda}_{non\text{-}dup}$ are the solutions to the KKT system

$$\begin{bmatrix} \mathbf{R} & \mathbf{C}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}^*_{dup}) \\ \mathbf{C} & \mathrm{diag}(\mathbf{C}\mathbf{u}^* - \mathbf{d}) \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{dup} \end{bmatrix} = \begin{bmatrix} -\nabla_{\mathbf{u}^*}\ell \\ \mathbf{0} \end{bmatrix}. \tag{27}$$

Having derived the KKT system for RDCP in Lemma 2, we can establish a relation between the KKT variables of the RNDCP and those of the RDCP in the following lemma,

Lemma 3. *Given that Assumption 3 holds, then it follows that,*

$$\lambda_{non\text{-}dup,l}\,\mathrm{d}\lambda_{non\text{-}dup,l} = \sum_{j=1}^{\xi_l} \lambda_{dup,\zeta_{l,j}}\,\mathrm{d}\lambda_{dup,\zeta_{l,j}},\ \forall l \in [\![1, \bar{N}_{ineq}]\!] \tag{28}$$

where $\mathrm{d}\boldsymbol{\lambda}_{dup}, \mathrm{d}\boldsymbol{\lambda}_{non\text{-}dup}$ denote the vectors of the variables of the KKT systems for the duplicate and non-duplicate problems respectively.

We invite the interested reader to refer to Section XI in the supplementary material for proofs of the lemmas stated above. We use these lemmas to relate the gradients of ℓ with respect to the parameters of interest of the RNDCP with those of the RDCP with the following theorem,

Theorem 1. *Let $\mathbf{M}$ denote the matrix describing the relationship between the duplicate and non-duplicate constraints:*

$$\mathbf{C} = \mathbf{M}\bar{\mathbf{C}}, \quad \mathbf{d} = \mathbf{M}\bar{\mathbf{d}} \tag{29}$$

Then, for loss ℓ, the gradients $\nabla_{\mathbf{R}}\ell, \nabla_{\mathbf{q}}\ell, \nabla_{\mathbf{C}}\ell, \nabla_{\mathbf{d}}\ell$ coincide for the duplicate and non-duplicate problems and are unique.

Proof Sketch: We first show that the gradients $\nabla_{\mathbf{R}}\ell, \nabla_{\mathbf{q}}\ell, \nabla_{\bar{\mathbf{C}}}\ell, \nabla_{\bar{\mathbf{d}}}\ell$ depend only on $\mathrm{d}\mathbf{u}$ and the sums of $\lambda_{\mathrm{dup},\zeta_{l,j}}$ and $\lambda_{\mathrm{dup},\zeta_{l,j}}\,\mathrm{d}\lambda_{\mathrm{dup},\zeta_{l,j}}$ associated to each unique constraint l of matrix $\bar{\mathbf{C}}$. Applying Lemmas 1 and 3 then shows that the theorem holds. ∎

We refer the reader to Section XII in the supplementary material for the complete proof. Finally, in Section XIII we establish a connection between the optimal solutions of the decentralized problem and the RNDCP similar to that between the RDCP and the RNDCP. This allows us to conclude that the solutions of the decentralized problem must coincide with those of the RDCP and thereby justifies using the solution $(\tilde{\mathbf{u}}^*, \mathbf{y}^*)$ of the decentralized problem computed in the forward pass to formulate the KKT conditions of the RDCP for the computation of the backward pass gradients.

In practice, there may be situations at certain time steps when Assumption 3 is violated, in which case the computed gradient serves as a noisy version of the true gradient.

D. Termination Criteria and Penalty Parameters Adaptation

The algorithm terminates when the norms of the primal and dual residuals get below their corresponding tolerance levels,

$$r_{pri,a} \leq \epsilon_{pri,a}, \quad r_{dual,a} \leq \epsilon_{dual,a}, \quad \forall a = 1, 2.$$

where $a = 1, 2$ refer to the QP and consensus residuals, respectively. Detailed expressions for the residual norms $r_{pri,a}$, $r_{dual,a}$ and the tolerances $\epsilon_{pri,a}$, $\epsilon_{dual,a}$ are provided in Section XV of the supplementary material.

The selection of the penalty parameters ρ_1 and ρ_2 is important since the former encourages the satisfaction of the local constraints of each agent subproblem, while the latter encourages achieving consensus. Low values of these parameters could result to slow convergence of the algorithm, thus requiring a large value of the ADMM maximum iterations parameter $l_{\max}$. On the other hand, if their values are too high then their corresponding terms in (17) will dominate the objective function. From a practical standpoint, we accommodate for these issues by adopting the following adaptation schemes for the penalty parameters:

$$\rho_1^{l+1} = \rho_1^l \sqrt{\frac{r^l_{pri,1}/\kappa^l_{pri,1}}{r^l_{dual,1}/\kappa^l_{dual,1}}}, \quad \rho_2^{l+1} = \rho_2^l \sqrt{\frac{r^l_{pri,2}/\kappa^l_{pri,2}}{r^l_{dual,2}/\kappa^l_{dual,2}}}.$$

This scheme is inspired by the adaptation rules that are used in the OSQP solver [43, Section 5.2]. Note that the specific choice of termination criteria and adaptation rules employed in this work would require computation being performed by a central node. Future work aims at proposing termination criteria and adaptation rules that can be performed in a fully decentralized manner as well.

E. Additional Constraints for Local Subproblems

In the algorithm proposed in Section V, achieving consensus—and thus ensuring the safety of the agents—fully relies on ADMM. To facilitate reaching a consensus, we suggest including in every subproblem of agent i: i) the obstacle avoidance constraints of its neighbors, and/or ii) the inter-agent constraints between its neighbors. The number of these additional constraints will be rN_o and $\binom{r}{2}$, respectively. Even after incorporating these constraints, we emphasize that, for $r \ll N$, the new $N_{\mathrm{ineq},i} = r + N_o + rN_o + \binom{r}{2}$ constraints in each agent's local QP will still be substantially smaller than the $N_{\mathrm{ineq}} = \binom{N}{2} + NN_o$ of the centralized problem.

VII. Simulation Results

We test the proposed approach on a system consisting of multiple agents with unicycle dynamics (similar to [48, Section V.B]) for any agent i given by,

$$\dot{x}_i = v_i\cos(\theta_i), \quad \dot{y}_i = v_i\sin(\theta_i), \quad \dot{\theta}_i = v_i u_i^{\theta}, \quad \dot{v}_i = u_i^{v}.$$

Constructing a state vector $\mathbf{x}_i = [x_i;\, y_i;\, \theta_i;\, v_i]$ and a control vector $\mathbf{u}_i = [u_i^\theta;\, u_i^v]$ and assuming that noise only enters the acceleration channels, the stochastic dynamics for agent i can be written as,

$$\mathrm{d}\mathbf{x}_i = \mathbf{f}_i \mathrm{d}t + \mathbf{G}_i \mathbf{u}_i \mathrm{d}t + \mathbf{\Sigma}_i \mathrm{d}\mathbf{w}_i$$

where $\mathbf{f}_i$, $\mathbf{G}_i$, and, $\mathbf{\Sigma}_i$ are given by,

$$\mathbf{f}_i = \begin{bmatrix} v_i \cos(\theta_i) \\ v_i \sin(\theta_i) \\ 0 \\ 0 \end{bmatrix}, \mathbf{G}_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ v_i & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{\Sigma}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma & 0 \\ 0 & 0 & 0 & \sigma \end{bmatrix}.$$

For our simulations we consider obstacle avoidance and collision avoidance safety constraints in four different types of tasks. These constraints are stated as functions of the positions of the agents and are therefore relative degree 2 (i.e., one needs to differentiate twice before the control shows up). However, the aforementioned SCBF in (7) assumes that the relative degree of h is 1. This is required to ensure that $\frac{\partial B}{\partial \mathbf{x}}^{\mathrm{T}} \mathbf{G}$ is not zero. Therefore, the original position constraint, which we hereon refer to as $h_{\text{pos}}(\mathbf{x})$, must be modified to ensure that the modified function has relative degree 1. In [35], this modification takes the following form,

$$h(\mathbf{x}) = h_{\text{pos}}(\mathbf{x}) - \mu\, v^2. \tag{30}$$

The parameter μ controls how fast the system can safely move inside the safe set and thereby, the implication of adding the $-\mu\, v^2$ term introduces constraints on the velocity in addition to the original position constraint. Intuitively, when $h_{\text{pos}} = 0$, for the system to stay safe (i.e., $h \geq 0$), the only allowable safe velocity is $v = 0$.

The main drawbacks of (30) are that it does not take into account the heading of the agents and that it penalizes positive and negative velocities equally. To overcome these we propose the following two types of barrier functions,

1) **Type-A**: This is a pairwise safety constraint concerning two agents and is constructed as follows:

$$h^A(\mathbf{x}) = h^A_{\text{pos}}(\mathbf{x}) - \mu\,(v_i\, \text{IP}_i + v_j\, \text{IP}_j) \tag{31}$$

where, $h^A_{\text{pos}}(\mathbf{x}) = \frac{1}{2}\Big((x_i - x_j)^2 + (y_i - y_j)^2 - 4r^2\Big)$,

$\text{IP}_i = \bar{p}_{ij}^{\mathrm{T}} \bar{\theta}_i$, and, $\text{IP}_j = \bar{p}_{ji}^{\mathrm{T}} \bar{\theta}_j$

for any two agents i and j, each with a radius of r. The inner-product (IP) terms depend on vectors $\bar{p}_{ij}$ and $\bar{p}_{ji}$ which denote relative position vectors from i to j and j to i respectively and on the vectors $\bar{\theta}_i = [\cos\theta_i;\, \sin\theta_i]$ and $\bar{\theta}_j = [\cos\theta_j;\, \sin\theta_j]$ which denote unit vectors along the headings of agents i and j respectively.

2) **Type-B**: This type of safety constraint concerns an agent i and an obstacle o. It is constructed as follows,

$$h^B(\mathbf{x}) = h^B_{\text{pos}}(\mathbf{x}) - \mu\, v_i\, \text{IP} \tag{32}$$

where,

$$h^B_{\text{pos}}(\mathbf{x}) = \frac{1}{2}\Big((x_i - x_o)^2 + (y_i - y_o)^2 - (r_i + r_o)^2\Big),$$

$$\text{and IP} = \bar{p}_{io}^{\mathrm{T}} \theta_i \tag{33}$$

where, (x_o, y_o) is the position of the obstacle's center and r_o is the obstacle's radius. The vectors $\bar{p}_{io}$ and θ_i are defined similar to the type-A constraint above.

Similar to work in [35] our simulations are also conducted in a safe reinforcement learning setting where it is required to be safe not only during inference but also during the entire training process. The reader is encouraged to refer to the video[1] to support this claim. The video shows the gradual emergence of optimal behavior as iterations progress, for each of the tasks described below, while staying safe during the entire training process. Additionally, it depicts the behavior of a single batch instance on the left frame and distributions over entire batches on the right frame wherein agents are depicted as particles. The frame on the right also demonstrates successful performance on average as justified by our choice of the mean (i.e., expected value) cost function (3).

A. Swapping Task

We first consider the swapping cars task presented in [35]. We show that our proposed approach is not only scalable to larger teams of cars but can handle the added complexity of obstacle avoidance. Each agent's goal is to swap positions with the diametrically opposite one while avoiding collisions. In Fig. 4a, the distributions of the positions show that the agents avoid collisions with a high probability. On closer inspection, one can argue that the problem is highly symmetrical as no matter where you stand on the initial circle, each agent effectively solves a similar problem. To prove that our approach can handle more complex scenarios, we added extra obstacles to create an asymmetrical version of the same problem. The final policy is depicted in Fig. 4b. As seen in the figure, the cars on the top right quadrant encounter obstacles much sooner along with encountering their neighbors. Observing the second plot of Fig. 4b, we see that the final policy now leads to the agents circling around the new highly non-convex obstacle shape in order to complete the task. For all swapping tasks, we used a neighborhood size of $r = 3$. For the symmetrical obstacle task, we used r type-A and 1 type-B constraints in every neighborhood. While for the unsymmetrical task, we used $\binom{r}{2}$ type-A and $N_o(r+1)$ type-B constraints in every neighborhood.

B. Bottleneck Task

For this task, the agents are required to pass through the bottleneck in the center (created by multiple circular obstacles) and achieve a desired formation on the other side of the bottleneck as seen in Fig. 5a. To train this policy we designed the running cost $c_i(\mathbf{x})$, and terminal cost $\phi_i(\mathbf{x})$, such that the task is divided into two objectives - (i.) pass through the bottleneck, and, (ii.) achieve the desired formation. This was done to "encourage" the agents to prioritize passing through the bottleneck, before attempting to achieve the desired formation on the other side. Without the first objective, some agents may end up getting stuck in the highly non-convex regions between

[1] https://youtu.be/qjPLUlaxJos

the circular obstacles. Thus, it helps to avoid these undesirable local minima. This objective was achieved by setting a target for the agents in the column close to the bottleneck at a point on the x-axis further away from the bottleneck and setting a target for the agents in the second column at a point on the x-axis close to the bottleneck. This *intermediate target* was set for 80% of the time horizon. For the remaining 20%, the targets were set so as to achieve the desired formation on the other side. For this task, we chose $r = 3$ with each neighborhood containing r type-A constraints between ego agent and neighbors and 6 type-B constraints between the ego agent and the obstacles.

C. Moving-obstacle (or uncooperative agent) Task

The goal of this task is the same as the swapping task with the added complexity of a moving obstacle. The moving obstacle can be interpreted as an uncooperative agent (i.e., one that cannot be controlled) and moves from bottom to top along the y-axis as seen in Fig. 5b. To train this policy, we first trained a policy containing the 8 agents without any obstacle on the swapping task. This pre-trained policy was then used as an initialization to train the policy to swap while avoiding an oncoming moving obstacle. The reason for this two step process being that a completely random initial policy is unable to safely explore in the presence of the moving obstacle. A common scenario that arises when one attempts to train the policy from scratch is that the agents try to directly move to their diametrically opposite targets but then come to a halt and congregate around the center because they encounter other agents. However, as the moving obstacle approaches, the closest agent to the obstacle is inevitably "run-over" by the moving obstacle, as it has no where to escape. Using a pre-trained policy embeds the agents with the ability to "move away from the moving obstacle's path" as they have pre-learned a behavior to initiate a coordinated turn in order to avoid colliding with other agents. For this task, we used a neighborhood size of $r = 3$ with r type-A constraints between the ego-agent and its neighbors and $r + 1$ type-B constraints for obstacle avoidance.

D. Large-Scale Formation Task

Here, we demonstrate the scalability of our framework by considering a task where a large-scale team of 32 agents must achieve a desired rectangular formation. Each agent has $r = 6$ neighbors and each local problem includes r type-A constraints and $N_o(r + 1) = 14$ type-B constraints. As shown in Fig. 5c, the distributions of the positions of agents successfully reach close to the desired targets.

VIII. Discussion

A. Constraints Reduction and Increased Memory Efficiency

In Table I, we compare the number of constraints considered by the centralized and decentralized approaches for each task presented in Section VII. Clearly, there is a substantial reduction on the number of constraints when using the proposed approach, implying that it is more scalable to large-scale systems than the equivalent centralized one. Table II also demonstrates the reduced required memory usage and training iteration time of our approach.

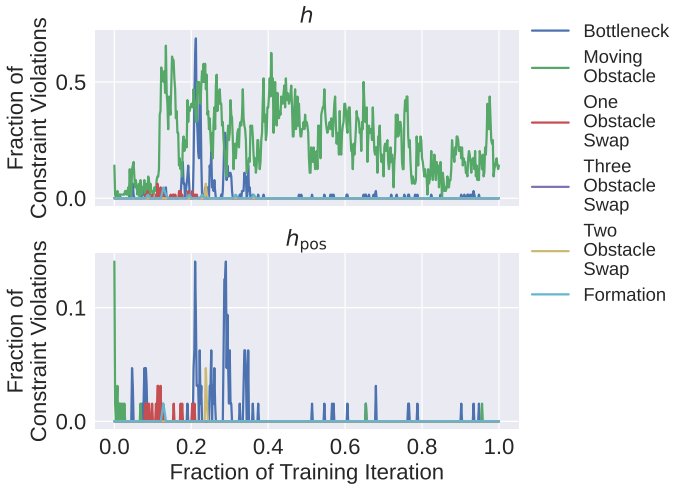


Fig. 3: Constraint Violations

B. Low Position Constraint Violation

In Figure 3, the top and bottom plots show the fraction of batch instances where $h < 0$ and $h_{\text{pos}} < 0$, respectively, against the number of training iterations. For all tasks, $h_{\text{pos}} < 0$ occurs much less frequently as compared to $h < 0$. This indicates that although there are many instances of h violations (i.e., agents moving with *unsafe* velocities in the vicinity of other agents or other obstacles), the number of physical collisions between agents or between agents and obstacles are much lower or even zero for some tasks. Thus, the new type-A and type-B barrier formulations allow for more aggressive behavior as compared to (30) which instead encourages conservative behaviors and therefore cannot scale for large numbers of agents. Another interesting observation is the sharp decline in h_{pos} violations after around 40% of the total number of iterations. An intuitive explanation of this is that position violations only occur during the initial exploration phase.

IX. Conclusion

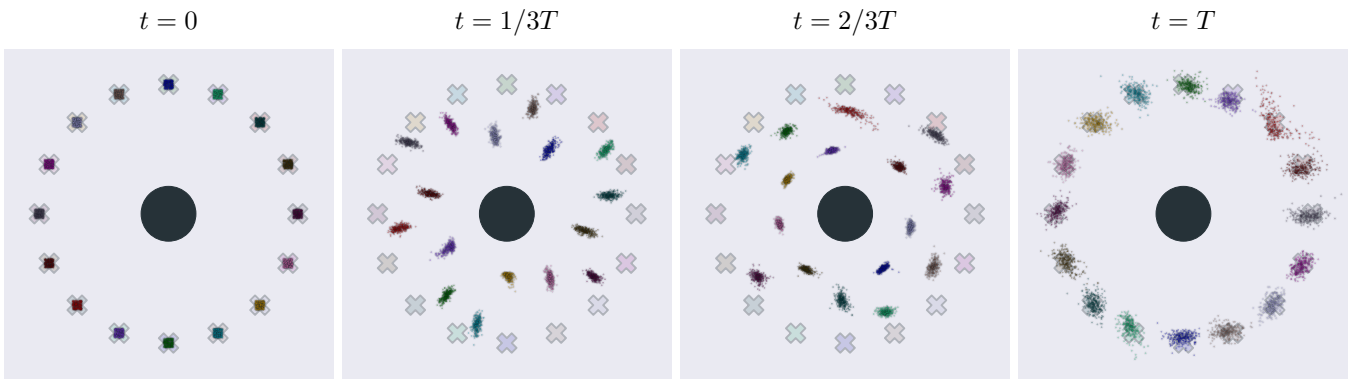
In this work, we introduced a novel and scalable deep-learning-based framework that extends the existing deep FBSDE framework to decentralized multi-agent safe stochastic optimal control problems. To achieve this, we proposed a new stochastic CBF formulation which encourages aggressive motion of moving agents, while still guaranteeing their safe operation with a high probability. Furthermore, we reformulated the multi-agent per-time-step Hamiltonian minimization problem into a decentralized version, which we solve by proposing a novel distributed ADMM-based method for large-scale quadratic optimization problems. The framework was successfully tested in simulation on several challenging multi-vehicle tasks with higher complexity and more agents as compared to previous work using a centralized approach and simpler tasks. The results demonstrate that the agents maintain

Task	Number of Agents	Decentralized	Centralized
Swapping	16	5	136
Bottleneck	8	10	76
Moving obstacle	8	9	36
Large-scale formation	32	16	560

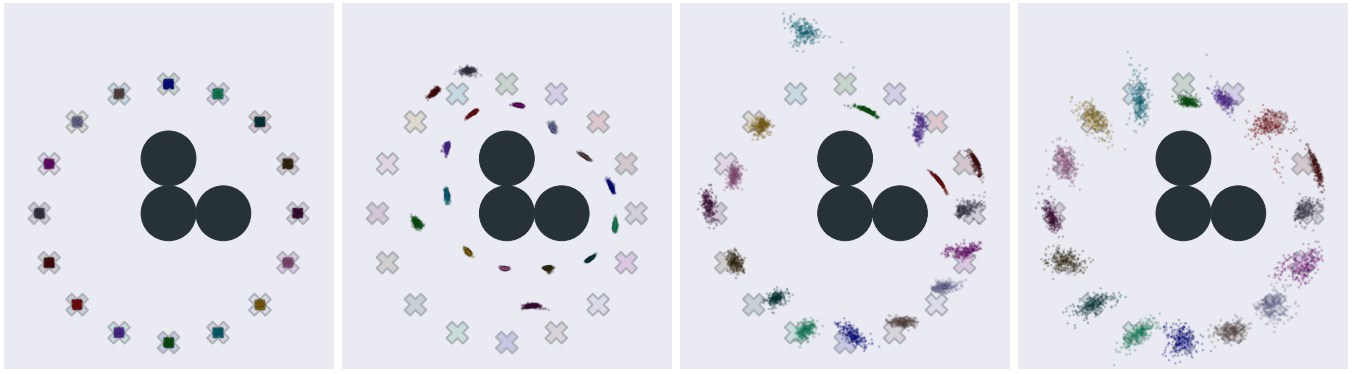
TABLE I: Comparison of number of constraints between the centralized problem and each local subproblem of the proposed decentralized approach.

	Max Memory Allocated (MiB)			Time per Training Iteration (s)		
Batch Size	Decentralized	Centralized	Percent Reduction	Decentralized	Centralized	Percent Reduction
32	1586	17879	-91.13%	14.75	1306	-98.87%
64	3124	N/A	N/A	17.92	N/A	N/A
384	18510	N/A	N/A	38.30	N/A	N/A

TABLE II: Comparison of memory usage (maximum of 10 iterations) and time per iteration (average over 10 iterations) between the decentralized and centralized formulations. The N/A values for batch sizes 64 and 384 indicate that the computer ran out of memory and was unable to run a single iteration.

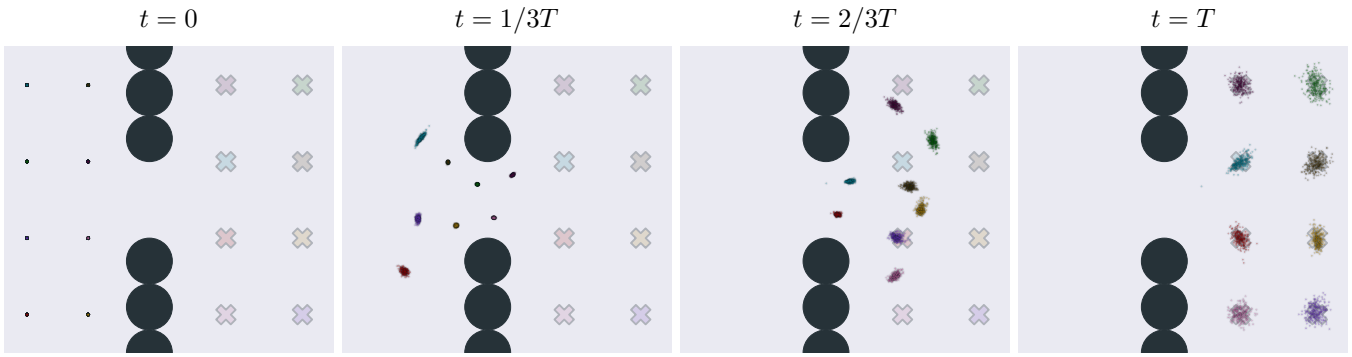


(a) Symmetrical Swapping Task with One Obstacle and 16 Agents.

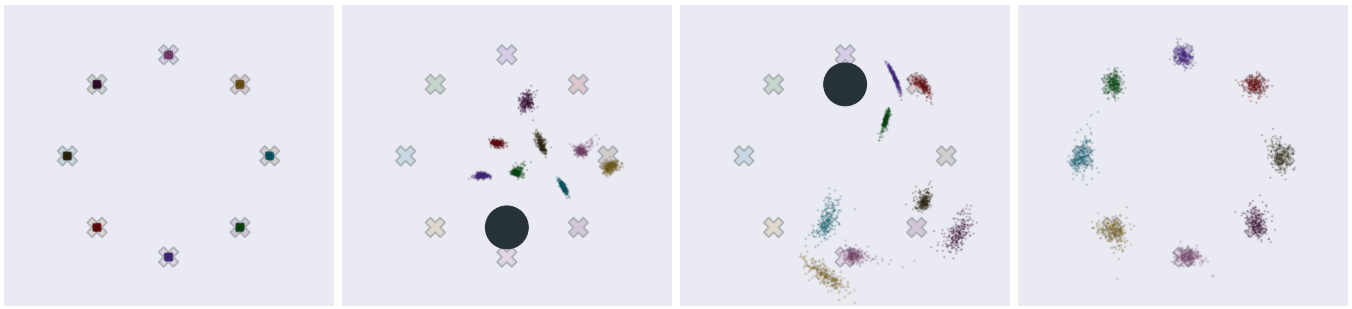


(b) Unsymmetrical Swapping Task with Three Obstacles and 16 Agents.

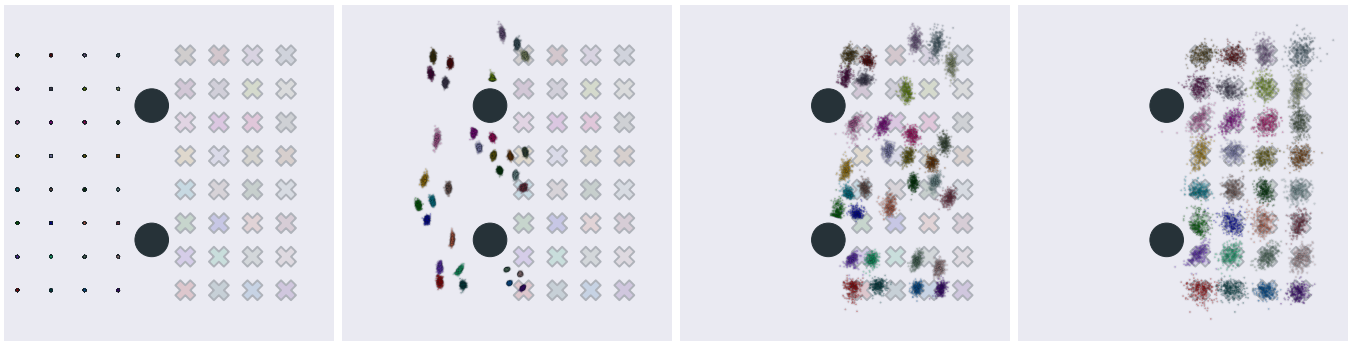
Fig. 4: In all figures, the black circles indicate obstacles and the **X** markers indicate target positions for each agent with the same color as the marker. The snapshots demonstrate the positions of each agent for each batch at time instants $t = 0, 1/3T, 2/3T, T$ with the fully trained policy. The proposed approach can not only handle symmertical problems similar to [35], but can also successfully solve unsymmertical problems.



(a) Bottleneck task with 8 agents.



(b) Swapping task with 8 agents and a moving obstacle (uncooperative agent).



(c) Large-scale formation task with 32 agents.

Fig. 5: Similar to Fig. 4, black circles indicate obstacles and the **X** markers indicate target positions. The top and bottom rows involve static obstacles, while the center row considers a moving obstacle.

their safe operation during training, thus the framework can also be appreciated from a safe reinforcement learning perspective. Finally, the scalability of the approach in terms of memory usage and training time is also validated.

Acknowledgments

This research was supported by the ARO Award #W911NF2010151. Augustinos Saravanos acknowledges financial support by the A. Onassis Foundation Scholarship.

References

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019.

[2] Javier Alonso-Mora, Eduardo Montijano, Mac Schwager, and Daniela Rus. Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 5356–5363. IEEE, 2016.

[3] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[4] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[5] Brandon Amos and J Zico Kolter. Optnet: Differentiable

optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[6] Brandon Amos and Denis Yarats. The differentiable cross-entropy method. In *International Conference on Machine Learning*, pages 291–302. PMLR, 2020.

[7] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *arXiv preprint arXiv:1810.13400*, 2018.

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

[9] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[10] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

[11] Tianrong Chen, Ziyi O Wang, Ioannis Exarchos, and Evangelos Theodorou. Large-scale multi-agent deep fbsdes. In *International Conference on Machine Learning*, pages 1740–1748. PMLR, 2021.

[12] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[13] Zilong Cheng, Jun Ma, Xiaoxue Zhang, Clarence W de Silva, and Tong Heng Lee. Admm-based parallel optimization for multi-agent collision-free model predictive control. *arXiv preprint arXiv:2101.09894*, 2021.

[14] Andrew Clark. Control barrier functions for stochastic systems. *Automatica*, 130:109688, 2021.

[15] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

[16] Ioannis Exarchos and Evangelos A Theodorou. Stochastic optimal control via forward and backward stochastic differential equations and importance sampling. *Automatica*, 87:159–165, 2018.

[17] Ioannis Exarchos, Evangelos A Theodorou, and Panagiotis Tsiotras. Game-theoretic and risk-sensitive stochastic optimal control via forward and backward stochastic differential equations. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6154–6160. IEEE, 2016.

[18] Ioannis Exarchos, Evangelos A Theodorou, and Panagiotis Tsiotras. Stochastic l1-optimal control via forward and backward sampling. *Systems & Control Letters*, 118: 101–108, 2018.

[19] Ioannis Exarchos, Evangelos Theodorou, and Panagiotis Tsiotras. Stochastic differential games: A sampling approach via fbsdes. *Dynamic Games and Applications*, 9(2):486–505, 2019.

[20] Ioannis Exarchos, Marcus A Pereira, Ziyi Wang, and Evangelos A Theodorou. Novas: Non-convex optimization via adaptive stochastic search for end-to-end learning and control. *arXiv preprint arXiv:2006.11992*, 2020.

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[22] Trevor Halsted, Ola Shorinwa, Javier Yu, and Mac Schwager. A survey of distributed optimization methods for multi-robot systems. *arXiv preprint arXiv:2103.12840*, 2021.

[23] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Harold J Kushner. Stochastic stability and control. Technical report, Brown Univ Providence RI, 1967.

[26] Viet-Anh Le and Truong X Nghiem. Gaussian process based distributed model predictive control for multi-agent systems using sequential convex programming and ADMM. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 31–36. IEEE, 2020.

[27] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

[28] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[29] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[31] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[32] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[33] Marcus Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Learning deep stochastic optimal control policies using forward-backward sdes. *arXiv preprint arXiv:1902.03986*, 2019.

[34] Marcus Pereira, Ziyi Wang, Tianrong Chen, Emily Reed, and Evangelos Theodorou. Feynman-kac neural network architectures for stochastic control using second-order

fbsde theory. In *Learning for Dynamics and Control*, pages 728–738. PMLR, 2020.

[35] Marcus Aloysius Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Safe optimal control using stochastic barrier functions and deep forward-backward sdes. *arXiv preprint arXiv:2009.01196*, 2020.

[36] Cesar Santoyo, Maxence Dutreix, and Samuel Coogan. A barrier function approach to finite-time stochastic system verification and control. *Automatica*, 125:109439, 2021.

[37] Augustinos D Saravanos, Alexandros Tsolovikos, Efstathios Bakolas, and Evangelos Theodorou. Distributed Covariance Steering with Consensus ADMM for Stochastic Multi-Agent Systems. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.075.

[38] Meenakshi Sarkar, Debasish Ghose, and Evangelos A Theodorou. High-relative degree stochastic control lyapunov and barrier functions. *arXiv preprint arXiv:2004.03856*, 2020.

[39] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

[40] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[41] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[42] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12 (4):637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL https://doi.org/10.1007/s12532-020-00179-2.

[43] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[44] Wentao Tang and Prodromos Daoutidis. Fast and stable nonconvex constrained distributed optimization: the ellada algorithm. *Optimization and Engineering*, pages 1–43, 2021.

[45] Gerd Wachsmuth. On licq and the uniqueness of lagrange multipliers. *Operations Research Letters*, 41(1):78–80, 2013.

[46] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[47] Ziyi Wang, Keuntaek Lee, Marcus A Pereira, Ioannis Exarchos, and Evangelos A Theodorou. Deep forward-backward sdes for min-max control. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6807–6814. IEEE, 2019.

[48] Zhaoming Xie, C Karen Liu, and Kris Hauser. Differential dynamic programming with nonlinear constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 695–702. IEEE, 2017.

[49] Shakiba Yaghoubi, Keyvan Majd, Georgios Fainekos, Tomoya Yamaguchi, Danil Prokhorov, and Bardh Hoxha. Risk-bounded control using stochastic barrier functions. *IEEE Control Systems Letters*, 5(5):1831–1836, 2020.

SUPPLEMENTARY MATERIAL

In this section, we provide additional derivations and details not covered in the main paper.

X. CONSTRUCTION OF CONTRAINT MATRIX FOR REDUCED DUPLICATE CENTRALIZED PROBLEM

For convenience of derivation and notation, it is assumed that the agents are homogeneous, i.e., all agents obey to the same dynamics with $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{u}_i \in \mathbb{R}^m$. For the same reasons, we will only assume the existence of inter-agent constraints - neglecting obstacle avoidance constraints. Nevertheless, the following derivation can be easily extended beyond these assumptions.

Let us start by restating that in the RDCP we only take into account the $NN_{\text{ineq},i} = Nr$ inter-agent constraints that appear between neighboring agents out of the total N_{ineq} constraints. This problem can be formulated as follows:

$$\min_{\mathbf{u}} \mathcal{H}(\mathbf{u}) = \sum_{i=1}^{N} \mathcal{H}_i(\tilde{\mathbf{u}}_i) \quad \text{s.t. } \mathbf{C}\mathbf{u} \leq \mathbf{d} \tag{34}$$

where $\mathcal{H}(\mathbf{u}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u} + \mathbf{q}^{\mathrm{T}}\mathbf{u}$ with $\mathbf{q} \coloneqq \mathbf{G}^{\mathrm{T}}\frac{\partial V}{\partial \mathbf{x}}$. Regarding the construction of matrix $\mathbf{C}$, let us first express all $\mathbf{A}_i \in \mathbb{R}^{r\times(r+1)m}$, $i \in [\![1, N]\!]$ as block matrices consisting of $r \times (r+1)$ blocks as follows:

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{a}_i^{1,1} & \mathbf{a}_i^{1,2} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{a}_i^{2,1} & \mathbf{0} & \mathbf{a}_i^{2,2} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_i^{r,1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{a}_i^{r,2} \end{bmatrix} \tag{35}$$

where each block $\mathbf{a}_i^{k,b} \in \mathbb{R}^m$, $k = [\![1, r]\!]$, $b = 1, 2$ is a row vector. The blocks $\mathbf{a}_i^{k,1}$ are the ones always multiplied with the ego part $\mathbf{u}_i$ of $\tilde{\mathbf{u}}_i$, while the blocks $\mathbf{a}_i^{k,2}$ correspond to the neighbors parts $\mathbf{u}_j^{(i)}$, $j \in \mathcal{N}_i$ of $\tilde{\mathbf{u}}_i$. Based on that, $\mathbf{C} = [\mathbf{C}_1; \dots; \mathbf{C}_N]$, where each $\mathbf{C}_i \in \mathbb{R}^{r\times Nm}$ can be viewed as a block matrix consisting of $r \times N$ blocks of size $\mathbb{R}^m$, and is constructed as follows: for all $k = [\![1, r]\!]$, i) each $\mathbf{A}_i[k, 1]$ (i.e., $\mathbf{a}_i^{k,1}$) is assigned to block $\mathbf{C}_i[k, i]$ and ii) each $\mathbf{A}_i[k, l]$ (i.e., $\mathbf{a}_i^{k,2}$) to $\mathbf{C}_i[k, j]$ with $l \in [\![2, r + 1]\!]$.

XI. Proof of Lemmas

Let $\mathcal{D} \coloneqq \{\mathcal{D}_l\}_{l=1}^{\bar{N}_{\text{ineq}}}$ with $\mathcal{D}_l \coloneqq \{\mathbf{c}_{\zeta_{l,j}}\}_{j=1}^{\xi_l}$ denote the set of equivalence classes induced by the equality equivalence relation

$$\mathbf{c}_{\zeta_{l,a}} \sim \mathbf{c}_{\zeta_{l,b}} \iff \mathbf{c}_{\zeta_{l,a}} = \mathbf{c}_{\zeta_{l,b}}, \quad \forall a, b \in [\![1, \xi_l]\!] \; \forall l \tag{36}$$

where $\zeta_{l,j}$ denotes the row corresponding to the j-th instance of the l-th unique constraint. In other words, there are ξ_l duplicates for each of the $\bar{N}_{\text{ineq}}$ unique constraint rows.

We will now proceed with proving Lemma 1, which we restate for the convenience of the reader.

Lemma 1. *Given that Assumption 3 holds, then it follows that,*

$$\lambda_{\textit{non-dup},l} = \sum_{j=1}^{\xi_l} \lambda_{\textit{dup},\zeta_{l,j}} \quad \forall l \in [\![1, \bar{N}_{\textit{ineq}}]\!] \tag{37}$$

where $\boldsymbol{\lambda}_{\textit{dup}}, \boldsymbol{\lambda}_{\textit{non-dup}}$ denote the vectors of Lagrange multipliers for the duplicate and non-duplicate problems respectively.

Proof: Let $\zeta_l \coloneqq \zeta_{l,1}$ denote a representative of the l-th equivalence class, where $c_{\zeta_l} = c_{\zeta_{l,j}}$ for all l, j. From the definition of $\zeta_{l,j}$, we can represent the non-duplicate $\bar{\mathbf{C}}$ and $\bar{\mathbf{d}}$ as,

$$\bar{\mathbf{C}} = [\mathbf{c}_{\zeta_1}; \ldots; \mathbf{c}_{\zeta_{\bar{N}_{\text{ineq}}}}] \implies \bar{\mathbf{c}}_l = \mathbf{c}_{\zeta_l} \tag{38}$$

$$\bar{\mathbf{d}} = [\mathbf{d}_{\zeta_1}; \ldots; \mathbf{d}_{\zeta_{\bar{N}_{\text{ineq}}}}] \implies \bar{\mathbf{d}}_l = \mathbf{d}_{\zeta_l}. \tag{39}$$

As we do not assume that agents will be mutual neighbors of each other, note that $\mathcal{D}$ may contain singleton sets, i.e., there may be constraints rows that are unique in $\mathbf{C}$.

The Lagrangian for the duplicate problem is given by,

$$\mathcal{L}_{\text{dup}} = \mathcal{H}(\mathbf{u}) + \boldsymbol{\lambda}_{\text{dup}}^{\mathrm{T}}(\mathbf{C}\mathbf{u} - \mathbf{d}) \tag{40}$$

where we remind the reader that matrices for the duplicate problem have dimension $\mathbf{C} \in \mathbb{R}^{Nr \times Nm}$ and $\mathbf{d} \in \mathbb{R}^{Nr}$, where based on Assumption 2. Since the constraint rows of each equivalence class are identical, by representing the matrix multiplication with $\mathbf{C}^{\mathrm{T}}$ as a sum over the rows of $\mathbf{C}$ we can factor out $\mathbf{c}_{\zeta_l}$ over each equivalence class, yielding

$$\begin{aligned}
\mathcal{L}_{\text{dup}} &= \mathcal{H}(\mathbf{u}) + \boldsymbol{\lambda}_{\text{dup}}^{\mathrm{T}}(\mathbf{C}\mathbf{u} - \mathbf{d}) && (41)\\
&= \mathcal{H}(\mathbf{u}) + \sum_{k=1}^{Nr} \lambda_{\text{dup},k}\,(\mathbf{c}_k\mathbf{u} - d_k) && (42)\\
&= \mathcal{H}(\mathbf{u}) + \left(\sum_{k=1}^{Nr} \lambda_{\text{dup},k}\mathbf{c}_k\right)\mathbf{u} - \sum_{k=1}^{Nr} \lambda_{\text{dup},k}\, d_k && (43)\\
&= \mathcal{H}(\mathbf{u}) + \left(\sum_{l=1}^{\bar{N}_{\text{ineq}}} \left(\sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\right)\mathbf{c}_{\zeta_l}\right)\mathbf{u} \\
&\quad - \sum_{l=1}^{\bar{N}_{\text{ineq}}} \left(\sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\right) d_{\zeta_l} && (44)\\
&= \mathcal{H}(\mathbf{u}) + \left(\sum_{l=1}^{\bar{N}_{\text{ineq}}} \nu_l \bar{\mathbf{c}}_l\right)\mathbf{u} - \sum_{l=1}^{\bar{N}_{\text{ineq}}} \nu_l \bar{d}_l && (45)\\
&= \mathcal{H}(\mathbf{u}) + \boldsymbol{\nu}^{\mathrm{T}}(\bar{\mathbf{C}}\mathbf{u} - \bar{\mathbf{d}}) && (46)
\end{aligned}$$

where we have defined the vector $\boldsymbol{\nu}$ such that

$$\nu_l = \sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}. \tag{47}$$

On the other hand, the Lagrangian for the non-duplicate problem reads,

$$\mathcal{L}_{\text{non-dup}} = \mathcal{H}(\mathbf{u}) + \boldsymbol{\lambda}_{\text{non-dup}}^{\mathrm{T}}(\bar{\mathbf{C}}\mathbf{u} - \bar{\mathbf{d}}). \tag{48}$$

We now assume that the control cost matrix, $\mathbf{R}$, is positive definite for our problem which is a standard assumption for stochastic optimal control. Under LICQ, the Lagrange multipliers corresponding to the saddle point of $\mathcal{L}_{\text{non-dup}}$ are unique. Thus, comparing (46) and (48), we must have that $\boldsymbol{\nu} = \boldsymbol{\lambda}_{\text{non-dup}}$, i.e.,

$$\lambda_{\text{non-dup},l} = \nu_l = \sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}} \quad \forall l \in [\![1, \bar{N}_{\text{ineq}}]\!]. \tag{49}$$

Thus, (37) holds. ∎

Lemma 2 (QP Gradients)**.** *Let $\mathbf{u}^*$ and $\boldsymbol{\lambda}^*_{\textit{dup}}$ denote the solutions to the reduced duplicate problem at any given time step t,*

$$\min_{\mathbf{u}(t)} \quad \frac{1}{2}\mathbf{u}(t)^{\mathrm{T}}\mathbf{R}(t)\mathbf{u}(t) + \mathbf{q}(t)^{\mathrm{T}}\mathbf{u}(t) \tag{50}$$

$$\textit{s.t.} \quad \mathbf{C}(t)\mathbf{u}(t) \le \mathbf{d}(t) \tag{51}$$

and let ℓ denote the overall neural-network training loss function. Then, the gradients of ℓ with respect to the data matrices $\mathbf{R}, \mathbf{q}, \mathbf{C}, \mathbf{d}$ for the time step t have the form,

$$\nabla_{\mathbf{q}}\ell = \mathrm{d}\mathbf{u} \tag{52a}$$

$$\nabla_{\mathbf{d}}\ell = -\mathrm{diag}(\boldsymbol{\lambda}^*_{\textit{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\textit{dup}} \tag{52b}$$

$$\nabla_{\mathbf{R}}\ell = \frac{1}{2}(\mathrm{d}\mathbf{u}\,\mathbf{u}^{*\mathrm{T}} + \mathbf{u}^*\,\mathrm{d}\mathbf{u}^{\mathrm{T}}) \tag{52c}$$

$$\nabla_{\mathbf{C}}\ell = \boldsymbol{\lambda}^*_{\textit{dup}}\,\mathrm{d}\mathbf{u}^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\lambda}^*_{\textit{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\textit{non-dup}}\mathbf{u}^{*\mathrm{T}} \tag{52d}$$

where $\mathrm{d}\mathbf{u}$ and $\mathrm{d}\boldsymbol{\lambda}_{\textit{non-dup}}$ are the solutions to the KKT system

$$\begin{bmatrix} \mathbf{R} & \mathbf{C}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}^*_{\textit{dup}}) \\ \mathbf{C} & \mathrm{diag}(\mathbf{C}\mathbf{u}^* - \mathbf{d}) \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{\textit{dup}} \end{bmatrix} = \begin{bmatrix} -\nabla_{\mathbf{u}^*}\ell \\ \mathbf{0} \end{bmatrix}. \tag{53}$$

Proof: The KKT conditions for the reduced duplicate problem are given by,

$$\mathbf{R}\mathbf{u}^* + \mathbf{q} + \mathbf{C}^{\mathrm{T}}\boldsymbol{\lambda}^*_{\text{dup}} = \mathbf{0} \tag{54a}$$
$$\operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})(\mathbf{C}\mathbf{u}^* - \mathbf{d}) = \mathbf{0} \tag{54b}$$
$$\boldsymbol{\lambda}^*_{\text{dup}} \geq \mathbf{0} \tag{54c}$$
$$\mathbf{C}\mathbf{u}^* \leq \mathbf{d}. \tag{54d}$$

Next, let us define:

$$\boldsymbol{\epsilon}(\boldsymbol{\delta}, \theta) = \begin{bmatrix} \mathbf{R}\mathbf{u}^* + \mathbf{q} + \mathbf{C}^{\mathrm{T}}\boldsymbol{\lambda}^*_{\text{dup}} \\ \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})(\mathbf{C}\mathbf{u}^* - \mathbf{d}) \end{bmatrix} \tag{55}$$

where $\boldsymbol{\delta} = [\mathbf{u}^*; \boldsymbol{\lambda}^*_{\text{dup}}]$ and θ denotes any parameter in (55), i.e., $\mathbf{R}, \mathbf{q}, \mathbf{C}, \mathbf{d}$. To obtain the gradient of the training loss ℓ with respect to the parameters θ, we have that,

$$\nabla_\theta \ell = (\nabla_\theta \boldsymbol{\delta})^{\mathrm{T}} (\nabla_{\boldsymbol{\delta}} \mathbf{u}^*)^{\mathrm{T}} \nabla_{\mathbf{u}^*} \ell \tag{56}$$

which leads to,

$$\begin{aligned} (\nabla_\theta \ell)^{\mathrm{T}} &= (\nabla_{\mathbf{u}^*} \ell)^{\mathrm{T}} \nabla_{\boldsymbol{\delta}} \mathbf{u}^* \nabla_\theta \boldsymbol{\delta} \\ &= (\nabla_{\mathbf{u}^*} \ell)^{\mathrm{T}} \nabla_{\boldsymbol{\delta}} \mathbf{u}^* \underbrace{\left(- \nabla_{\boldsymbol{\delta}} \boldsymbol{\epsilon}(\boldsymbol{\delta}, \theta)^{-1} \nabla_\theta \boldsymbol{\epsilon}(\boldsymbol{\delta}, \theta) \right)}_{\text{using the Implicit Function Theorem}} \end{aligned} \tag{57}$$

Next, we write the above equation as follows,

$$\nabla_\theta \ell^{\mathrm{T}} = \boldsymbol{\omega}\, \nabla_\theta \boldsymbol{\epsilon}(\boldsymbol{\delta}, \theta) \tag{58}$$

where we define $\boldsymbol{\omega}$ as,

$$\boldsymbol{\omega} = \begin{bmatrix} \mathrm{d}\mathbf{u}^{\mathrm{T}} & \mathrm{d}\boldsymbol{\lambda}^{\mathrm{T}}_{\text{dup}} \end{bmatrix} = -(\nabla_{\mathbf{u}^*} \ell)^{\mathrm{T}} (\nabla_{\boldsymbol{\delta}} \mathbf{u}^*) \nabla_{\boldsymbol{\delta}} \boldsymbol{\epsilon}(\boldsymbol{\delta}, \theta)^{-1}. \tag{59}$$

This is equivalent to solving the following linear system of equations for $\boldsymbol{\omega}$,

$$\begin{aligned} \boldsymbol{\omega} \nabla_{\boldsymbol{\delta}} \boldsymbol{\epsilon}(\boldsymbol{\delta},\, \theta) &= -(\nabla_{\mathbf{u}^*} \ell)^{\mathrm{T}}\ \nabla_{\boldsymbol{\delta}} \mathbf{u}^* \\ &= -(\nabla_{\mathbf{u}^*} \ell)^{\mathrm{T}}\ \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \\ &= -\begin{bmatrix} (\nabla_{\mathbf{u}^*} \ell)^{\mathrm{T}} & \mathbf{0} \end{bmatrix}. \end{aligned} \tag{60}$$

Now, given that,

$$\nabla_{\boldsymbol{\delta}} \boldsymbol{\epsilon}(\boldsymbol{\delta},\, \theta) = \begin{bmatrix} \mathbf{R} & \mathbf{C}^{\mathrm{T}} \\ \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\mathbf{C} & \operatorname{diag}(\mathbf{C}\mathbf{u}^* - \mathbf{d}) \end{bmatrix} \tag{61}$$

taking the transpose of both sides in (60), we obtain,

$$\begin{bmatrix} \mathbf{R} & \mathbf{C}^{\mathrm{T}} \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}}) \\ \mathbf{C} & \operatorname{diag}(\mathbf{C}\mathbf{u}^* - \mathbf{d}) \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{\text{dup}} \end{bmatrix} = \begin{bmatrix} -\nabla_{\mathbf{u}^*} \ell \\ \mathbf{0} \end{bmatrix}. \tag{62}$$

Having solved for $\boldsymbol{\omega}$ with (62), we first compute the gradients of ℓ w.r.t the vectors $\mathbf{q}$ and $\mathbf{d}$ by using (58),

$$\begin{aligned} \nabla_{\mathbf{q}} \ell^{\mathrm{T}} &= \boldsymbol{\omega}\, \nabla_{\mathbf{q}} \boldsymbol{\epsilon} = \begin{bmatrix} \mathrm{d}\mathbf{u}^{\mathrm{T}} & \mathrm{d}\boldsymbol{\lambda}^{\mathrm{T}}_{\text{dup}} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix} = \mathrm{d}\mathbf{u}^{\mathrm{T}} \\ \therefore \nabla_{\mathbf{q}} \ell &= \mathrm{d}\mathbf{u} \end{aligned} \tag{63}$$

$$\begin{aligned} \nabla_{\mathbf{d}} \ell^{\mathrm{T}} &= \boldsymbol{\omega}\, \nabla_{\mathbf{d}} \boldsymbol{\epsilon} = \begin{bmatrix} \mathrm{d}\mathbf{u}^{\mathrm{T}} & \mathrm{d}\boldsymbol{\lambda}^{\mathrm{T}}_{\text{dup}} \end{bmatrix} \begin{bmatrix} 0 \\ -\operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}}) \end{bmatrix} \\ \therefore \nabla_{\mathbf{d}} \ell &= -\operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\, \mathrm{d}\boldsymbol{\lambda}_{\text{dup}}. \end{aligned} \tag{64}$$

For the matrix parameters $\mathbf{R}, \mathbf{C}$, note that, for example, $\nabla_{\mathbf{R}} \boldsymbol{\epsilon}$ is a three-dimensional tensor. To avoid tensor operations, we will instead use the following relation to solve for the gradient,

$$\mathrm{d}f(X) = \operatorname{tr}\left[\nabla_{\mathbf{X}} f(\mathbf{X})^{\mathrm{T}}\, \mathrm{d}\mathbf{X} \right] \tag{65}$$

for any function $f : \mathbb{R}^{n \times n} \to \mathbb{R}$, where $\mathrm{d}f$ and $\mathrm{d}\mathbf{X}$ denote the differential of f and $\mathbf{X}$ respectively. For $\mathbf{R}$, we get,

$$\begin{aligned} \mathrm{d}\boldsymbol{\epsilon} &= \begin{bmatrix} \mathrm{d}\mathbf{R}\mathbf{u}^* \\ \mathbf{0} \end{bmatrix} \\ \implies \mathrm{d}\ell &= \begin{bmatrix} \mathrm{d}\mathbf{u}^{\mathrm{T}} & \mathrm{d}\boldsymbol{\lambda}^{\mathrm{T}}_{\text{dup}} \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{R}\mathbf{u}^* \\ 0 \end{bmatrix} \\ &= \mathrm{d}\mathbf{u}^{\mathrm{T}}\, \mathrm{d}\mathbf{R}\, \mathbf{u}^* \\ &= \frac{1}{2}\Big(\mathrm{d}\mathbf{u}^{\mathrm{T}}\, \mathrm{d}\mathbf{R}\, \mathbf{u}^* + \mathbf{u}^{*\mathrm{T}}\, \mathrm{d}\mathbf{R}\, \mathrm{d}\mathbf{u} \Big) \\ &= \frac{1}{2} \operatorname{tr}\Big((\mathbf{u}^*\, \mathrm{d}\mathbf{u}^{\mathrm{T}} + \mathrm{d}\mathbf{u}\, \mathbf{u}^{*\mathrm{T}})\, \mathrm{d}\mathbf{R} \Big) \\ &= \operatorname{tr}\Big(\frac{1}{2} (\mathrm{d}\mathbf{u}\, \mathbf{u}^{*\mathrm{T}} + \mathbf{u}^*\, \mathrm{d}\mathbf{u}^{\mathrm{T}})^{\mathrm{T}}\, \mathrm{d}\mathbf{R} \Big) \\ \implies \nabla_{\mathbf{R}} \ell &= \frac{1}{2} (\mathrm{d}\mathbf{u}\, \mathbf{u}^{*\mathrm{T}} + \mathbf{u}^*\, \mathrm{d}\mathbf{u}^{\mathrm{T}}). \end{aligned} \tag{66}$$

Similarly for $\mathbf{C}$, we have,

$$\begin{aligned} \mathrm{d}\boldsymbol{\epsilon} &= \begin{bmatrix} \mathrm{d}\mathbf{C}^{\mathrm{T}} \boldsymbol{\lambda}^*_{\text{dup}} \\ \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\, \mathrm{d}\mathbf{C}\mathbf{u}^* \end{bmatrix} \\ \implies \mathrm{d}\ell &= \begin{bmatrix} \mathrm{d}\mathbf{u}^{\mathrm{T}} & \mathrm{d}\boldsymbol{\lambda}^{\mathrm{T}}_{\text{dup}} \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{C}^{\mathrm{T}} \boldsymbol{\lambda}^*_{\text{dup}} \\ \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\, \mathrm{d}\mathbf{C}\mathbf{u}^* \end{bmatrix} \\ &= \mathrm{d}\mathbf{u}^{\mathrm{T}}\, \mathrm{d}\mathbf{C}^{\mathrm{T}} \boldsymbol{\lambda}^*_{\text{dup}} + \mathrm{d}\boldsymbol{\lambda}^{\mathrm{T}}_{\text{dup}} \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\, \mathrm{d}\mathbf{C}\mathbf{u}^* \\ &= \operatorname{tr}\Big[\Big(\boldsymbol{\lambda}^*_{\text{dup}}\, \mathrm{d}\mathbf{u}^{\mathrm{T}} + \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\, \mathrm{d}\boldsymbol{\lambda}_{\text{dup}} \mathbf{u}^{*\mathrm{T}} \Big)^{\mathrm{T}}\, \mathrm{d}\mathbf{C} \Big] \\ \implies \nabla_{\mathbf{C}} \ell &= \boldsymbol{\lambda}^*_{\text{dup}}\, \mathrm{d}\mathbf{u}^{\mathrm{T}} + \operatorname{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\, \mathrm{d}\boldsymbol{\lambda}_{\text{dup}} \mathbf{u}^{*\mathrm{T}}. \end{aligned} \tag{67}$$

■

Lemma 3. *Given that Assumption 3 holds, then it follows that,*

$$\lambda_{\textit{non-dup},l}\, \mathrm{d}\lambda_{\textit{non-dup},l} = \sum_{j=1}^{\xi_l} \lambda_{\textit{dup},\zeta_{l,j}}\, \mathrm{d}\lambda_{\textit{dup},\zeta_{l,j}},\ \forall l \in [\![1, \bar{N}_{\textit{ineq}}]\!] \tag{68}$$

where $\mathrm{d}\boldsymbol{\lambda}_{\textit{dup}}, \mathrm{d}\boldsymbol{\lambda}_{\textit{non-dup}}$ denote the vectors of the variables of the KKT systems for the duplicate and non-duplicate problems respectively.

Proof: We drop the $*$ from $\mathbf{u}^*$ and $\boldsymbol{\lambda}^*_{\text{dup}}$ for ease of notation. To show that that (68) holds, consider the first row-block of the KKT system for the duplicate problem

$$\mathbf{R}\mathrm{d}\mathbf{u} + \mathbf{C}^{\mathrm{T}} \operatorname{diag}(\boldsymbol{\lambda}_{\text{dup}}) \mathrm{d}\boldsymbol{\lambda}_{\text{dup}} = -\nabla_{\mathbf{u}} \ell. \tag{69}$$

Representing the matrix multiplication with $\mathbf{C}^{\mathrm{T}}$ as a sum over the rows of $\mathbf{C}$, we can factor out $\mathbf{c}_{\zeta_l}$ over each equivalence

class, yielding,

$$\mathbf{C}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{dup}} = \sum_{k=1}^{Nr} \lambda_{\text{dup},k}\ \mathrm{d}\lambda_{\text{dup},k}\mathbf{c}_k^{\mathrm{T}} \tag{70}$$

$$= \sum_{l=1}^{\bar{N}_{\text{ineq}}}\sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\ \mathrm{d}\lambda_{\text{dup},\zeta_{l,j}}\mathbf{c}_{\zeta_{l,j}}^{\mathrm{T}} \tag{71}$$

$$= \sum_{l=1}^{\bar{N}_{\text{ineq}}}\left(\sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\ \mathrm{d}\lambda_{\text{dup},\zeta_{l,j}}\right)\mathbf{c}_{\zeta_l}^{\mathrm{T}} \tag{72}$$

$$= \sum_{l=1}^{\bar{N}_{\text{ineq}}} \chi_l \mathbf{c}_{\zeta_l}^{\mathrm{T}} \tag{73}$$

$$= \bar{\mathbf{C}}^{\mathrm{T}}\boldsymbol{\chi} \tag{74}$$

where we have defined the vector $\boldsymbol{\chi} \in \mathbb{R}^{\bar{N}_{\text{ineq}}}$ such that

$$\chi_l \coloneqq \sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\ \mathrm{d}\lambda_{\text{dup},\zeta_{l,j}}. \tag{75}$$

The KKT system for the duplicate problem reads

$$\begin{bmatrix}\mathbf{R} & \mathbf{C}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{dup}}) \\ \mathbf{C} & \mathrm{diag}(\mathbf{C}\mathbf{u}-\mathbf{d})\end{bmatrix}\begin{bmatrix}\mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{\text{dup}}\end{bmatrix} = \begin{bmatrix}-\nabla_{\mathbf{u}}\ell \\ \mathbf{0}\end{bmatrix}. \tag{76}$$

In order to eliminate the KKT variables related to the inactive constraints (i.e., $\mathrm{d}\boldsymbol{\lambda}_{\text{dup,inactive}}$), we need to first rewrite the above KKT matrix. For this, let $\mathbf{C}_{\text{active}}$ and $\mathbf{C}_{\text{inactive}}$ denote the blocks of the matrix $\mathbf{C}$ representing rows of active and inactive constraints. Then w.l.o.g., we can assume that the decision variables $\mathrm{d}\mathbf{u}$ are sorted in such a way that the KKT matrix above can be rewritten as follows,

$$\begin{bmatrix}\mathbf{R} & \mathbf{C}_{\text{active}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{active}}) & \mathbf{C}_{\text{inactive}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{inactive}}) \\ \mathbf{C}_{\text{active}} & \mathrm{diag}(\mathbf{C}_{\text{active}}\mathbf{u}-\mathbf{d}_{\text{active}}) & 0 \\ \mathbf{C}_{\text{inactive}} & 0 & \mathrm{diag}(\mathbf{C}_{\text{inactive}}\mathbf{u}-\mathbf{d}_{\text{inactive}})\end{bmatrix} \tag{77}$$

The Schur complement of the bottom right block given by,

$$\begin{bmatrix}\mathbf{R} & \mathbf{C}_{\text{active}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{active}}) \\ \mathbf{C}_{\text{active}} & 0\end{bmatrix} \tag{78}$$

is easily derived by applying complementary slackness to $\boldsymbol{\lambda}_{\text{inactive}}$ to observe that $\mathbf{C}_{\text{inactive}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{inactive}}) = 0$ and $\mathrm{diag}(\mathbf{C}_{\text{active}}\mathbf{u}-\mathbf{d}_{\text{active}}) = 0$.

Next, we can eliminate $\mathrm{d}\boldsymbol{\lambda}_{\text{dup,inactive}}$ and solve for the remaining KKT variables using the Schur complement as follows,

$$\begin{bmatrix}\mathbf{R} & \mathbf{C}_{\text{active}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{dup,active}}) \\ \mathbf{C}_{\text{active}} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{\text{dup,active}}\end{bmatrix} = \begin{bmatrix}-\nabla_{\mathbf{u}}\ell \\ \mathbf{0}\end{bmatrix}. \tag{79}$$

Since the (2,2) block of the Schur complement is zero, noting (74), we perform a change of variable with $\boldsymbol{\chi}$ to give

$$\begin{bmatrix}\mathbf{R} & \bar{\mathbf{C}}_{\text{active}}^{\mathrm{T}} \\ \bar{\mathbf{C}}_{\text{active}} & 0\end{bmatrix}\begin{bmatrix}\mathrm{d}\mathbf{u} \\ \chi_{\text{active}}\end{bmatrix} = \begin{bmatrix}-\nabla_{\mathbf{u}}\ell \\ \mathbf{0}\end{bmatrix}. \tag{80}$$

We now connect this with the KKT system of the non-duplicate problem, which reads

$$\begin{bmatrix}\mathbf{R} & \bar{\mathbf{C}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}) \\ \bar{\mathbf{C}} & \mathrm{diag}(\bar{\mathbf{C}}\mathbf{u}-\mathbf{d})\end{bmatrix}\begin{bmatrix}\mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{\text{non-dup}}\end{bmatrix} = \begin{bmatrix}-\nabla_{\mathbf{u}}\ell \\ \mathbf{0}\end{bmatrix}. \tag{81}$$

Similar to the duplicate problem, we can use the Schur complement to eliminate the KKT variables corresponding to the inactive constraints for the non-duplicate problem giving,

$$\begin{bmatrix}\mathbf{R} & \mathbf{C}_{\text{active}}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{non-dup,active}}) \\ \mathbf{C}_{\text{active}} & 0\end{bmatrix}\begin{bmatrix}\mathrm{d}\mathbf{u} \\ \mathrm{d}\boldsymbol{\lambda}_{\text{non-dup,active}}\end{bmatrix} = \begin{bmatrix}-\nabla_{\mathbf{u}}\ell \\ \mathbf{0}\end{bmatrix}. \tag{82}$$

Define $\boldsymbol{\gamma} \coloneqq \mathrm{diag}(\boldsymbol{\lambda}_{\text{non-dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{non-dup}}$. Performing a change of variables to consider $\boldsymbol{\gamma}_{\text{active}}$ instead yields

$$\begin{bmatrix}\mathbf{R} & \mathbf{C}_{\text{active}}^{\mathrm{T}} \\ \mathbf{C}_{\text{active}} & 0\end{bmatrix}\begin{bmatrix}\mathrm{d}\mathbf{u} \\ \boldsymbol{\gamma}_{\text{active}}\end{bmatrix} = \begin{bmatrix}-\nabla_{\mathbf{u}}\ell \\ \mathbf{0}\end{bmatrix}. \tag{83}$$

Consequently, we see that the KKT matrices in (80) and (83) are equivalent. Since $\mathbf{R}$ is positive definite and hence invertible, invoking LICQ gives us that the KKT matrix is also invertible. Hence, the unique solution to both (80) and (83) are equivalent, i.e., $\mathrm{d}\mathbf{u}$ and the active rows of $\boldsymbol{\chi}_{\text{active}}$ and $\boldsymbol{\gamma}_{\text{active}}$ must coincide.

For the inactive rows of $\boldsymbol{\chi}$ and $\boldsymbol{\gamma}$, note that both $\boldsymbol{\lambda}_{\text{dup,inactive}}$ and $\boldsymbol{\lambda}_{\text{non-dup,inactive}}$ are zero by complementary slackness. Consequently,

$$\boldsymbol{\gamma}_{\text{inactive}} = \boldsymbol{\chi}_{\text{inactive}} = 0. \tag{84}$$

Hence, $\boldsymbol{\gamma} = \boldsymbol{\chi}$. In other words,

$$\lambda_{\text{non-dup},l}\ \mathrm{d}\lambda_{\text{non-dup},l} = \sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\ \mathrm{d}\lambda_{\text{dup},\zeta_{l,j}}, \quad \forall l \in [\![1, \bar{N}_{\text{ineq}}]\!], \tag{85}$$

and thus (68) holds. ■

XII. Proof of Theorem 1

We are now ready to prove Theorem 1, which we restate below for the convenience of the reader.

Theorem 1. *Let* $\mathbf{M}$ *denote the matrix describing the relationship between the duplicate and non-duplicate constraints:*

$$\mathbf{C} = \mathbf{M}\bar{\mathbf{C}}, \quad \mathbf{d} = \mathbf{M}\bar{\mathbf{d}} \tag{86}$$

Then, for loss ℓ*, the gradients* $\nabla_{\mathbf{R}}\ell, \nabla_{\mathbf{q}}\ell, \nabla_{\mathbf{C}}\ell, \nabla_{\mathbf{d}}\ell$ *coincide for the duplicate and non-duplicate problems and are unique.*

Proof: Note that from Assumption 3 and Lemma 3 we have that $\mathbf{u}^*$ and $\mathrm{d}\mathbf{u}$ coincide for the duplicate and non-duplicate problems. Consequently, since $\nabla_{\mathbf{q}}\ell$ and $\nabla_{\mathbf{R}}\ell$ are only functions of $\mathbf{u}^*$ and $\mathrm{d}\mathbf{u}$, these must coincide.

Now, using the relationship between $\mathbf{d}$ and $\bar{\mathbf{d}}$ from (39), the definition of $\mathbf{M}$ in (86) implies that

$$\mathbf{M}[(l-1)r+1 : lr,\ l] = 1 \tag{87}$$

$$\mathbf{M}[(l-1)r+1 : lr,\ b] = 0 \tag{88}$$

$\forall l,\ b \in [\![1, \bar{N}_{\text{ineq}}]\!]$ and $j \neq l$.

Hence,

$$(\mathbf{M}^{\mathrm{T}}\boldsymbol{\lambda}_{\text{dup}})_l = \sum_{k=1}^{Nr} \mathbf{M}_{k,l}\lambda_{\text{dup},k},\ \forall l \in [\![1, \bar{N}_{\text{ineq}}]\!] \tag{89}$$

$$= \sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}} \tag{90}$$

$$= \boldsymbol{\nu}_l \tag{91}$$

and thus $\boldsymbol{\nu} = \mathbf{M}^{\mathrm{T}}\boldsymbol{\lambda}_{\text{dup}}$. We can similarly show that $\boldsymbol{\chi} = \mathbf{M}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{dup}}$:

$$\left(\mathbf{M}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}_{\text{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{dup}}\right)_l$$

$$= \sum_{k=1}^{Nr} \mathbf{M}_{k,l}\lambda_{\text{dup},k}\ \mathrm{d}\lambda_{\text{dup},k},\ \forall l \in [\![1, \bar{N}_{\text{ineq}}]\!] \tag{92}$$

$$= \sum_{j=1}^{\xi_l} \lambda_{\text{dup},\zeta_{l,j}}\ \mathrm{d}\lambda_{\text{dup},\zeta_{l,j}} \tag{93}$$

$$= \boldsymbol{\chi}_l. \tag{94}$$

Hence, for $\nabla_{\bar{\mathbf{d}}}\ell$, applying the chain rule yields,

$$\nabla_{\bar{\mathbf{d}}}\ell = (\nabla_{\bar{\mathbf{d}}}\mathbf{d})(\nabla_{\mathbf{d}}\ell) \tag{95}$$

$$= \mathbf{M}^{\mathrm{T}}\nabla_{\mathbf{d}}\ell \tag{96}$$

$$= -\mathbf{M}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{dup}} \tag{97}$$

$$= -\boldsymbol{\chi} \tag{98}$$

$$= -\boldsymbol{\gamma} \tag{99}$$

which coincides with (52b). Similarly for $\nabla_{\bar{\mathbf{C}}}\ell$, Noting that $\bar{\mathbf{c}}_i^{\mathrm{T}}$ represents the i^{th}column of $\bar{\mathbf{C}}$ and that $\bar{\mathbf{c}}_i^{\mathrm{T}} = \mathbf{M}\mathbf{c}_i^{\mathrm{T}}$ we have,

$$\nabla_{\bar{\mathbf{c}}_i^{\mathrm{T}}}\ell = (\nabla_{\bar{\mathbf{c}}_i^{\mathrm{T}}}\mathbf{c}_i^{\mathrm{T}})(\nabla_{\mathbf{c}_i^{\mathrm{T}}}\ell) \tag{100}$$

$$= \mathbf{M}^{\mathrm{T}}\nabla_{\mathbf{c}_i^{\mathrm{T}}}\ell \tag{101}$$

Now collecting these vectors into matrices we have,

$$\nabla_{\bar{\mathbf{C}}}\ell = (\nabla_{\bar{\mathbf{C}}}\mathbf{C})(\nabla_{\mathbf{C}}\ell)$$

$$= \mathbf{M}^{\mathrm{T}}\left(\boldsymbol{\lambda}^*_{\text{dup}}\,\mathrm{d}\mathbf{u}^{\mathrm{T}} + \mathrm{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{dup}}\mathbf{u}^{*\mathrm{T}}\right) \tag{102}$$

$$= \underbrace{\mathbf{M}^{\mathrm{T}}\boldsymbol{\lambda}^*_{\text{dup}}}_{\boldsymbol{\nu}}\,\mathrm{d}\mathbf{u}^{\mathrm{T}} + \underbrace{\mathbf{M}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\lambda}^*_{\text{dup}})\,\mathrm{d}\boldsymbol{\lambda}_{\text{dup}}}_{\boldsymbol{\chi}}\mathbf{u}^{*\mathrm{T}} \tag{103}$$

$$= \boldsymbol{\nu}\,\mathrm{d}\mathbf{u}^{\mathrm{T}} + \boldsymbol{\chi}\mathbf{u}^{*\mathrm{T}} \tag{104}$$

$$= \boldsymbol{\lambda}^*_{\text{non-dup}}\,\mathrm{d}\mathbf{u}^{\mathrm{T}} + \boldsymbol{\gamma}\mathbf{u}^{*\mathrm{T}} \tag{105}$$

which coincides with (52d). ∎

XIII. Connection between the Merged-OSQP Problem and the RNDCP

Assuming that convergence (i.e., consensus) is achieved, we can ignore the consensus related constraints and restate the KKT conditions (and the KKT system) for the Merged-OSQP problem so as to establish a connection to the KKT conditions (and the KKT system) of the non-duplicate problem.

The per-agent (or per neighborhood) objective function for the Merged-OSQP problem is given by,

$$\mathcal{H}_i(\tilde{\mathbf{u}}_i) = \frac{1}{2}\tilde{\mathbf{u}}_i^{\mathrm{T}}\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i^{\mathrm{T}}\tilde{\mathbf{u}}_i$$

$$\text{where, } \tilde{\mathbf{R}}_i = \mathrm{bdiag}(\mathbf{R}_i, \{\mathbf{0}_{\mathbf{m}\times\mathbf{m}}\}_{j=1}^{r}) \text{ and,}$$

$$\tilde{\mathbf{q}}_i = \left[\mathbf{G}_i^{\mathrm{T}}\frac{\partial V}{\partial \mathbf{x}_i}; \{\mathbf{0}_{\mathbf{m}\times\mathbf{1}}\}_{j=1}^{r}\right].$$

The definitions of $\tilde{\mathbf{R}}_i$ and $\tilde{\mathbf{q}}_i$ above imply that the objective functions of each local QP consider only the objectives of the respective ego agents. This allows us to restate the objective of the Merged-OSQP problem and equate it to that of the non-duplicate centralized problem. Thus, we have,

$$\sum_{i=1}^{N}\mathcal{H}_i(\tilde{\mathbf{u}}_i) = \mathcal{H}(\mathbf{u}) \tag{106}$$

where $\mathcal{H}(\mathbf{u})$ corresponds to the objective of the non-duplicate (or the full centralized) problem.

Next, assuming we have converged, a complementary slackness condition for the Merged-OSQP problem can be constructed. To do this we refer to the following equations from the OSQP paper [42, Equation 9], but based on notation from our main paper,

$$\boldsymbol{\lambda}_{i_-}^{*\,\mathrm{T}}(\hat{\mathbf{z}}_i^* - \hat{\mathbf{z}}_i^{lb}) = 0 \tag{107}$$

$$\boldsymbol{\lambda}_{i_-}^* = \min(\boldsymbol{\lambda}_i^*, 0) \tag{108}$$

where, $\hat{\mathbf{z}}_i^{lb}$ is the lower bound on $\hat{\mathbf{z}}_i^*$. Now, based on the constraints established in the main paper (i.e. $\mathbf{A}_i\tilde{\mathbf{u}}_i \leq \mathbf{d}_i$), we have,

$$-\infty < \hat{\mathbf{z}}_i^* \leq \mathbf{d}_i, \quad \forall\, i \in [[1, N]]$$

Since the $\hat{\mathbf{z}}_i^{lb} \to -\infty$, $(\hat{\mathbf{z}}_i^* - \hat{\mathbf{z}}_i^{lb})$ will tend to a vector of large positive numbers. Additionally, because $\boldsymbol{\lambda}_{i_-}^*$ can only be non-positive (based on (108)), $\boldsymbol{\lambda}_{i_-}^*$ must be equal to $\mathbf{0}$ so that the inner-product condition (107) is satisfied (because a linear combination of large positive numbers with negative coefficients can never be equal to zero). This establishes that the following lower bound,

$$\boldsymbol{\lambda}^* \geq \mathbf{0}. \tag{109}$$

On convergence we have, $\mathbf{A}_i\tilde{\mathbf{u}}_i^* = \tilde{\mathbf{z}}_i^* = \hat{\mathbf{z}}_i^*$. The condition on the upper bound of $\boldsymbol{\lambda}$ mentioned in the OSQP paper then becomes,

$$\boldsymbol{\lambda}_{i_+}^{*\,\mathrm{T}}(\mathbf{A}_i\tilde{\mathbf{u}}_i^* - \mathbf{d}_i) = 0. \tag{110}$$

Now, there are two possibilities,

1) **Tight constraints**: Where we have, $\mathbf{A}_i\tilde{\mathbf{u}}_i^* = \mathbf{d}_i$. In this case we have $\boldsymbol{\lambda}_{i_+}^* \geq 0$ (based on (109).
2) **Loose constraints**: Where we have, $\mathbf{A}_i\tilde{\mathbf{u}}_i^* < \mathbf{d}_i$. Thus, for (110) to hold, $\boldsymbol{\lambda}_{i_+}^* = \mathbf{0}$ (because the coefficients of a linear combination of negative numbers have to be zero so that the sum is equal to zero).

Thus, when constraints are loose, the upper bound on $\boldsymbol{\lambda}$ will be 0. This therefore implies,

$$\mathrm{diag}(\boldsymbol{\lambda}_i^*)(\mathbf{A}_i\tilde{\mathbf{u}}_i^* - \mathbf{d}_i) = \mathbf{0}. \tag{111}$$

The equations (109) and (111) are the complementary slackness conditions that the solution of the converged Merged-OSQP must satisfy.

Next, we can combine the constraints of all neighborhoods ($\mathbf{A}_i$) into a single matrix. This would precisely be the matrix $\mathbf{C}$ constructed from the matrices $\mathbf{A}_i$ in the duplicate problem defined in Section X. Thus, the complementary slackness condition satisfied by the converged solution of Merged-OSQP is given by,

$$\begin{aligned} &\mathrm{diag}(\boldsymbol{\lambda}^*_{\text{mosqp}})(\mathbf{C}\mathbf{u}^*_{\text{mosqp}} - \mathbf{d}) = \mathbf{0} \\ &\boldsymbol{\lambda}^*_{\text{mosqp}} \geq \mathbf{0} \end{aligned} \tag{112}$$

where, $\boldsymbol{\lambda}^*_{\text{mosqp}} = [\boldsymbol{\lambda}^*_1; \boldsymbol{\lambda}^*_2; \ldots; \boldsymbol{\lambda}^*_{Nr}]$ and $\mathbf{u}^*_{\text{mosqp}} = \left[\tilde{\mathbf{u}}^*_{1,1}; \tilde{\mathbf{u}}^*_{2,1}; \ldots; \tilde{\mathbf{u}}^*_{N,1}\right]$ where $\tilde{\mathbf{u}}^*_{i,1}$ is the first block of the output of the i^{th} local QP ($\tilde{\mathbf{u}}^*_i$). However, because the solution of Merged-OSQP must also satisfy $\tilde{\mathbf{u}}^*_i = \tilde{\mathbf{g}}_i$ (i.e. we have consensus), we have $\mathbf{u}^*_{\text{mosqp}} = \mathbf{u}^*_{\text{non-dup}}$.

The Lagrangian that the converged solution satisfies can be written as,

$$\begin{aligned} \mathcal{L}_{\text{mosqp}} &= \sum_{i=1}^{N} \left(\mathcal{H}_i(\tilde{\mathbf{u}}^*_i) + \sum_{i=1}^{N} {\boldsymbol{\lambda}^*_i}^{\mathrm{T}} (\mathbf{A}_i \tilde{\mathbf{u}}^*_i - \mathbf{d}_i) \right) \\ &= \mathcal{H}(\mathbf{u}^*_{\text{non-dup}}) + {\boldsymbol{\lambda}^*_{\text{mosqp}}}^{\mathrm{T}} (\mathbf{C}\mathbf{u}^*_{\text{non-dup}} - \mathbf{d}) \end{aligned}$$

where, as mentioned earlier, we have ignored the consensus related variables.

The KKT conditions are given by,

$$\boldsymbol{\epsilon}_{\text{mosqp}} = \nabla_{\boldsymbol{\delta}_{\text{mosqp}}} \mathcal{L}_{\text{mosqp}} = \begin{bmatrix} \nabla_{\boldsymbol{\delta}_{\text{mosqp}}} \mathcal{H}(\tilde{\mathbf{u}}^*) + \mathbf{C}^{\mathrm{T}} \boldsymbol{\lambda}^*_{\text{mosqp}} \\ \mathrm{diag}(\boldsymbol{\lambda}^*_{\text{mosqp}})(\mathbf{C}\mathbf{u}^*_{\text{mosqp}} - \mathbf{d}) \end{bmatrix} = \mathbf{0}.$$

Thus, we have,

$$\begin{aligned} \boldsymbol{\epsilon}_{\text{mosqp}} &= \begin{bmatrix} \mathbf{R}\mathbf{u}^*_{\text{mosqp}} + \mathbf{C}^{\mathrm{T}} \boldsymbol{\lambda}^*_{\text{mosqp}} \\ \mathrm{diag}(\boldsymbol{\lambda}^*_{\text{mosqp}})(\mathbf{C}\mathbf{u}^*_{\text{mosqp}} - \mathbf{d}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}\mathbf{u}^*_{\text{non-dup}} + \mathbf{C}^{\mathrm{T}} \boldsymbol{\lambda}^*_{\text{mosqp}} \\ \mathrm{diag}(\boldsymbol{\lambda}^*_{\text{mosqp}})(\mathbf{C}\mathbf{u}^*_{\text{non-dup}} - \mathbf{d}) \end{bmatrix} = \mathbf{0} \end{aligned}$$

where $\boldsymbol{\delta}_{\text{mosqp}} = [\mathbf{u}^*_{\text{mosqp}}; \boldsymbol{\lambda}^*_{\text{mosqp}}]$. The Jacobian of the KKT conditions is given by,

$$\nabla_{\boldsymbol{\delta}_{\text{mosqp}}} \boldsymbol{\epsilon}_{\text{mosqp}} = \begin{bmatrix} \mathbf{R} & \mathbf{C}^{\mathrm{T}} \\ \mathrm{diag}(\boldsymbol{\lambda}^*_{\text{mosqp}})\mathbf{C} & \mathrm{diag}(\mathbf{C}\mathbf{u}^*_{\text{non-dup}} - \mathbf{d}) \end{bmatrix} \tag{113}$$

which yields the following KKT system with the Jacobian matrix transposed,

$$\begin{bmatrix} \mathbf{R} & \mathbf{C}^{\mathrm{T}} \mathrm{diag}(\boldsymbol{\lambda}^*_{\text{mosqp}}) \\ \mathbf{C} & \mathrm{diag}(\mathbf{C}\mathbf{u}^*_{\text{non-dup}} - \mathbf{d}) \end{bmatrix} \begin{bmatrix} \mathrm{d}\mathbf{u}_{\text{mosqp}} \\ \mathrm{d}\boldsymbol{\lambda}_{\text{mosqp}} \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{u}^*_{\text{non-dup}}} \ell \\ \mathbf{0} \end{bmatrix}. \tag{114}$$

The connection between the KKT system above that the Merged-OSQP solution satisfies and the KKT system that the non-duplicate problem satisfies can be similarly established as was done for the duplicate centralized problem by combining KKT variables corresponding to duplicate constraints. This justifies the use of Lagrange multipliers obtained as a solution of Merged-OSQP with a duplicate centralized system for computation of backward pass gradients. So long as the LICQ is satisfied by the non-duplicate problem, the sum of the Lagrange multipliers will be unique.

XIV. Derivation of Non-Linear Feynman-Kac

For this derivation we will consider a simple stochastic optimal control problem for a control-affine system without any safety constraints. The corresponding HJB-PDE is given by,

$$\begin{aligned} V_t + \inf_{\mathbf{u}} \Big\{ &\frac{1}{2} \mathrm{tr}(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}) + V_{\mathbf{x}}^{\mathrm{T}} (\mathbf{f} + \mathbf{G}\mathbf{u}) \\ &+ \sum_{i=1}^{N} c_i + \frac{1}{2} \mathbf{u}^{\mathrm{T}} \mathbf{R} \mathbf{u} \Big\} = 0 \\ V(\mathbf{x}(T), T) &= \phi(\mathbf{x}(T)) \end{aligned}$$

where the subscripts $\mathbf{x}$ and t imply gradients with respect to $\mathbf{x}$ and derivative with respect to time respectively.

By setting the derivative of $\mathcal{H}$ with respect to $\mathbf{u}$ equal to zero, we can derive the following closed-form expression for the optimal control,

$$\mathbf{u}^* = -\mathbf{R}^{-1} \mathbf{G}^{\mathrm{T}} V_{\mathbf{x}}$$

Next we substitute for the optimal control into the HJB-PDE which allows us to drop the infimum operator giving,

$$\begin{aligned} V_t + \frac{1}{2} \mathrm{tr}(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}) + V_{\mathbf{x}}^{\mathrm{T}} \mathbf{f} \\ + \sum_{i=1}^{N} c_i - \frac{1}{2} V_{\mathbf{x}}^{\mathrm{T}} \mathbf{G} \mathbf{R}^{-1} \mathbf{G}^{\mathrm{T}} V_{\mathbf{x}} = 0 \end{aligned}$$

Rearranging the above equation, we get the following expression for the derivative of V w.r.t time t,

$$\begin{aligned} V_t = -\frac{1}{2} \mathrm{tr}(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}) - V_{\mathbf{x}}^{\mathrm{T}} \mathbf{f} \\ - \sum_{i=1}^{N} c_i + \frac{1}{2} V_{\mathbf{x}}^{\mathrm{T}} \mathbf{G} \mathbf{R}^{-1} \mathbf{G}^{\mathrm{T}} V_{\mathbf{x}} \end{aligned}$$

Now, we apply Ito's formula to the value-function to obtain,

$$\begin{aligned} \mathrm{d}V &= V_t \mathrm{d}t + V_{\mathbf{x}}^{\mathrm{T}} \mathrm{d}\mathbf{x} + \frac{1}{2} \mathrm{tr}\left(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}\right) \mathrm{d}t \\ \mathrm{d}V &= \Big(-\frac{1}{2} \mathrm{tr}(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}) - V_{\mathbf{x}}^{\mathrm{T}} \mathbf{f} \\ &\quad - \sum_{i=1}^{N} c_i + \frac{1}{2} V_{\mathbf{x}}^{\mathrm{T}} \mathbf{G} \mathbf{R}^{-1} \mathbf{G}^{\mathrm{T}} V_{\mathbf{x}} \Big) \mathrm{d}t \\ &\quad + V_{\mathbf{x}}^{\mathrm{T}} \mathrm{d}\mathbf{x} + \frac{1}{2} \mathrm{tr}\left(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}\right) \mathrm{d}t \\ &= \Big(-\frac{1}{2} \mathrm{tr}(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}) - V_{\mathbf{x}}^{\mathrm{T}} \mathbf{f} \\ &\quad - \sum_{i=1}^{N} c_i + \frac{1}{2} V_{\mathbf{x}}^{\mathrm{T}} \mathbf{G} \mathbf{R}^{-1} \mathbf{G}^{\mathrm{T}} V_{\mathbf{x}} \Big) \mathrm{d}t \\ &\quad + V_{\mathbf{x}}^{\mathrm{T}} \Big(\underbrace{(\mathbf{f} + \mathbf{G}\mathbf{u}^*)\mathrm{d}t + \Sigma \mathrm{d}\mathbf{w}}_{\mathrm{d}\mathbf{x}} \Big) + \frac{1}{2} \mathrm{tr}\left(V_{\mathbf{xx}} \Sigma\Sigma^{\mathrm{T}}\right) \mathrm{d}t \end{aligned}$$

$$
\begin{aligned}
&= -\cancel{\frac{1}{2}\mathrm{tr}(V_{\mathbf{xx}}\Sigma\Sigma^{\mathrm{T}})\mathrm{d}t} - \cancel{V_{\mathbf{x}}^{\mathrm{T}}\mathbf{f}\,\mathrm{d}t} \\
&\quad - \Big(\sum_{i=1}^{N} c_i - \frac{1}{2}V_{\mathbf{x}}^{\mathrm{T}}\mathbf{G}\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\mathbf{x}}\Big)\mathrm{d}t + \cancel{V_{\mathbf{x}}^{\mathrm{T}}\mathbf{f}\,\mathrm{d}t} \\
&\quad + V_{\mathbf{x}}^{\mathrm{T}}(\mathbf{G}\mathbf{u}^{*}\mathrm{d}t + \Sigma\mathrm{d}\mathbf{w}) + \cancel{\frac{1}{2}\,\mathrm{tr}\,(V_{\mathbf{xx}}\Sigma\Sigma^{\mathrm{T}})\mathrm{d}t} \\
&= -\Big(\sum_{i=1}^{N} c_i - \frac{1}{2}V_{\mathbf{x}}^{\mathrm{T}}\mathbf{G}\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\mathbf{x}}\Big)\mathrm{d}t \\
&\quad + V_{\mathbf{x}}^{\mathrm{T}}(\mathbf{G}\mathbf{u}^{*}\mathrm{d}t + \Sigma\mathrm{d}\mathbf{w}), \quad \text{(cancelling out common terms)} \\
&= -\Big(\sum_{i=1}^{N} c_i - \frac{1}{2}V_{\mathbf{x}}^{\mathrm{T}}\mathbf{G}\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\mathbf{x}}\Big)\mathrm{d}t \\
&\quad + V_{\mathbf{x}}^{\mathrm{T}}\mathbf{G}\Big(\underbrace{-\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\mathbf{x}}}_{\mathbf{u}^{*}}\Big)\mathrm{d}t + V_{\mathbf{x}}^{\mathrm{T}}\Sigma\mathrm{d}\mathbf{w} \\
\mathrm{d}V &= -\Big(\underbrace{\sum_{i=1}^{N} c_i + \frac{1}{2}V_{\mathbf{x}}^{\mathrm{T}}\mathbf{G}\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\mathbf{x}}}_{\text{running cost}}\Big)\mathrm{d}t + V_{\mathbf{x}}^{\mathrm{T}}\Sigma\mathrm{d}\mathbf{w}
\end{aligned}
$$

This final expression gives us the required BSDE of the FBSDE system. The FSDE of the FBSDE system is the dynamics given by,

$$\mathrm{d}\mathbf{x}(t) = \Big(\mathbf{f}\big(\mathbf{x}(t)\big) + \mathbf{G}\big(\mathbf{x}(t)\big)\mathbf{u}(t)\Big)\,\mathrm{d}t + \Sigma\big(\mathbf{x}(t)\big)\,\mathrm{d}\mathbf{w}(t)$$

Putting everything togther, the Non-linear Feynman Kac theorem connects the unique solution of the HJB-PDE with the following system of FBSDEs,

$$
\begin{aligned}
&\mathrm{d}\mathbf{x} = \Big(\mathbf{f}\big(\mathbf{x}\big) + \mathbf{G}\big(\mathbf{x}\big)\mathbf{u}\Big)\,\mathrm{d}t + \Sigma\big(\mathbf{x}\big)\,\mathrm{d}\mathbf{w}, \quad \mathbf{x}(0) = \mathbf{x}_0 \\
&\mathrm{d}V = -\Big(\sum_{i=1}^{N} c_i + \frac{1}{2}V_{\mathbf{x}}^{\mathrm{T}}\mathbf{G}\mathbf{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\mathbf{x}}\Big)\mathrm{d}t + V_{\mathbf{x}}^{\mathrm{T}}\Sigma\mathrm{d}\mathbf{w}, \\
&V(\mathbf{x}(T), T) = \phi(\mathbf{x}(T)).
\end{aligned}
$$

XV. Merged-CADMM Method Additional Expressions

The norms of the primal and dual residuals used in the Merged-CADMM termination criteria are given by:

$$
\begin{aligned}
r_{pri,1} &= \max_{i\in[\![1,N]\!]} \|\mathbf{A}_i\tilde{\mathbf{u}}_i - \tilde{\mathbf{z}}_i\|_\infty \\
r_{pri,2} &= \max_{i\in[\![1,N]\!]} \|\tilde{\mathbf{u}}_i - \tilde{\mathbf{g}}_i\|_\infty, \\
r_{dual,1} &= \max_{i\in[\![1,N]\!]} \|\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i + \tilde{\mathbf{q}}_i + \mathbf{A}_i^{\mathrm{T}}\mathbf{y}_i\|_\infty, \\
r_{dual,2} &= \max_{i\in[\![1,N]\!]} \|\rho_2(\tilde{\mathbf{g}}_i^{l} - \tilde{\mathbf{g}}_i^{l-1})\|_\infty,
\end{aligned}
$$

The corresponding thresholds are the following:

$$\epsilon_{pri,a} = \epsilon_{\text{abs}} + \epsilon_{\text{rel}}\kappa_{pri,a},\ \epsilon_{dual,a} = \epsilon_{\text{abs}} + \epsilon_{\text{rel}}\kappa_{dual,a},\ a = 1, 2$$

where:

$$
\begin{aligned}
\kappa_{pri,1} &= \max_{i\in[\![1,N]\!]} \max\{\|\mathbf{A}_i\tilde{\mathbf{u}}_i\|_\infty, \|\tilde{\mathbf{z}}_i\|_\infty\}, \\
\kappa_{pri,2} &= \max_{i\in[\![1,N]\!]} \max\{\|\tilde{\mathbf{u}}_i\|_\infty, \|\tilde{\mathbf{g}}_i\|_\infty\}, \\
\kappa_{dual,1} &= \max_{i\in[\![1,N]\!]} \max\{\|\tilde{\mathbf{R}}_i\tilde{\mathbf{u}}_i\|_\infty, \|\tilde{\mathbf{q}}_i\|_\infty, \|\mathbf{A}_i^{\mathrm{T}}\mathbf{y}_i\|_\infty\}, \\
\kappa_{dual,2} &= \max_{i\in[\![1,N]\!]} \|\boldsymbol{\zeta}_i\|_\infty.
\end{aligned}
$$

XVI. Failure Probabilities for SCBF

Restating the result from [36, Proposition 1] here for convenience of the reader: *suppose there exists a twice differentiable function $B(\mathbf{x})$, that satisfies the following inequalities,*

$$B(\mathbf{x}) \geq 0 \quad \forall\,\mathbf{x} \in \mathbb{R}^{Nn_i} \tag{115}$$

$$B(\mathbf{x}) \geq 1 \quad \forall\,\mathbf{x} \in (\mathbb{R}^{Nn_i}\backslash\mathcal{S}) \tag{116}$$

$$\frac{\partial B}{\partial \mathbf{x}}^{\mathrm{T}}\Big(\mathbf{f} + \mathbf{G}\mathbf{u}\Big) + \frac{1}{2}tr\Big(\frac{\partial^2 B}{\partial \mathbf{x}^2}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\Big) \leq -\alpha B(\mathbf{x}) + \beta, \tag{117}$$

where (117) is satisfied $\forall\, t \in [0,\, T]$ and $\forall\,\mathbf{x} \in \mathbb{R}^{Nn_i}$ for some $\alpha \geq 0$ and $\beta \geq 0$. Define,

$$\rho_u = \mathbb{P}\big\{\mathbf{x}(t) \in (\mathbb{R}^{Nn_i}\backslash\mathcal{S}) \text{ for some } t \in [0,\, T]\big\},\ \text{and,}$$

$$\rho_B = \mathbb{P}\Big\{\sup_{t\in[0,\,T]} B(\mathbf{x}) \geq 1\Big\}$$

then, we have the following bounds,

- *if $\alpha > 0$ and $\beta \leq \alpha$,* $\quad \rho_u \leq \rho_B \leq 1 - (1 - B_0)e^{-\beta T}$
- *if $\alpha > 0$ and $\beta \geq \alpha$,* $\quad \rho_u \leq \rho_B \leq \frac{B_0 + (e^{\beta T} - 1)\frac{\beta}{\alpha}}{e^{\beta T}}$
- *if $\alpha = 0$,* $\quad \rho_u \leq \rho_B \leq B_0 + \beta T$

where $B_0 = B\big(\mathbf{x}(0)\big)$.