DARE: The Deep Adaptive Regulator for Closed-Loop Predictive Control

Anonymous Authors[1]

Abstract

A fundamental challenge in optimal control (OC) and machine learning is how to find the optimal policy of an agent that operates in changing and uncertain environments.Traditional OC methods face challenges in scalability and adaptability due to the curse-of-dimensionality and the reliance on fixed prior models of the environment. Model Predictive Control (MPC) addresses these issues but is limited to open-loop controls, i.e., policies without feedback to adapt. Another approach is Reinforcement Learning (RL) which can scale well to high-dimensional applications but is often computationally expensive and can be unreliable in highly stochastic continuous-time setups. This paper presents the **D**eep **A**daptive **Re**gulator (`DARE`) which combines deep learning with OC to compute closed-loop adaptive policies by solving continuously updated OC problems that explicitly trade off exploration with exploitation. We show that our method effectively transfers learning to unseen environments and is suited for online decision-making in environments that change in real time. We test `DARE` in various setups and demonstrate its superior performance over traditional methods, especially in scenarios with misspecified priors and nonstationary dynamics.

1. Introduction

This paper considers the decision-making problem of an agent who seeks to control a system optimally while learning both (i) the system dynamics and (ii) the reward function through interaction with the environment. Optimal control (OC) is a ubiquitous framework to solve such problems in both deterministic and stochastic settings. Classical OC methods, however, generally assume known and stationary environments. In real-world settings such as biology (Iglesias & Ingalls, 2010) and finance (Cartea et al., 2015), agents infer dynamics from noisy and non-stationary data, restricting the applicability of OC methods in practice.

Model predictive control (MPC) is one approach for controlling uncertain and non-stationary environments. In MPC, agents optimize control inputs by solving a sequence of *open-loop* optimization problems based on a predictive model over a receding time horizon, allowing for real-time adjustments of the agent's policy (see Garcia et al., 1989). One efficiently obtains open-loop policies through the Pontryagin Maximum Principle, but computational bottlenecks limit their effectiveness to systems that undergo only gradual changes (see Todorov & Li, 2005). Another common approach is Reinforcement Learning (RL), in which which agents compute optimal policies iteratively through trial and error (Sutton & Barto, 2018). However, most RL methods require substantial computational overhead and perform poorly in highly stochastic and continuous-time setups or when the environment is irregularly sampled (see Tallec et al., 2019; Yildiz et al., 2021). Hence, flexible methods that compute optimal closed-loop policies efficiently in uncertain, non-stationary environments are desirable.

In this paper, we propose the Deep Adaptive Regulator (`DARE`). `DARE` is a deep learning-based method for solving decision-making problems in continuous-time. Our method consists of two distinct phases: offline and online. In both phases, deep neural networks (DNNs) parameterize the agent's value function and policy and are optimized using Monte Carlo integration of a variational formulation of a Hamilton–Jacobi–Bellman (HJB) equation, similar to the Deep Galerkin Method (DGM) (Sirignano & Spiliopoulos, 2018). In contrast to DGM, we use a multi-objective loss function similar to that in (Al-Aradi et al., 2022) which is suitable to non-parametric models of the environment.

In the *offline* phase, the agent uses an initial estimate of the environment to construct and solve an approximate OC problem and to obtain an initial optimal policy. To accelerate training, we propose a novel inductive bias. In particular, we initialize the value function network around the terminal reward function, and we initialize the policy network around a locally optimal policy using the Iterative Linear-Quadratic Gaussian (ILQG) method (Todorov & Li, 2005).

In the *online* phase, the agent updates their estimates of

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

the environment and solves a stream of updated OC problems over a receding horizon to continuously adapt their policy, similar to MPC. In contrast to MPC, however, `DARE` computes *closed-loop* policies. To account for the agent's uncertainty about the environment, we consider a modification of the agent's objective function which explicitly balances exploration and exploitation. The efficient transfer of knowledge throughout the stream of updated OC problems is key to the success of `DARE` in the online phase. We use the tools of regular perturbation theory to provide a theoretical justification for the efficacy of Transfer Learning (TL) in `DARE` and to show that DNN approximations of HJB solutions are continuous with respect to their parameters.

To benchmark `DARE`, we study its performance in two adaptive OC problems: (i) a Linear-Quadratic-Gaussian (LQG) Regulator problem where the agent learns the drift of the system, (ii) a nonlinear MPC problem where the agent learns a reward function modeled by a Gaussian Process (GP), and (iii) a high-dimensional nonlinear MPC problem motivated by algorithmic trading in finance (Cartea et al., 2015). Our results show that our method significantly improves training speeds and approximation accuracy over existing DNN architectures in the offline phase. In the online phase, we demonstrate that `DARE` outperforms continuous-time Kalman filtering and A2C (Mnih et al., 2016) to solve the LQG problem. In the MPC problems, we show that `DARE` is robust to noise and abrupt changes of the environment, particularly when the agent encourages exploration through a modified objective function.

In summary, this paper:

- proposes the deep learning-based method `DARE` to compute *globally* optimal closed-loop controls efficiently in data-driven decision-making problems,
- proposes an OC problem formulation that explicitly trades off exploration and exploitation to adapt to non-stationary system dynamics and reward functions,
- proposes a novel inductive bias based on ILQG and the structure of the HJB which significantly improves convergence speed and performance of DGM,
- demonstrates experimentally that `DARE` transfers knowledge efficiently to unseen environments,
- justifies theoretically the TL ability of `DARE` using perturbation theory in OC.

2. Related Work

Deep Learning Methods for Control. Deep learning is extensively used to solve HJB equations because of its flexibility and scalability to high dimensions (Huré et al., 2021; Bachouch et al., 2022; Onken et al., 2022; Kunisch & Walter, 2021). Among earlier examples, (Han et al., 2018; Sirignano & Spiliopoulos, 2018) provide two contrasting approaches. The former proposes the Deep Galerkin Method, which uses Monte Carlo integration to minimize a variational form of the HJB in a mesh-free manner, while the latter proposes the Deep BSDE method, which reformulates the PDE as a backward stochastic differential equation and approximates the gradient of the solution with a neural network. Global convergence of DGM was recently established in (Jiang et al., 2023). There are numerous extensions to their method, including adaptive Monte Carlo sampling in (Aristotelous et al., 2023), augmented loss function for non-parametric running penalties and drifts in (Al-Aradi et al., 2022), and optimally weighted loss objectives in (van der Meer et al., 2022).

Model Predictive Control. Classical methods in MPC are the foundation of many online control optimization methods in both the deterministic and stochastic settings (Allgower et al., 2004; Mesbah, 2016). Recently, deep learning was integrated with MPC, with applications in controlling uncertain nonlinear systems such as unsteady fluid flow and high-performance autonomous systems (Lenz et al., 2015; Mishra et al., 2023; Bieker et al., 2020; Salzmann et al., 2023; Nagabandi et al., 2018). These approaches leverage neural networks to enhance dynamic modeling capacity and real-world control performance.

Continuous-Time Reinforcement Learning. Methods in deep RL are highly effective in several complex decision-making problems (Mnih et al., 2013; Lillicrap et al., 2015; Schulman et al., 2017). Continuous-time environments pose significant challenges to RL methods (Tallec et al., 2019; Yildiz et al., 2021). In particular many RL methods optimize incorrect objectives (see Jia & Zhou, 2022) when environments are noisy, e.g., temporal difference (TD) learning (Doya, 2000). Recently, (Wang et al., 2020) study the exploration-exploitation trade-off in stochastic and continuous-time RL, and prove that the optimal exploration policy is Gaussian in a Linear-Quadratic setting. Subsequent work (Jia & Zhou, 2022; 2023; Hoglund et al., 2023; Basei et al., 2022) extend RL methods to stochastic and continuous-time environments.

3. Problem Formulation

Let $X_t \in \mathbb{R}^{d_X}$ be a stochastic system evolving continuously in time. We consider an agent who controls X with a policy $u_t \in \mathbb{R}^{d_u}$ over a fixed time horizon $T > 0$ to maximize a terminal reward $g : \mathbb{R}^{d_X} \to \mathbb{R}$. The agent's actions u_t on the system X_t incur a penalty modelled by a function $f : \mathbb{R}^{d_X} \times \mathbb{R}^{d_u} \to \mathbb{R}$, and their impact on the system dynamics is modelled by a drift function $h : \mathbb{R}^{d_X} \times \mathbb{R}^{d_u}$. The system evolves according to the dynamics

$$dX_t = h(X_t, u_t)\,\mathrm{d}t + \tilde{\Sigma}\,\mathrm{d}W_t\,, \quad X_0 \in \mathbb{R}^{d_X}\,, \tag{1}$$

where W is a d_X-dimensional Brownian motion and $\tilde{\Sigma} \in \mathbb{R}^{d_X \times d_X}$ is a covariance matrix. We assume $\tilde{\Sigma}, T$ and g are fixed and known to the agent, and $\mathfrak{p} := (h, f)$ represents the modelling assumptions of the agent over the environment. We refer to $\mathfrak{p}$ as the *OC pair*.

Classical OC approaches assume a fixed and known pair $\mathfrak{p}$ to compute an optimal policy. In practice, the agent uses an uncertain estimate $\widehat{\mathfrak{p}}$ of the true environment. To account for this uncertainty, DARE solves the decision-making problem in two phases: offline and online. In the offline phase, the agent solves an OC problem according to an initial estimate of the environment $\widehat{\mathfrak{p}}_0$. In the online phase, the agent receives noisy samples of the true OC pair and updates their estimate of the environment and their control policy accordingly. The agent is uncertain of their estimate, so it may be profitable to explore unknown domains of the system for potentially higher rewards. Hence, a balance must be struck between exploring new information and exploiting existing knowledge.

Offline phase. At time $t = 0$, the agent assumes an initial estimate $\widehat{\mathfrak{p}}_0 = (\widehat{h}_0, \widehat{f}_0)$ of the OC pair. To explicitly account for the exploration-exploitation trade-off, the agent seeks an optimal policy u^* which maximizes the following performance criterion:

$$J(s,x;u) = \mathbb{E}\Big[g(X_T) - \int_s^T \mathbb{E}\left[\widehat{f}_0\right](X_r, u_r)\,\mathrm{d}r \qquad (2)$$
$$+ \phi \int_s^T \mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right](X_r, u(r, X_r))\,\mathrm{d}r \;\Big|\; \mathcal{G}_s\Big],$$

for all $s \in [0, T]$, where $\mathcal{G}_s$ is the information known to the agent at time s, $\mathbb{E}\left[\widehat{f}_0\right]$ denotes the mean prediction of the estimate $\widehat{f}_0$ and $\mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right]$ is the sum of the variance of each each estimator in $\widehat{\mathfrak{p}}_0$, and we assume that X_s follows the dynamics

$$dX_s = \widehat{h}_0(X_s, u_s)\,\mathrm{d}s + \tilde{\Sigma}\,\mathrm{d}W_s, \quad X_0 \in \mathbb{R}^{d_X}. \qquad (3)$$

The objective (2) in the offline phase of DARE is a novel adjusted formulation of the classical OC objective. Here, the agent explicitly rewards or penalizes uncertainty on their estimate of the environment. More precisely, When $\phi > 0$ (resp. < 0) the agent rewards (resp. penalizes) exploration, i.e., the agent is encouraged to visit areas of the environment with higher (resp. lower) uncertainty. We show in Section 6.3 that the exploration parameter ϕ is key to the performance of decision-making problems in noisy and non-stationary environments.

To solve the problem (2), the agent defines the value function

$$V(s,x) = \sup_u J(s,x;u). \qquad (4)$$

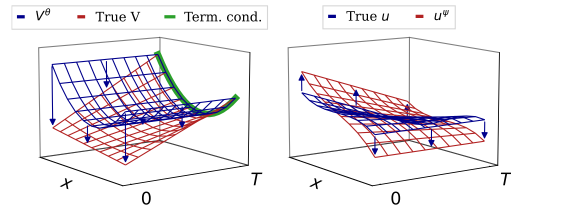


Figure 1. Illustration of the initialization of V^{θ_0}, u^{ψ_0} in the offline phase. The value function network is initialized around the terminal condition g in (2), and the control policy network is initialized around an affine approximation to the true optimal control.

We assume that the dynamic programming principle holds for $\mathbb{E}\left[\widehat{f}_0\right]$ and $\mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right]$, so V solves the HJB equation:

$$0 = V_t + \frac{1}{2}\mathrm{Tr}(\Sigma\, \nabla_{xx} V) + \sup_{u \in \mathbb{R}^{d_u}} H(x, u, \nabla_x V(t,x); \widehat{\mathfrak{p}}_0), \qquad (5)$$

subject to terminal condition $V(T, x) = g(x)$, where $\Sigma = \tilde{\Sigma}\,\tilde{\Sigma}^\intercal$. For $\ell \in \mathbb{R}^{d_X}$, the Hamiltonian H in (5) is defined as

$$H(x,u,p;\widehat{\mathfrak{p}}_0) = \widehat{h}_0(x,u)^\intercal \ell + \mathbb{E}\left[\widehat{f}_0\right](x,u)$$
$$- \phi\,\mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right](x, u(t,x)).$$

The policy u^*, which the agent implements at time $t = 0$, maximizes (2) and is obtained in feedback form, i.e., as a function of the system, for $s \in [0, T]$:

$$u^*(s,x;\widehat{\mathfrak{p}}_0) = \arg\max_{u \in \mathbb{R}} H(x, u, V_x(s,x); \widehat{\mathfrak{p}}_0), \qquad (6)$$

where $V(t, x)$ solves the nonlinear PDE (5).

In contrast to several approaches in RL which address the exploration-exploitation trade-off through penalization or reward of *random* control processes (see Wang et al., 2020, in a continuous-time setup), our method learns a control policy that is a *deterministic* function of the environment and which explores domain regions in which the agent is uncertain of their estimates of the OC pair.

Online Problem. At each time $t \in (0, T]$, the agent takes an action u_t, observes a noisy sample of the true environment $\mathfrak{p}(u_t, X_t) + \epsilon_t$ for some i.i.d. noise $\{\epsilon_t\}$, and updates their estimate $\widehat{\mathfrak{p}}_t$ accordingly.[1] Conditionally on the new estimate, the policy of the offline phase is not optimal. To adapt the optimal policy, the agent maximizes the updated objective, for $s \in [t, T]$:

[1] We do not assume a particular estimation procedure, but this can be achieved with function approximators suitable for online learning, e.g., Gaussian Processes or Bayesian DNNs as in (Duran-Martin et al., 2022).

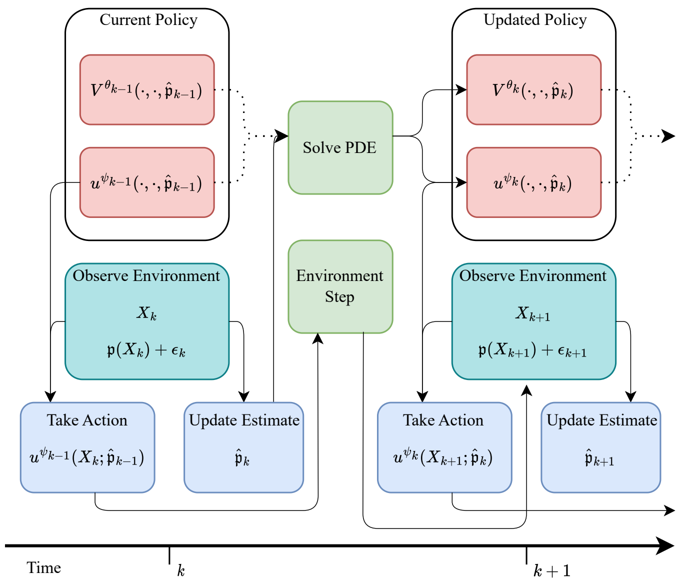


Figure 2. A schematic of DARE. We write k for t_k to simplify notation. At time k, the agent has value function V^{θ_k} and policy u^{ψ_k}. The agent observes the system X_k and environment $\mathfrak{p}(X_k)+\epsilon_k$. Next, the agent takes an action according to the policy u^{ψ_k} and updates the environment estimate to $\widehat{\mathfrak{p}}_{k+1}$. Between times k and $k+1$, the agent solves an updated OC problem according to the new estimate $\widehat{\mathfrak{p}}_{k+1}$, transferring knowledge (dotted line) from V^{θ_k}, u^{ψ_k} to compute $V^{\theta_{k+1}}, u^{\psi_{k+1}}$.

$$J(s,x;u) = \mathbb{E}\Bigg[g(X_T) - \int_s^T \mathbb{E}\left[\widehat{f}_t\right](X_r,u_r)\,\mathrm{d}r \qquad (7)$$
$$+\phi \int_s^T \mathrm{Var}\left[\widehat{\mathfrak{p}}_t\right](X_r,u(r,X_r))\,\mathrm{d}r \,\Bigg|\, \mathcal{G}_s\Bigg],$$

and follows similar steps as those of the offline phase.

When the uncertainty $\mathrm{Var}(\widehat{\mathfrak{p}}_t)$ is high and the agent rewards exploration, i.e., $\phi < 0$, the DARE policy focuses on improving the agent's estimate of their environment. As the estimation accuracy increases, the contribution of the variance term in the objective decreases. Hence, the DARE policy naturally balances the exploration-exploitation trade-off throughout the online phase. We demonstrate the benefit of the exploration term to overcome misspecified priors or nonstationary environments in Section 6.

4. The Deep Adaptive Regulator

DARE offline phase. To obtain the initial policy corresponding to the pair $\widehat{\mathfrak{p}}_0$, we use NN approximations $V^{\boldsymbol{\theta}}$ and $u^{\boldsymbol{\psi}}$ of the value function V and the optimal control u that solve the HJB (5). We initialize $V^{\boldsymbol{\theta}}$ and $u^{\boldsymbol{\psi}}$, for $s \in [0,T]$, as follows:

$$\begin{cases} V^{\boldsymbol{\theta}}(s,x) &= g(x) + \mathfrak{X}^{\boldsymbol{\theta}}(s,x), \\ u^{\boldsymbol{\psi}}(s,x) &= \widehat{u}_{X_0}(s,x) + \mathfrak{X}^{\boldsymbol{\psi}}(s,x), \end{cases} \qquad (8)$$

where g is the terminal reward function and $\widehat{u}_{X_0}$ is a locally optimal linear approximation of the true optimal control. We use the ILQG method of (Todorov & Li, 2005) to compute $\widehat{u}_{X_0}$ starting from X_0 and we set $\mathfrak{X}^{\boldsymbol{\theta}}$ and $\mathfrak{X}^{\boldsymbol{\psi}}$ to be fully-connected feedforward networks; see Figure 3.

To train $V^{\boldsymbol{\theta}}$ and $u^{\boldsymbol{\psi}}$, we devise a multi-objective loss function which considers (i) the HJB (5), (ii) the Hamiltonian satisfying first-order conditions, and (iii) the terminal condition. Similar to DGM, we use Monte Carlo integration to minimize the loss function on a compact domain $K \subset \mathbb{R}^{d_X}$. We let $\|\cdot\| = \|\cdot\|_{L^2([0,T]\times K)}$ and we reformulate (5) in a variational form to define the loss:

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\psi};\widehat{\mathfrak{p}}) = \mathcal{L}_{\text{HJB}} + \mathcal{L}_{\text{hamiltonian}} + \mathcal{L}_{\text{terminal}}\,, \qquad (9)$$

where

$$\begin{cases} \mathcal{L}_{\text{HJB}} = \|V_t^{\boldsymbol{\theta}} + \frac{\tilde{\Sigma}^2\, V_{xx}^{\boldsymbol{\theta}}}{2} + H(\cdot, u^{\boldsymbol{\psi}}(\cdot,\cdot), V_x^{\boldsymbol{\theta}}(\cdot,\cdot);\widehat{\mathfrak{p}})\|\,, \\ \mathcal{L}_{\text{hamiltonian}} = \|\partial_u H(\cdot, u^{\boldsymbol{\psi}}(\cdot,\cdot), V_x^{\boldsymbol{\theta}}(\cdot,\cdot);\widehat{\mathfrak{p}})\|\,, \\ \mathcal{L}_{\text{terminal}} = \|V^{\boldsymbol{\theta}}(T,\cdot) - g\|\,. \end{cases} \qquad (10)$$

The loss (9) is similar to that in (Al-Aradi et al., 2022). However, here we use a first-order condition for the Hamiltonian component, which we found to improve training performance when H is concave. Otherwise, we set

$$\mathcal{L}_{\text{hamiltonian}} = -\|H(\cdot,\cdot,u^{\boldsymbol{\psi}};\widehat{\mathfrak{p}})\|\,. \qquad (11)$$

We summarize the procedure in Algorithm A.

DARE online Phase. Let $\mathcal{T} = \{t_0,\dots,t_n\} \subset [0,T]$ be a set of potentially irregularly spaced times which are unknown to the agent at time $t=0$. In the online phase, the agent uses new estimates of the environment to update their control policy as follows. Suppose the agent calculated $V^{\boldsymbol{\theta}_{t_{k-1}}}(\cdot,\cdot;\widehat{p}_{t_{k-1}})$ and $u^{\boldsymbol{\psi}_{t_{k-1}}}(\cdot,\cdot;\widehat{p}_{t_{k-1}})$ at time t_k, where $\boldsymbol{\theta}_{t_{k-1}}$ and $\boldsymbol{\psi}_{t_{k-1}}$ minimize the loss $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\psi},\widehat{p}_{t_{k-1}})$. At time t_k, the agent (i) takes the action $u^{\boldsymbol{\psi}_{t_{k-1}}}(t_k, X_{t_k};\widehat{p}_{t_{k-1}})$ and (ii) computes the new estimate $\widehat{p}_{t_k}$. Then, over the period $[t_k,t_{k+1})$, the agent minimizes $\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\psi},\widehat{p}_{t_k})$ to compute the parameters $(\boldsymbol{\theta}_{t_k},\boldsymbol{\psi}_{t_k})$. This loss minimization uses a gradient-based method (e.g., ADAM), with $\left(\boldsymbol{\theta}_{t_{k-1}},\boldsymbol{\psi}_{t_{k-1}}\right)$ as a warm start for the neural networks; our method is outlined in Figure 3 and Algorithm A.

When the environment changes smoothly, we show in the next section that the value function and the control policy also change smoothly. Moreover, we show that the DNN parameters $\boldsymbol{\theta}_t$ and $\boldsymbol{\psi}_t$ change smoothly, justifying our approach.

5. Online Deep Transfer Learning

TL encompasses methods in which knowledge acquired from an initial source task is used to improve performance on a related target task (see Pan & Yang, 2009; Zhuang et al., 2020; Niu et al., 2020; Suder et al., 2023; Tan et al., 2018, for an overview of transfer learning). One can study the efficiency of `DARE` in the online phase from the perspective of TL because its performance hinges on successive transferring of knowledge (parameters) between DNNs corresponding to the solutions to "similar" OC problems; see Figure 3. In this section, we provide a theoretical justification for our method. More precisely, we analyze the smoothness of OC problems with respect to the OC pair describing the environment and the resulting smoothness of DNN parameters.

To provide a theoretical foundation to this claim, we use the tools of regular perturbation in OC and a notion of continuity of the `DARE` network parameters. Later, Section 6.1 explores specific examples and quantifies empirically the improvement achieved from TL in the online phase of `DARE`. In particular, we use the number of iterations required to attain, on average, a prespecified loss in the target task to measure the *strength of transfer*.

To streamline our analysis, assume $d_X = d_u = 1$ and consider an agent who receives observations of the OC pair and updates their estimate $\widehat{\mathfrak{p}}_t$, accordingly.[2] In practice, between two sufficiently close observation times $r, s \in [0,T]$ with $r < s$, we assume that the estimate $\widehat{\mathfrak{p}}_r$ at time r remains close to the estimate $\widehat{\mathfrak{p}}_s$ at time s. Hence, we write $\widehat{\mathfrak{p}}_s$ as a perturbation of $\widehat{\mathfrak{p}}_r$. This is formalized in the following assumption.

Assumption 5.1. For any ϵ , there are suitable perturbation functions p^f and p^h such that

$$\widehat{f}_s = \widehat{f}_r + \epsilon\, p^f \quad \text{and} \quad \widehat{h}_r = \widehat{h}_r + \epsilon\, p^h \,. \tag{12}$$

Fix $t \in [0,T]$ and consider the value and control functions associated to $\widehat{\mathfrak{p}}_r$ and $\widehat{\mathfrak{p}}_s$ on $[t,T]$. That is, for $\rho \in \{r,s\}$, let

$$V^\rho(t,x) = \sup_u \mathbb{E}\left[\widehat{g}(X_T^\rho) + \int_t^T \widehat{f}_\rho(X_\tau^\rho, u_\tau)\,\mathrm{d}\tau \,\middle|\, X_t^\rho = x\right], \tag{13}$$

where

$$\mathrm{d}X_\tau^\rho = \widehat{h}_\rho(X_\tau^\rho, u_\tau)\,\mathrm{d}\tau + \tilde{\Sigma}\,\mathrm{d}W_\tau \,.$$

Theorem 5.2 first shows that small perturbations in the OC pair lead to small perturbations in the optimal policy and the value function. Next, the result examines the continuity of the parameters of the DNNs approximating the value and control functions. This continuity is considered with respect to the function space in which the functions are defined. Intuitively, when a DNN is trained to a particular function, one expects that marginal changes to this function will result in marginal changes to the network parameters. Providing such a result in a general setting poses an intricate challenge. Thus, we simplify the setting by reducing the class of DNNs to that of single-layer perceptrons. However, our empirical findings suggest that it generalizes to more general cases.

Theorem 5.2. *Suppose that $\widehat{f}^r$, $\widehat{f}^s$, $\widehat{h}^r$, $\widehat{h}^s$, p^f, $p^h \in C_b^{1,2}([0,T];K)$ and ϵ is defined as in (12).*[3] *Moreover, assume that solutions $(V^r, u^{r,\star})$ and $(V^s, u^{s,\star})$ to (13) exist and are unique. For a value of ϵ which is sufficiently small, then there exists $L > 0$ such that for any $\gamma > 0$ and single-layer perceptron approximations $(V^{\boldsymbol{\theta}_r}, u^{\boldsymbol{\psi}_r})$ and $(V^{\boldsymbol{\theta}_s}, u^{\boldsymbol{\psi}_s})$ of (V^r, u^r) and (V^s, u^s), respectively, with precision γ, such that*

$$\|\boldsymbol{\theta}^r - \boldsymbol{\theta}^s\| + \|\boldsymbol{\psi}^r - \boldsymbol{\psi}^s\| \le L\epsilon^2 \,. \tag{14}$$

The proof of Theorem 5.2 is given in Appendix C. Although the above result justifies the performance of `DARE` for two consecutive policy updates with two fixed OC pairs, the results extend to the case of a dynamic estimate of the environment. That is, suppose $(h,f) = (h_t, f_t)$ evolves throughout $t \in [0,T]$. Then, if (h_t, f_t) changes smoothly, we expect the DNNs parameterizing the corresponding solutions to vary smoothly. Finally, our numerical results indicate that `DARE` also adapts efficiently to large and abrupt changes in the environment.

6. Numerical Experiments

This section investigates the performance of `DARE` in several setups. We use tractable OC and MPC examples to test our approach and to demonstrate that it produces sensible solutions in noisy and changing environments. In particular, we consider two classes of problems to test our method.

Linear-Quadratic-Gaussian. Consider the classical LQG setup where a system evolves with dynamics

$$\mathrm{d}X_t = (b + c\,u_t)\,\mathrm{d}t + \tilde{\Sigma}\,\mathrm{d}W_t\,, \quad X_0 \in \mathbb{R}, \tag{15}$$

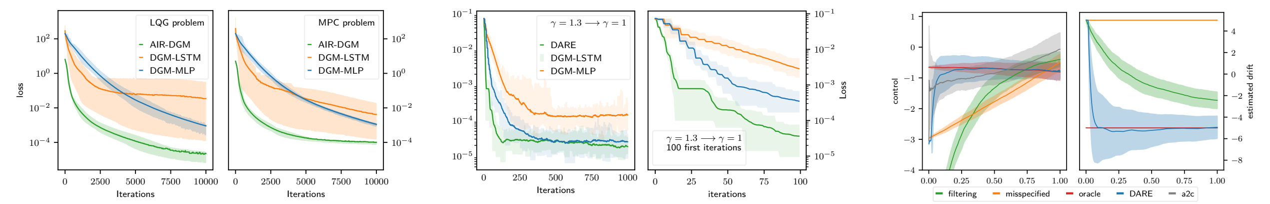
where b is a constant drift, $c > 0$ scales the linear impact of an agent on the system, and $\tilde{\Sigma} > 0$ is the variance of the observation noise. The agent maximizes the LQ criterion

$$\mathbb{E}\left[X_T - \alpha\, X_T^2 - \phi \int_0^T u_t^2\,\mathrm{d}t\right], \tag{16}$$

where $\phi > 0$ scales the running quadratic penalty and $\alpha > 0$ scales the terminal quadratic penalty. The running penalty $-\phi\, u^2$ is known, while the drift b is assumed unknown.

[2] It is straightforward to generalize to multi-dimensional setups.

[3] $C_b^{1,2}([0,T];K)$ denotes the set of functions defined on $[0,T] \times K$ with continuous first derivative in t and continuous and bounded second derivatives in x.



(a) Mean and std dev of the training loss (9) from 100 training tasks of DARE, DGM-MLP, and DGM-LSTM in the offline phase. We set $b = 0$ for LQG and $\gamma = 1.3$, $\varphi = 0$ for MPC.

(b) Mean and std dev of the training loss (9) from 100 training tasks of DARE, DGM-MLP, and DGM-LSTM in the online phase. We set $\gamma = 1$, and $\varphi = 0$ for MPC.

(c) Mean and std dev of control policy and drift estimation from 100 simulated paths for filtering, oracle, misspecified, DARE, and a2c in the online phase of the LQG problem.

Figure 3. Training loss and performance of DARE in the experiments of Sections 6.1 and 6.2.

Table 1. Default parameter values for the LQG problem.

PARAM.	b	c	σ	ϕ	α	x_0	T
VALUE	-5	1	1	1	0.3	10	1

Model Predictive Control. Let $\widehat{f}$ be a predictive model of the true nonlinear running penalty f. The system evolves according to (15) and the agent maximizes the objective

$$\mathbb{E}\left[X_T - \alpha X_T^2 - \phi \int_0^T \mathbb{E}[\widehat{f}](u_t)\,\mathrm{d}t - \varphi \int_0^T \mathrm{Var}[\widehat{f}](u_t)\mathrm{d}t\right]. \tag{17}$$

We fix the parameters of the LQG problem (15)-(16) in Table 1 and those of the MPC problem (15)-(17) in Table 2, unless otherwise noted.

Table 2. Default parameters values for the MPC problem.

Param.	b	c	σ	ϕ	φ	α	x_0	T
Value	0	1	1	0.15	0.1	0.05	100	1

6.1. Training performance

Offline. To demonstrate the effectiveness of our inductive bias (8), we compare DARE to DNN initializations of the value function and control policy without such bias. That is, we consider two methods, DGM-MLP and DGM-LSTM with initializations

$$V^{\boldsymbol{\theta}}(t, x) = \mathfrak{X}^{\boldsymbol{\theta}}(t, x) \quad \text{and} \quad u^{\boldsymbol{\psi}}(t, x) = \mathfrak{X}^{\boldsymbol{\psi}}(t, x)\,,$$

where $\mathfrak{X}^{\boldsymbol{\theta}}$ and $\mathfrak{X}^{\boldsymbol{\psi}}$ are feedforward DNNs with Xavier initialization in DGM-MLP and LSTM-like networks (Sirignano & Spiliopoulos, 2018) in DGM-LSTM.

In the MLPs of both DARE and DGM-MLP, there are 2 layers and 20 hidden units. In DGM-LSTM, there are two hidden LSTM-like layers between two single layer feedforward neural networks of width 20. Figure 3(a) shows that, on average, DARE substantially outperforms other solvers in convergence speed in both the LQG and MPC problems. While existing work emphasizes the importance of network architecture for performance, our findings indicate that inductive biases may hold greater importance.

Transfer Learning. We devise a TL experiment to investigate the ability of all approaches to adapt to changing environments. We define two running penalty functions $f_i = |u|^{1+\gamma_i}$ for $i \in \{0, 1\}$, where $\gamma_0 = 1.3$ and $\gamma_1 = 1$. We consider agents who use a Gaussian Process (GP) $\widehat{f}$ as a predictive model for the running penalty; see Appendix G for details on GPs. First, the agents fit two Gaussian Process $\widehat{f}_i$ for $i \in \{0, 1\}$ to ten noisy, random samples of the running penalty f_i. Next, we use DARE, DGM-MLP, and DGM-LSTM to solve for solutions $V^{\boldsymbol{\theta}_0}, u^{\boldsymbol{\psi}_0}$ relative to $\widehat{f}_0$. Once all methods have converged, we change the agents' estimate of the running penalty to $\widehat{f}_1$ and re-train $V^{\boldsymbol{\theta}_0}, u^{\boldsymbol{\psi}_0}$ with this updated penalty function. We record the loss in the adaptive phase in Figure 3(b).

All methods learn the new policy with comparable precision after a few thousands iterations, and we report training performance in Appendix D. However, DARE clearly transfers knowledge to the new environment more efficiently, taking less than 20 ADAM steps to achieve satisfactory precision. Each iteration lasts 0.00446 seconds on average in our experiments so DARE is suited for online problems with near continuous observations in nonstationary environments.

6.2. Online Performance: Filtering

To test the performance of DARE in the online phase, we devise an adaptive OC problem in the LQG setup. We assume that the true drift of the system (15) is $b = -5$, but the agent uses a misspecified prior $b_0 = 5$. Throughout the period $[0, T]$, the agent learns b through observations of X_t.

We compare our method to classical stochastic filtering. This approach is only suited to handle uncertainty in the drift, so our experiment assumes known running and terminal cost functions. The agent observes the state X_t of the system 1000 times throughout the period $[0, T]$, and we fix model parameters as in Table 1. We compare the following methodologies to solve the adaptive OC problem:

- **`oracle`**: the agent knows the true drift b. Standard results show that the optimal control is obtained analytically; see Appendix E.
- **`misspecified`**: the agent uses a misspecified prior b_0, does not update the drift estimate, and computes the optimal control analytically as in Appendix E.
- **`filtering`**: the agent assumes the drift is a random variable μ drawn from a Gaussian prior $\mathcal{N}(b_0, \Pi_0)$ where $\Pi_0 = 3$ and uses Bayesian learning to update the parameters of μ. We solve this problem rigorously in Appendix E. The optimal adaptive strategy is obtained analytically in (28).
- **`DARE`**: the value and control function networks are initialized as in (8) and trained in the offline phase using the misspecified drift b_0. In the online phase, the agent observes X_t and uses an exponential moving average with smoothing $\lambda = 0.95$ to estimates the drift b_t. After each update, we use 10 ADAM steps to update the optimal policy, i.e., the value and control functions $V^{\boldsymbol{\theta}}(t, x; b_t), u^{\boldsymbol{\psi}}(t, x; b_t)$.[4]
- **`A2C`**: Let $u^{\boldsymbol{\psi}}(t, x; b) = \mathfrak{X}^{\boldsymbol{\psi}}(t, x, b)$. The agent uses the drift estimation as a state variable to learn the solution to (15). More precisely, to train the algorithm offline, we use training data with drifts sampled from the prior distribution $\mathcal{N}(b_0, \Pi_0)$, where $\Pi_0 = 3$, and we use the same drift estimator as that in `DARE`. In the online phase, the agent does not update $\mathfrak{X}^{\boldsymbol{\psi}}$. See Appendix H for more details on the `A2C` implementation.

Figure 3(c) shows the distribution of the control policy and the estimated drift for 100 sample paths generated with the true drift b. The `misspecified` agent is not adaptive and the `filtering` agent suffers from misspecification ($b_0 << b^\star$ and Π_0 is small). `a2c` learns the `oracle` policy after a few iterations, however, the algorithm's performance degrades over time, see Appendix H for an in-depth discussion. On average, the adaptive optimal policy of `DARE` adapts quickly and remains close to the `oracle` policy throughout the online phase. Thus, our method is sufficiently flexible and robust to misspecification.

Finally, to test robustness with respect to the true drift, we run 100 simulations in which the value of the true drift is drawn from a distribution $b \sim \mathcal{N}(5, 3)$. We run the different algorithms described above to solve the adaptive OC problem for each simulated value of b. Table 3 shows the mean and standard deviation of the performance of each algorithm and showcases the superior performance of `DARE`. It is key to note that while `A2C` and other RL approaches need to train on (realistic) simulations of the environment, `DARE` adapts to new environments without any pre-training.

[4] In our simulations, each update iteration is performed, on average, in 0.00446 seconds; see Appendix D.

6.3. Online Performance: nonlinear MPC with GPs

Noiseless Cost Observations. Here, we test the performance of `DARE` for an MPC problem using GPs to approximate the nonlinear running penalty function. The true running penalty function is $u \mapsto |u|^{1+\gamma^*}$ with $\gamma^* = 1$ and the agent's prior is $\gamma_0 = 1.3$. In the online phase, the agent receives noiseless samples of the true penalty function evaluated at the agents' control; that is, the agent observes $|u^{\boldsymbol{\psi}}(t, X_t)|^{1+\gamma^*}$. We compare the following methods:

- **`oracle`**: the agent knows the true functional form of the running penalty $|u|^{1+\gamma^*}$ and solves (5) using the offline solver of `DARE` to obtains the optimal policy.
- **`misspecified`**: the agent uses the prior γ_0 and solves (5) using `DARE` to obtain the optimal policy.
- **`DARE`**: the agent fits a GP to the prior $|u|^{1+\gamma_0}$ and uses `DARE` (8) with the mean prediction of the GP to solve the offline problem. In the online phase, the agent receives noiseless observations $|u^{\boldsymbol{\psi}}(t, X_t)|^{1+\gamma^*}$ at 300 evenly spaced times in $[0, T]$, and uses a fixed lookback window of 15 points to update the GP estimation.[5] We set the exploration parameter $\varphi = 0$ because the observations are noiseless. At any time $t \in \mathcal{T}$, the agent updates the optimal policy with 30 ADAM steps.[6]

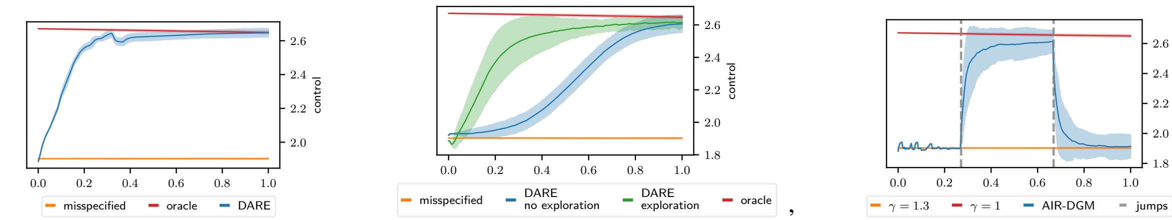
We test each method on 100 sample paths of the system (15). Figure 4(a) shows the distribution of the system X_t and the policy u_t throughout time. On average, `DARE` adapts the policy from the misspecified model of the environment to the true one quickly.

Exploration-Exploitation. We consider the case when the agent's observations of the running penalty in the online problem are corrupted by noise. That is, the agent observes $|u^{\boldsymbol{\psi}}(t, X_t)|^{1+\gamma^*} + \epsilon_t$ for $\epsilon_t \sim \mathcal{N}(0, .02)$. In this setting, it is beneficial for the agent to explore to ensure they have an accurate model of the running penalty function. To showcase the effect of exploration on performance, we test `DARE` with different values of the exploration parameter φ. Recall that $\varphi > 0$ penalizes exploration while $\varphi < 0$ rewards it.

We run 100 simulations and compare an agent that rewards exploration to an agent who is indifferent to exploration. Figure 4(b) and Table 3 show that in the presence of noise, adding an exploration term in the objective significantly improves the overall performance. In particular, the agent learns the true policy faster by encouraging exploration. Often, the absence of exploration in noisy environments leads to local optima in the value function and control policy network parameters, because a (wrong) mean prediction

[5] More precisely, the agent uses (at most) 15 points from the set of all observations gathered throughout $[0, t)$ for the GP prior.

[6] In our simulations, each update iteration is performed in 0.0024 seconds on average.



(a) Mean and std dev of policy for `oracle`, `misspecified`, and `DARE`.

(b) Mean and std dev of policy for `oracle`, `misspecified`, and `DARE` when $\varphi = 0$ (no exploration), and when $\varphi = 5 \cdot 10^{-3}$ (exploration).

(c) Mean and std dev of policy for `DARE` when the true value of γ jumps between 1.3 and 1.

Figure 4. Performance of `DARE` in the online phase for the MPC problem.

leads to a specific policy which prevents accurate learning of the cost function in the whole domain of controls.

Non-Stationary Environment. Finally, we consider a simulation setup where the true form of the cost function randomly switches between that of $\gamma_1^* = 1.3$ and $\gamma_2^* = 1$ according to a Poisson process with intensity 0.005, i.e., with 1.5 switches, on average, per simulation. We consider an environment which starts with the cost functional $\gamma_1^* = 1.3$, and the observations of the running penalty $|u^\psi(t, X_t)|^{1+\gamma^*} + \epsilon_t$ are corrupted with noise $\epsilon_t \sim \mathcal{N}(0, .1)$. Similar to the previous experiment, the agent uses a GP to model the running penalty.

To illustrate how `DARE` adapts to noisy and non-stationary environments, we draw and fix a Poisson path and run 100 simulations for the three methods described above. Figure 4(c) shows the distribution of the control policy u and illustrates the robustness of `DARE` to unpredictably changing environments. In particular, the policy followed by our methodology is, on average, close to the optimal one. We report the performance of `DARE` in Table 3 when the Poisson path is not fixed throughout simulations.

Table 3. Distribution of the final performance (mean, std dev) computed as $X_T - \alpha X_T^2 - \phi \int_0^T u_t^2 \, \mathrm{d}t$ for each simulation path.

Setup	Algorithm	Performance
filtering	`oracle`	$(-4.07, 6.55)$
	`DARE`	$(\mathbf{-4.16}, 6.44)$
	`a2c`	$(-4.20, 6.21)$
	`misspecified`	$(-5.95, \mathbf{5.60})$
	`filtering`	$(-9.81, 5.78)$
zero-noise	`oracle`	$(-8.82, 0.94)$
	`misspecified`	$(-8.83, 0.94)$
	`DARE`	$(\mathbf{-8.91}, \mathbf{0.94})$
noise	`oracle`	$(-8.83, 0.93)$
	`misspecified`	$(-8.91, 0.93)$
	`DARE` (no explor.)	$(-8.86, 0.93)$
	`DARE` (explor.)	$(\mathbf{-8.83}, \mathbf{0.92})$

6.4. High-Dimensional Control

In recent years, regulators have urged financial institutions to manage the risk of their trading activity within very large portfolios called central risk books. The aggregated trading activity of large institutions is often conducted at very high frequency and can be modeled as an OC problem. The controlled system is described by the agent's inventory $Q_t \in \mathbb{R}^d$, the asset prices $S_t \in \mathbb{R}^d$, and running wealth $X_t \in \mathbb{R}$, with dynamics:

$$\mathrm{d}Q_t = u_t \, \mathrm{d}t \,, \; \mathrm{d}S_t = \tilde{\Sigma} \, \mathrm{d}W_t \,, \; \mathrm{d}X_t = -u_t^\intercal S_t \, \mathrm{d}t - f(u_t) \, \mathrm{d}t \,,$$

where $u_t \in \mathbb{R}^d$ denotes the trader's speed of trading. The agent incurs transaction costs according to some unknown function of the trading speed $f(u_t) \in \mathbb{R}^d$, and maximizes the exponential utility of their terminal wealth for some estimate $\widehat{f}$ of the true transaction costs (See Appendix F for a detailed motivation for this problem):

$$\sup_v \mathbb{E}\Big[- \exp\left(-\gamma \left(X_T + Q_T^\intercal S_T - Q_T^\intercal \Gamma \, Q_T\right)\right) \Big] \,,$$

The left panel of Figure 5 shows the training performance of the three methods `DARE`, `DGM-LSTM`, and `DGM-MLP` in the offline phase, when f corresponds to $f : u \mapsto u^{\gamma \intercal} \, \eta \, u^\gamma$ where $\gamma = 1.3$. The last two panels of Figure 5 show the transfer strength when the exponent changes to $\gamma = 1$ and the agent must adapt their optimal policy. The results showcase the superior performance of `DARE` in the offline and online phases.

Impact Statement

This paper presents a novel deep learning methodology for solving decision-making problems in noisy and non-stationary environments, with wide-ranging applications in finance, robotics, and biology. Our contribution is a highly accurate and efficient method for solving model predictive control problems. Possible implications include more efficient and effective risk management in finance, safer robot-human interaction, and improved biomedical engineering. We use tractable examples to test our approach and to demonstrate that our model produces reasonable policies.

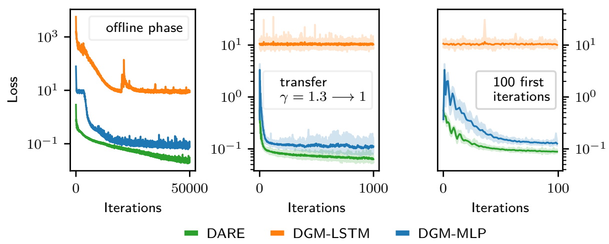


Figure 5. Training loss (9) of DARE, DGM-MLP, and DGM-LSTM, in the offline and online phase averaged through 100 training tasks for a 10−dimensional trading problem. Model parameters are $\Gamma = 10^{-2} I_{10}, \eta = 10^{-1} I_{10}, \gamma = 10^{-2}$, and Σ is a random positive semi-definite matrix.

Before implementing our model for critical problems, we believe further specific experimentation and validation is necessary.

References

Al-Aradi, A., Correia, A., Jardim, G., de Freitas Naiff, D., and Saporito, Y. Extensions of the deep galerkin method. *Applied Mathematics and Computation*, 430: 127287, 2022.

Allgower, F., Findeisen, R., Nagy, Z. K., et al. Nonlinear model predictive control: From theory to application. *Journal-Chinese Institute Of Chemical Engineers*, 35(3): 299–316, 2004.

Aristotelous, A. C., Mitchell, E. C., and Maroulas, V. Adlgm: An efficient adaptive sampling deep learning galerkin method. *Journal of Computational Physics*, 477: 111944, 2023.

Bachouch, A., Huré, C., Langrené, N., and Pham, H. Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *Methodology and Computing in Applied Probability*, 24(1):143–178, 2022.

Basei, M., Guo, X., Hu, A., and Zhang, Y. Logarithmic regret for episodic continuous-time linear-quadratic reinforcement learning over a finite-time horizon. *The Journal of Machine Learning Research*, 23(1):8015–8048, 2022.

Bensoussan, A. Perturbation methods in optimal control. *(No Title)*, 1988.

Bieker, K., Peitz, S., Brunton, S. L., Kutz, J. N., and Dellnitz, M. Deep model predictive flow control with limited sensor data and online learning. *Theoretical and computational fluid dynamics*, 34:577–591, 2020.

Cartea, Á., Jaimungal, S., and Penalva, J. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

Doya, K. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

Drissi, F. Solvability of differential riccati equations and applications to algorithmic trading with signals. *Applied Mathematical Finance*, 29(6):457–493, 2022. doi: 10.1080/1350486X.2023.2241130. URL https://doi.org/10.1080/1350486X.2023.2241130.

Duran-Martin, G., Kara, A., and Murphy, K. Efficient online bayesian inference for neural bandits. In *International Conference on Artificial Intelligence and Statistics*, pp. 6002–6021. PMLR, 2022.

Garcia, C. E., Prett, D. M., and Morari, M. Model predictive control: Theory and practice—a survey. *Automatica*, 25 (3):335–348, 1989.

Han, J., Jentzen, A., and E, W. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34): 8505–8510, 2018.

Hoglund, M., Ferrucci, E., Hernandez, C., Gonzalez, A. M., Salvi, C., Sanchez-Betancourt, L., and Zhang, Y. A neural rde approach for continuous-time non-markovian stochastic control problems, 2023.

Huré, C., Pham, H., Bachouch, A., and Langrené, N. Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis. *SIAM Journal on Numerical Analysis*, 59(1):525–557, 2021.

Iglesias, P. A. and Ingalls, B. P. *Control theory and systems biology*. MIT press, 2010.

Jia, Y. and Zhou, X. Y. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *The Journal of Machine Learning Research*, 23 (1):12603–12652, 2022.

Jia, Y. and Zhou, X. Y. q-learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61, 2023.

Jiang, D., Sirignano, J., and Cohen, S. N. Global convergence of deep galerkin and pinns methods for solving partial differential equations. *arXiv preprint arXiv:2305.06000*, 2023.

Kunisch, K. and Walter, D. Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation. *ESAIM: Control, Optimisation and Calculus of Variations*, 27:16, 2021.

Lenz, I., Knepper, R. A., and Saxena, A. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10, pp. 25. Rome, Italy, 2015.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mesbah, A. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

Mhaskar, H. N. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8 (1):164–177, 1996.

Mishra, P. K., Gasparino, M. V., Velasquez, A. E. B., and Chowdhary, G. Deep model predictive control, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

Niu, S., Liu, Y., Wang, J., and Song, H. A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, 1(2):151–166, 2020.

Onken, D., Nurbekyan, L., Li, X., Fung, S. W., Osher, S., and Ruthotto, L. A neural network approach for high-dimensional optimal control applied to multiagent path finding. *IEEE Transactions on Control Systems Technology*, 31(1):235–251, 2022.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359, 2009.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Salzmann, T., Kaufmann, E., Arrizabalaga, J., Pavone, M., Scaramuzza, D., and Ryll, M. Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, apr 2023. doi: 10.1109/lra.2023.3246839. URL https://doi.org/10.1109%2Flra.2023.3246839.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Suder, P. M., Xu, J., and Dunson, D. B. Bayesian transfer learning. *arXiv preprint arXiv:2312.13484*, 2023.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tallec, C., Blier, L., and Ollivier, Y. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pp. 6096–6104. PMLR, 2019.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pp. 270–279. Springer, 2018.

Todorov, E. and Li, W. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 300–306. IEEE, 2005.

van der Meer, R., Oosterlee, C. W., and Borovykh, A. Optimally weighted loss functions for solving pdes with neural networks. *Journal of Computational and Applied Mathematics*, 405:113887, 2022.

Wang, H., Zariphopoulou, T., and Zhou, X. Y. Reinforcement learning in continuous time and space: A stochastic control approach. *The Journal of Machine Learning Research*, 21(1):8145–8178, 2020.

Yildiz, C., Heinonen, M., and Lähdesmäki, H. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.

Yong, J. and Zhou, X. Y. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media, 1999.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A. Algorithms

Algorithm 1 : `DARE` - Offline Phase

Inputs:
- Initial OC pair $\widehat{\mathfrak{p}}_0 = (\widehat{h}_0, \widehat{f}_0)$
- Weights θ_0, ψ_0
- Area of integration $K \subset \mathbb{R}$
- Number of training iterations $N \in \mathbb{N}$
- Initial state X_0

$\widehat{u}_{X_0} \leftarrow \mathrm{ILQG}(X_0, \widehat{\mathfrak{p}}_0)$
$V^{\theta_0} \leftarrow \widehat{g}_0 + \mathfrak{X}^{\theta_0}$
$u^{\psi_0} \leftarrow \widehat{u}_{X_0} + \mathfrak{X}^{\psi_0}$
for $n = 1, ..., N$ **do**
  $\mathcal{D} \leftarrow$ *Batch of uniform samples of* $[0, T) \times K$
  $\ell \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(t,x)\in\mathcal{D}} \mathcal{L}(t, x, V^{\theta_0}, u^{\psi_0}, \widehat{\mathfrak{p}}_0)$
  $\theta_n, \psi_n \leftarrow \mathrm{ADAM}(\theta_{n-1}, \psi_{n-1}, \mathit{loss} = \ell)$
end for

Algorithm 2 : `DARE` - Online Phase

Input:
- OC pair $\widehat{\mathfrak{p}}_0 = (\widehat{h}_0, \widehat{f}_0)$
- Initial weights θ, ψ
- Area of integration $K \subset \mathbb{R}$
- Number of ADAM updates in offline step $N_{off} \in \mathbb{N}$
- Number of ADAM updates per online step $N_{on} \in \mathbb{N}$
- Time discretization $\mathcal{T} \subset (0, T]$
- Initial State X_0

$V^{\theta_0}, u^{\psi_0} \leftarrow \mathtt{DARE}(\widehat{\mathfrak{p}}_0, \theta, \psi, K, N_{off}, X_0)$
$t_{prev} \leftarrow 0$
$\widehat{\mathfrak{p}}_{prev} \leftarrow \widehat{\mathfrak{p}}_0$
for $t \in \mathcal{T}$ **do**
  $X_t \leftarrow \mathrm{System}(t, X_{t_{prev}}, u^{\psi_{t_{prev}}}(t, X_{t_{prev}}))$
  $\widehat{\mathfrak{p}}_t \leftarrow \mathrm{Approx}(\widehat{\mathfrak{p}}_{prev}, \widehat{\mathfrak{p}}_{prev}(X_t, u^{\boldsymbol{\psi}}_{t_{prev}}(t, X_t)) + \epsilon)$
  $V^{\theta_t}, u^{\psi_t} \leftarrow \mathtt{DARE}(\widehat{\mathfrak{p}}_t, \theta_{t_{prev}}, \psi_{t_{prev}}, t, N_{on})$
  $t_{prev} \leftarrow t$
end for

B. ILQG

We provide a brief overview of the ILQG method from (Todorov & Li, 2005) that we use to initialize the control policy. Let the system X_t evolve as:

$$\mathrm{d}X_t = h(X_t, u_t)\mathrm{d}t + \Sigma(X_t, u_t)\mathrm{d}W_t$$

and let the performance criterion be

$$J(t, x; u) = \mathbb{E}\left[g(X_T) + \int_t^T f(\tau, X_\tau, u_\tau)\mathrm{d}\tau\right] .$$

In this section, we assume that the agent seeks to *minimize* $J(t, x; u)$. Let $\overline{u}_t$ be a random *open-loop* control policy, and consider

$$\mathrm{d}\overline{X}_t = f(\overline{X}_t, \overline{u}_t) .$$

Next, we linearize the original system around $\overline{X}_t, \overline{u}_t$ and discretize time $k = \{0, \ldots, K\}$ with $\Delta t = \frac{T}{K-1}$ and $t_k = k\delta t$.

Define the discrepancies $\delta X_t = X_t - \overline{X}_t, \delta u_t = u_t - \overline{u}_t$, which evolve (approximately) as

$$\begin{aligned} \delta X_{k+1} &= A_k \delta X_k + B_k \delta u_k + \mathcal{C}_k(\delta u_k)\xi_k \\ \mathcal{C}_k &= c_{1,k} + C_{1,k}\delta u_k + \cdots + C_{d,d_u} \\ \mathrm{cost}_k &= q_k + \delta X_k^\intercal \boldsymbol{q}_k + \frac{1}{2}\delta X_k^\intercal Q_k \delta X_k \\ &\quad + \delta u_k^\intercal \boldsymbol{r}_k + \frac{1}{2}\delta u_k^\intercal R_k \delta u_k + \delta u_k^\intercal P_k \delta X_k , \end{aligned}$$

where $\delta X_0 = 0$, $\xi_k \sim N(0, I_{d_X})$,

$$A_k = I_{d_X} + \Delta t\, h_x \qquad\qquad \boldsymbol{q}_k = \Delta t\, f_x$$

$$B_k = \Delta t\, h_u \qquad Q_k = \Delta_t\, f_{xx}$$
$$c_{i,k} = \sqrt{\Delta t}\, \Sigma^i \qquad \boldsymbol{r}_k = \Delta t\, f_u$$
$$C_{i,k} = \sqrt{\Delta t}\, \Sigma^i_u \qquad R_k = \Delta t\, f_{uu}$$
$$q_k = \Delta t\, f \qquad P_k = \Delta t\, f_{ux}\,,$$

and $q_K = g$, $\boldsymbol{q}_K = g_x$, and $Q_K = g_{xx}$.

Above, all functions are evaluated at $\overline{X}_k, \overline{u}_k$, and Σ^i denotes the i-th row of Σ. It is shown in (Todorov & Li, 2005) that the optimal control to the linearized system δu^* is affine, with

$$\delta u^*(\delta X) = l_k + L_k\, \delta X\,. \tag{18}$$

When δu takes the form (18), the value function is quadratic and we write

$$V_k(\delta X) = s_k + \delta X_k^\intercal \boldsymbol{s}_k + \frac{1}{2}\delta X_k^\intercal S_k \delta X_k\,.$$

On can obtain an explicit representation of $S_k, \boldsymbol{s}_k, s_k$ by first defining

$$\begin{aligned}
\boldsymbol{g}_k &= \boldsymbol{r}_k + B_k^\intercal \boldsymbol{s}_k + \sum_i C_{i,k}^\intercal S_{k+1} c_{i,k} \\
G_k &= P_k + B_k^\intercal S_{k+1} A_k \\
H_k &= R_k + B_K^\intercal B_k + \sum_i C_{i,k}^\intercal S_{k+1} C_{i,k}\,,
\end{aligned}$$

which leads to the following equalities

$$\begin{aligned}
S_k &= Q_k + A_k^\intercal S_{k+1} A_k - L_k^\intercal H_k L_k + L_k^\intercal G_k + G_k^\intercal L_k \\
\boldsymbol{s}_k &= \boldsymbol{q}_k + A_k^\intercal \boldsymbol{s}_{k+1} + L_k^\intercal H_k l_k + L_k^\intercal \boldsymbol{g}_k + G_k^\intercal l_k \\
s_k &= q_k + s_{k+1} + \frac{1}{2}\sum_i c_{i,k} S_{k+1} c_{i,k} + \frac{1}{2} l_k^\intercal H_k l_k + l_k^\intercal \boldsymbol{g}_k\,,
\end{aligned}$$

where $S_K = Q_K, \boldsymbol{s}_K = \boldsymbol{q}_K, s_K = q_K$. Consequently, we obtain

$$\begin{aligned}
l_k &= -H_k^{-1} \boldsymbol{g}_k \\
L_k &= -H_k^{-1} G_k\,.
\end{aligned}$$

When f or g are not convex, H may have negative eigenvalues. This generally causes numerical issues due to the the minimization problem being unbounded. In this case, we use the Levenberg-Marquardt method to achieve an approximate inverse, by forcing all negative eigenvalues of H to be equal to some $\lambda > 0$.

C. Transfer Learning Neural Networks

C.1. Definitions

First, we introduce the notation used throughout the section.

Definition C.1 (Single-Layer Perceptron (SLP))**.** Denote $d_\mathrm{i}, d_\mathrm{h}, d_\mathrm{o} \in \mathbb{N}$, $\sigma : \mathbb{R} \to \mathbb{R}$. A *single-layer perceptron* is defined as

$$F : \begin{array}{ccc} \mathbb{R}^{d_\mathrm{i}} & \longrightarrow & \mathbb{R}^{d_\mathrm{o}} \\ x & \longmapsto & \sum_{i=1}^{d_\mathrm{h}} (C^\intercal)_i \phi \bullet (A_i x + b_i) \end{array}$$

with $A_i \in \mathbb{R}^{d_\mathrm{i}}$, $b_i \in \mathbb{R}$, $(C^\intercal)_i \in \mathbb{R}^{d_\mathrm{o}}$ for $i \in \{1, \ldots, d_\mathrm{h}\}$ and $\bullet$ denotes the component-wise application. We denote with $\theta := (A, b, C) \in \mathbb{R}^d$ the parameters of this SLP, with $d = d_\mathrm{i} d_\mathrm{h} + d_\mathrm{h} + d_\mathrm{o} d_\mathrm{h}$, and F_θ is an SLP with parameter θ.

C.2. Proof of Theorem 5.2

We split the proof of Theorem 5.2 into two results. Proposition C.2 shows that perturbations in the environment lead to perturbations of similar scale in the value function and the optimal policy of OC problems. Next, Proposition C.3 shows that for small perturbations of the value function and optimal policy, the parameters of the networks used to approximate these functions are continuous.

Proposition C.2. *There is a constant C such that*

$$|V^r(t,x) - V^s(t,x)| \leq C\,\epsilon^2\,, \tag{19}$$

$$|u^{r,\star} - u^{s,\star}| \leq C\,\epsilon\,. \tag{20}$$

Proof First, note that the assumption of Theorem 5.2 ensure that the functional J is well defined. Observe that the OC problem V^r is a perturbation of V^s but also V^s is a perturbation of V^r . In particular, first write

$$J^r(t,x,u) = \mathbb{E}\left[\widehat{g}(X_T^s) + \int_t^T \widehat{f}_s(X_\tau^s, u_\tau)\,\mathrm{d}\tau \,\middle|\, X_t^s = x\right] \tag{21}$$

$$J^r(t,x,u) = \mathbb{E}\left[\widehat{g}(X_T^r) + \int_t^T \widehat{f}_s(X_\tau^r, u_\tau)\,\mathrm{d}\tau \,\middle|\, X_t^r = x\right]\,. \tag{22}$$

Next, use Theorem 2.1 of Chapter III and Theorem 2.1 of Chapter IV in (Bensoussan, 1988) to write

$$\begin{cases} |V^r(t,x) - J^r(t,x,u^{s,\star})| & \leq C_0\,\epsilon^2, \\ |V^s(t,x) - J^s(t,x,u^{r,\star})| & \leq C_1\,\epsilon^2\,, \end{cases}$$

for suitable constants C_0 and C_1. Finally, use the asymptotic expansions (Section 2.3, Chapter III, in (Bensoussan, 1988)) to write

$$\begin{cases} |V^r(t,x) - J^s(t,x,u^{s,\star})| & \leq L_0\,\epsilon^2, \\ |V^s(t,x) - J^r(t,x,u^{r,\star})| & \leq L_1\,\epsilon^2\,, \end{cases}$$

for suitable constants L_0 and L_1. □

Proposition C.3. *Let $K \subseteq \mathbb{R}^{d_\mathrm{i}}$ be compact; $f \in C_b^2(K;\mathbb{R})$. There exists $\delta, L > 0$ such that for every $f' : K \to \mathbb{R}^{d_\mathrm{o}}$ with $\|f - f'\| < \delta$ and every $\gamma > 0$, there exists parameters $\theta, \theta' \in \mathbb{R}^d$ such that*

$$\|F_\theta - f\|_{C_b^2(K;\mathbb{R})} \leq \gamma, \tag{23}$$

$$\|F_{\theta'} - f'\|_{C_b^2(K;\mathbb{R})} \leq \gamma, \tag{24}$$

and

$$\|\theta' - \theta\| < L\|f - f'\|_{C_b^2(K;\mathbb{R})}\,, \tag{25}$$

where F_θ is a single-layer perceptron with ReLU activation of width d_h.

Proof Without loss of generality, assume that K includes an open set around 0, i.e. there exists $0 < \delta < \epsilon$ s.t. $\mathcal{D}^\delta - f \subseteq C_b^2(K;\mathbb{R})$, where

$$\mathcal{D}^\delta := \left\{ f' \in C_b^2(K;\mathbb{R}) \middle| \|f - f'\|_{C_b^2(K;\mathbb{R})} < \delta \right\}.$$

Fix an arbitrary $\gamma > 0$. According to (Mhaskar, 1996), Theorem 2.1, we can find a hidden dimension $d_\mathrm{h} \in \mathbb{N}$, a matrix $A \in \mathbb{R}^{d_\mathrm{h} \times d_\mathrm{i}}$, a vector $b \in \mathbb{R}^{d_\mathrm{h}}$, and a continuous linear functional $\mathcal{C} : C_b^2(K;\mathbb{R}) \to \mathbb{R}^{d_\mathrm{h}}$ such that

$$\|f' - F_{(A,b,\mathcal{C}(f'))}\|_p \leq \gamma, \quad f \in \mathcal{D}^\delta.$$

Since for $f' \in \mathcal{D}^\delta$ holds

$$F_{(A,b,\mathcal{C}(f'))} = F_{(A,b,\mathcal{C}(f'-f))} + F_{(A,b,\mathcal{C}(f))},$$

due to linearity of $\mathcal{C}$ and Definition C.1, we have

$$\|\theta' - \theta\| = \|\mathcal{C}(f' - f)\| \leq L\|f - f'\|_{C_b^2(K;\mathbb{R})}$$

where we used the fact that the operator norm $\|\mathcal{C}\|_T$ of a continuous operator is finite. We conclude $\|\theta' - \theta\| < \epsilon$ and finish the proof. □

Table 4. Run time of different algorithms in the experiments.

Section	Test	Algorithm	Run time in seconds per 1000 iteration
6.1 Training performance	Offline LQG (Figure 3(a))	DGM-LSTM	4.26
		DGM-MLP	1.95
		DARE	2.24
	Offline MPC (Figure 3(a))	DGM-LSTM	6.02
		DGM-MLP	2.28
		DARE	2.34
	Transfer Learning (Figure 3(b))	DGM-LSTM	7.84
		DGM-MLP	3.90
		DARE	4.46
6.2 Filtering	Online phase (Figure 3)	DARE	7.47
6.3 MPC	No noise (Figure 4(a))	DARE	2.40
	Noise (Figure 4(b))	DARE	2.46
	Non-stationary (Figure 4(c))	DARE	2.52
6.4 High-dimensional	Offline phase (Figure 5)	DGM-LSTM	6.01
		DGM-MLP	2.41
		DARE	4.01
	Online phase (Figure 5)	DGM-LSTM	6.64
		DGM-MLP	3.26
		DARE	3.92

D. Performance

We report training performance of all the methods tested in Section 6 in Table 4. All tests have been conducted on a standard MacBook Pro M1. We make our code public in the following (anonymized) repo: `https://anonymous.4open.science/r/dare-7136/README.md`

E. Filtering mathematics

E.1. Perfect knowledge of the drift

This section solves the OC problem (16) when the agent fixed the value of the drift and does not update their belief throughout the time window.

When the drift is known and fixed, the OC problem (16) can be solved with standard methods (Yong & Zhou, 1999), and is

$$u^\star = \frac{c}{2\,\phi}\,\left(2\,A\,(t)\;x + B\,(t) + 1\right)\,, \tag{26}$$

where A and B solve the ODE system

$$\begin{cases} -A^{'}\,(t) = \frac{c\,A(t)^2}{2\,\phi} \\ -B^{'}\,(t) = 2\,\mu\,A\,(t) + \frac{c^2\,A(t)\,(B(t)+1)}{\phi}\,. \end{cases} \tag{27}$$

E.2. Bayesian filtering of the Gaussian drift

This section solves the OC problem when the agent uses a Gaussian prior to continuously update their estimation of the drift throughout the time window of the OC problem.

Consider the control problem in (16). When the agent uses a Gaussian prior $\mathcal{N}\,(b_t, \Pi_0)$ for μ then it can be shown that the dynamics of x can be written

$$dx_t \;=\; \beta_t\,\mathrm{d}t + c\,u_t\,\mathrm{d}t + \sigma\,d\widehat{W}_t$$

in a different filtration in which $\widehat{W}$ is a Gaussian process. $\beta_t = \mathbb{E}[\mu|\mathcal{F}_t]$ is the best estimate of μ at time t and can be obtained analytically as

$$\beta_t = -\frac{\mathit{\Pi}(t)}{\sigma}\left(x_0 - \frac{\sigma\, b_0}{\Pi_0} - x_t + q_t\right)$$

where $\mathit{\Pi}(t) = \left(\mathit{\Pi}_0^{-1} + \frac{t}{\sigma}\right)^{-1}$ and $q_t = \int_0^t c\, u_t\, \mathrm{d}t$.

Using the learning dynamics above to solve the control problem (see (Drissi, 2022) for details) gives the optimal control $\tilde{u}^\star$ given by

$$\tilde{u}^\star = \frac{c}{2\,\phi}\,\left(2\,A\,(t) + B\,(t)\right)\,x + \left(2\,C\,(t) + B\,(t)\right)\,q \tag{28}$$

$$+ \left(1 + D\,(t) + \,E\,(t)\right)\,, \tag{29}$$

where A, B, C solve the Riccati equation in

$$P\,(t) = \begin{pmatrix} A\,(t) & \frac{1}{2}B\,(t) \\ \frac{1}{2}B\,(t)^\intercal & C\,(t) \end{pmatrix}$$

$$0 = P'\,(t) + Y\,(t)^\intercal\,P\,(t) + P\,(t)\,Y\,(t) + P\,(t)\,U\,P\,(t)\,,$$

$$Y\,(t) = \begin{pmatrix} \frac{\mathit{\Pi}(t)}{\sigma} & \frac{\mathit{\Pi}(t)}{\sigma} \\ 0 & 0 \end{pmatrix},\quad U = \begin{pmatrix} \frac{c^2}{\phi} & \frac{c^2}{\phi} \\ \frac{c^2}{\phi} & \frac{c^2}{\phi} \end{pmatrix},$$

$$P\,(T) = \begin{pmatrix} -\alpha & 0 \\ 0 & 0 \end{pmatrix},$$

and D and E solve the ODE system

$$\begin{cases} 0 = & D^{'}\,(t) + 2\,\overline{\mathit{\Pi}}\,\mathit{\Pi}\,(t)\;A\,(t) - \mathit{\Pi}\,(t)\,(1 + D\,(t)) \\ & + \frac{c^2}{4\,\phi}\,\left(2\,A\,(t) + B\,(t)\right)^2 \\ 0 = & E^{'}\,(t) + \overline{\mathit{\Pi}}\,\mathit{\Pi}^\intercal\,B\,(t) + \mathit{\Pi}^\intercal\,\left(1 + D\,(t)\right) \\ & + \frac{c^2}{4\,\phi}\,\left(2\,C\,(t) + B\,(t)\right)^2\,, \end{cases}$$

with terminal conditions $D(T) = E(T) = 0$.

F. Algorithmic trading in high dimension

We motivate the multidimensional setup in our experiments of Section 6.4. Consider the case of the trading desk of a large bank that must execute a number $d \in \mathbb{N}^*$ of large transactions in d correlated financial assets throughout a trading window $[0, T]$. The trading desk must minimize their trading costs while minimizing the risk of their positions. Throughout this section, we consider a filtered probability space $\left(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}\right)$, with $T > 0$, satisfying the usual conditions and supporting all the processes we introduce.

Let $Q_0 \in \mathbb{R}^d$ represent the transaction sizes in every asset. The inventory of the agent is modeled by $(Q_t)_{t\in[0,T]} = \left(Q_t^1, \ldots, Q_t^d\right)^\intercal_{t\in[0,T]}$ and it evolves with the trading speed $(u_t)_{t\in[0,T]} = (u_t^1, \ldots, u_t^d)^\intercal_{t\in[0,T]}$ in each asset:[7]

$$dQ_t = u_t\,\mathrm{d}t.$$

The prices $(S_t)_{t\in[0,T]} = \left(S_t^1, \ldots, S_t^d\right)^\intercal_{t\in[0,T]}$ of the d assets are modeled as correlated Brownians with dynamics

$$dS_t = \tilde{\Sigma}\,\mathrm{d}W_t\,,$$

where $W = \left(W^1, \ldots, W^d\right)$ is a d-dimensional standard Brownian motion and $S_0 \in \mathbb{R}^d$ is known. The matrix $\tilde{\Sigma} \in \mathcal{M}_d(\mathbb{R})$ measures the correlation of the prices and we define the covariance matrix $\Sigma = \tilde{\Sigma}\,\tilde{\Sigma}^\intercal \in S_d^{++}(\mathbb{R})$.[8]

[7] The superscript $^\intercal$ is the transpose operator.

[8] $\mathcal{M}_d(\mathbb{R}) := \mathcal{M}_{d,d}(\mathbb{R})$ is the set of $d \times d$ real square matrices, $\mathcal{S}_d(\mathbb{R})$ is the set of real symmetric $d \times d$ matrices, and $\mathcal{S}_d^{++}(\mathbb{R})$ is the set of positive matrices.

Trading activity of the agent generates transaction costs, driven by some function of the trading speed $f(u_t)$ so the cash from their trading activity evolves as

$$\mathrm{d}X_t = -u_t^\intercal S_t \,\mathrm{d}t - f(u_t)\,\mathrm{d}t, \quad X_0 = 0.$$

The agent maximizes the exponential utility of their terminal wealth so their objective is

$$\begin{aligned} V(t,x,q,s) = &\sup_v \mathbb{E}\Big[- \exp\Big(-\gamma\big(Q_T^\intercal S_T - Q_T^\intercal \Gamma\, Q_T & (30) \\ & - \int_t^T u_s^\intercal S_s \,\mathrm{d}s - \int_t^T f(u_s)\,\mathrm{d}s\big)\Big)\Big], & (31) \end{aligned}$$

for values $Q_t = q$, $X_t = x$, and $S_t = s$ at time t.

The dynamic programming principle holds and the HJB equation associated with the problem

$$\begin{aligned} 0 = &\partial_t V + \frac{1}{2}\mathrm{Tr}\left(\Sigma\, D^2_{SS} V\right) \\ &+ \sup_{u\in\mathbb{R}^d} \left(-(u^\intercal s + f(u))\partial_x V + v^\intercal \nabla_q V\right), \end{aligned} \tag{32}$$

with terminal condition

$$V(T,x,q,s) = -\exp\left(-\gamma\left(q^\intercal s - q^\intercal \Gamma q\right)\right). \tag{33}$$

In the experiment of Section 6.4, we solve the HJB (32)-(33) using `DARE` to obtain the optimal policy of the trading agent.

When all the parameters of the problem are known and fixed, i.e., the agent does not adapt to new information, the problem described above admits an analytical solution which we use to study the performance of `DARE`.

To solve the problem semi-analytically, the function f must be a quadratic form, that is, there is some $\eta \in S_d^{++}(\mathbb{R}^{d\times d})$ with

$$f(u) = u^\intercal\, \eta\, u. \tag{34}$$

We follow the standard steps in linear-exponential quadratic Gaussian (LEQG) control and we propose the following form for the value function

$$\begin{aligned} &V(t,x,q,s) = \\ &\quad -\exp\left(-\gamma\left(x + q^\intercal S + Q^\intercal A(t) q + B(t)^\intercal q + C(t)\right)\right), \end{aligned}$$

and straightforward calculations find that the problem reduces to solving the following ODE system

$$\begin{cases} A'(t) = \frac{\gamma}{2}\Sigma - A(t)\eta^{-1}A(t) \\ B'(t) = -A(t)\eta^{-1}B(t) \\ C'(t) = -\frac{1}{4}B(t)^\intercal \eta^{-1} B(t), \end{cases} \tag{35}$$

with terminal conditions

$$A(T) = -\Gamma, \quad B(T) = C(T) = 0. \tag{36}$$

Clearly, the solutions for B and C are $B = C = 0$. To obtain a solution, we use the change of variables

$$a(t) = \eta^{-\frac{1}{2}} A(t) \eta^{-\frac{1}{2}} \quad \forall t \in [0,T],$$

so the problem reduces to the following terminal value problem

$$\begin{cases} a'(t) = & \hat{A}^2 - a(t)^2 \\ a(T) = & -C, \end{cases} \tag{37}$$

where

$$\hat{A} = \sqrt{\frac{\gamma}{2}} \left(\eta^{-\frac{1}{2}} \, \Sigma \, \eta^{-\frac{1}{2}} \right)^{\frac{1}{2}} \in \mathcal{S}_d^{++}(\mathbb{R}) \, ,$$

and

$$C = \eta^{-\frac{1}{2}} \Gamma \eta^{-\frac{1}{2}} \in \mathcal{S}_d^{+}(\mathbb{R}) \, .$$

We solve (37) in the next result.

Proposition F.1. *Define* $\xi : [0, T] \to \mathcal{S}_d(\mathbb{R})$

$$\begin{aligned} \xi\left(t\right) = & -\frac{\hat{A}^{-1}}{2} \left(I - e^{-2\hat{A}(T-t)} \right) \\ & - e^{-\hat{A}(T-t)} \left(C + \hat{A} \right)^{-1} e^{-\hat{A}(T-t)} \end{aligned} \tag{38}$$

as the unique solution to the ODE system

$$\begin{cases} \xi'(t) = \hat{A}\xi(t) + \xi(t)\hat{A} + I_d \\ \xi(T) = -\left(C + \hat{A} \right)^{-1} . \end{cases} \tag{39}$$

Then $\forall t \in [0, T]$, $\xi(t)$ *is invertible and*

$$a : t \in [0, T] \to \hat{A} + \xi(t)^{-1} \in \mathcal{S}_d(\mathbb{R})$$

is the unique solution of (37).

Thus, the value function, which we use as the oracle in Section 6.4 is given by

$$\begin{aligned} & V(t, x, q, s) = \\ & \quad - \exp\left(-\gamma \left(x + q^{\intercal} S + Q^{\intercal} A(t) q + B(t)^{\intercal} q + C(t) \right) \right) , \end{aligned}$$

where

$$\begin{aligned} A(t) = \eta^{\frac{1}{2}} \Bigg(\hat{A} - \Bigg\{ & \frac{\hat{A}^{-1}}{2} \left(I - e^{-2\hat{A}(T-t)} \right) \\ & + e^{-\hat{A}(T-t)} \left(C + \hat{A} \right)^{-1} e^{-\hat{A}(T-t)} \Bigg\}^{-1} \Bigg) \eta^{\frac{1}{2}} . \end{aligned}$$

Finally, Figure F shows the true value function and the solution learned by `DARE` for a set of model parameters in dimension 5, and Figure 6 shows the associated training loss.

G. Gaussian Process mathematics

Formally, a GP is a random function $f : \mathcal{X} \mapsto \mathbb{R}$, such that, for any finite set of points $\mathbf{X}_\star \subseteq \mathcal{X}$, the random vector $\boldsymbol{f_\star} = \{f(\boldsymbol{x})\}_{\boldsymbol{x} \in \mathbf{X}_\star}$ follows a multivariate Gaussian distribution. The shape of the function f is determined by a finite set of (training) observations $\boldsymbol{y} = \{y_i\}_{i \in \{1,\dots,n\}}$ collected at the (training) observation points $\mathbf{X} = \{\mathbf{x}_i\}_{i \in \{1,\dots,n\}}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$ is subject to i.i.d. Gaussian measurement noise $\epsilon_i \sim \mathcal{N}(0, s^2)$ for $s > 0$. GPs are fully specified by a mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and a covariance (kernel) function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, In particular, if $f \sim \mathcal{GP}(\mu, k)$ and $\mathbf{X}_\star$ is a set of test points in the domain $\mathcal{X}$ of the GP, then the set of random variables $\boldsymbol{f_\star}$ is Gaussian with parameters $\mathcal{N}(\boldsymbol{\mu}_\star, \boldsymbol{K}_{\star,\star})$, where

$$\boldsymbol{\mu}_\star = \{\mu(\boldsymbol{x})\}_{\boldsymbol{x} \in \mathbf{X}_\star} \quad \text{and} \quad \boldsymbol{K}_{\star,\star} = \{k(\boldsymbol{x}, \boldsymbol{x'})\}_{(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbf{X}_\star} .$$

A convenient property of GPs is that one computes the posterior distribution with analytic formulae. Suppose we collect n noisy observations $\boldsymbol{y} = \{y_1, \dots, y_n\}$ at the domain points $\mathbf{X} = \{\boldsymbol{x_1}, \dots, \boldsymbol{x_n}\}$, where $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}\left(0, s^2\right)$.

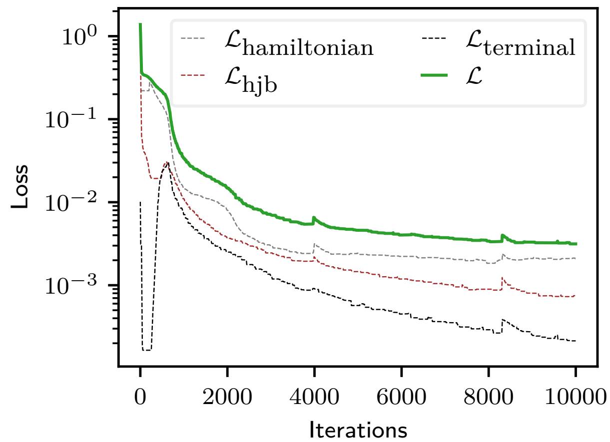


Figure 6. Training loss (9) of `DARE` for the multidimensional OC problem (30). The parameter values are $\Gamma = 10^{-2} \times I_5$,

$$\eta = 10^{-3} \times \begin{pmatrix} 25.13 & 10.41 & 11.67 & 13.75 & 22.21 \\ 10.41 & 6.42 & 7.68 & 9.12 & 12.4 \\ 11.67 & 7.68 & 12.95 & 11.2 & 18.71 \\ 13.75 & 9.12 & 11.2 & 17.04 & 17.25 \\ 22.21 & 12.4 & 18.71 & 17.25 & 29.02 \end{pmatrix}, \text{ and } \Sigma = \begin{pmatrix} 2.83 & 2.02 & 2.53 & 1.91 & 1.59 \\ 2.02 & 1.96 & 2.04 & 1.28 & 1.31 \\ 2.53 & 2.04 & 2.85 & 2.04 & 1.32 \\ 1.91 & 1.28 & 2.04 & 1.76 & 0.96 \\ 1.59 & 1.31 & 1.32 & 0.96 & 1.06 \end{pmatrix}.$$

Then, the posterior distribution over f given the previous (training) observations $\mathbf{X}$ and $\boldsymbol{y}$, is also a GP with mean function μ_{post} and covariance function k_{post} given by

$$\begin{cases} \mu_{\text{post}}(\boldsymbol{x}_\star) & = \boldsymbol{k}(\boldsymbol{x}_\star, \mathbf{X})(\boldsymbol{K} + s^2\,\boldsymbol{I})^{-1}\,\boldsymbol{y}\,, \\ k_{\text{post}}(\boldsymbol{x}_\star, \boldsymbol{x}'_\star) & = k(\boldsymbol{x}_\star, \boldsymbol{x}'_\star) \\ & \quad -\boldsymbol{k}(\boldsymbol{x}_\star, \mathbf{X})(\boldsymbol{K} + s^2\,\boldsymbol{I})^{-1}\,\boldsymbol{k}(\mathbf{X}, \boldsymbol{x}'_\star)\,, \end{cases} \tag{40}$$

where

$$\boldsymbol{k}(\boldsymbol{x}_\star, \mathbf{X}) = \boldsymbol{k}(\mathbf{X}, \boldsymbol{x}_\star)^\intercal = (k(\boldsymbol{x}_\star, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_\star, \boldsymbol{x}_n))$$

is the n-dimensional covariance vector of the test point $\mathrm{x}_\star$ with training points $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{K} = (k(\boldsymbol{x_i}, \boldsymbol{x_j}))_{i,j\in\{1,\ldots,n\}}$ is the positive semi-definite kernel matrix from training data, and $\boldsymbol{I}$ is the n-dimensional identity matrix. See Figure G for an example.

Let the elements of the vector $\boldsymbol{\theta} \in \Theta$ be hyper-parameters of the prior's kernel function and s^2 is the variance of the i.i.d. Gaussian noise that corrupts reward observations. Both $\boldsymbol{\theta}$ and s^2 are inferred with the log marginal likelihood of the data given by

$$\begin{aligned} L(\boldsymbol{\theta}, s) &= \log p(\,\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\theta}, s) \\ &= -\frac{1}{2}\log\Big(\det\left(\boldsymbol{K_\theta} + s^2\,\boldsymbol{I}\,\right)\Big) \\ &\quad -\frac{1}{2}\,\boldsymbol{y}^\intercal\left(\boldsymbol{K_\theta} + s^2\,\boldsymbol{I}\,\right)^{-1}\boldsymbol{y} - \frac{n}{2}\log(2\,\pi)\,, \end{aligned} \tag{41}$$

for a zero-mean GP, where $\mathbf{X}$ and $\boldsymbol{y}$ are the n training samples and $\boldsymbol{K_\theta}$ is the prior's positive covariance matrix with kernel $k_{\boldsymbol{\theta}}$. The vector of hyper-parameters $\boldsymbol{\theta}$ and the variance s^2 maximize the quantity (41), i.e., $(\boldsymbol{\theta}^\star, s^\star) \in \underset{\boldsymbol{\theta}\in\Theta, s\in\mathbb{R}^+}{\arg\max}\ L(\boldsymbol{\theta}, s)$, which one solves with classical gradient descent-based optimization algorithms.

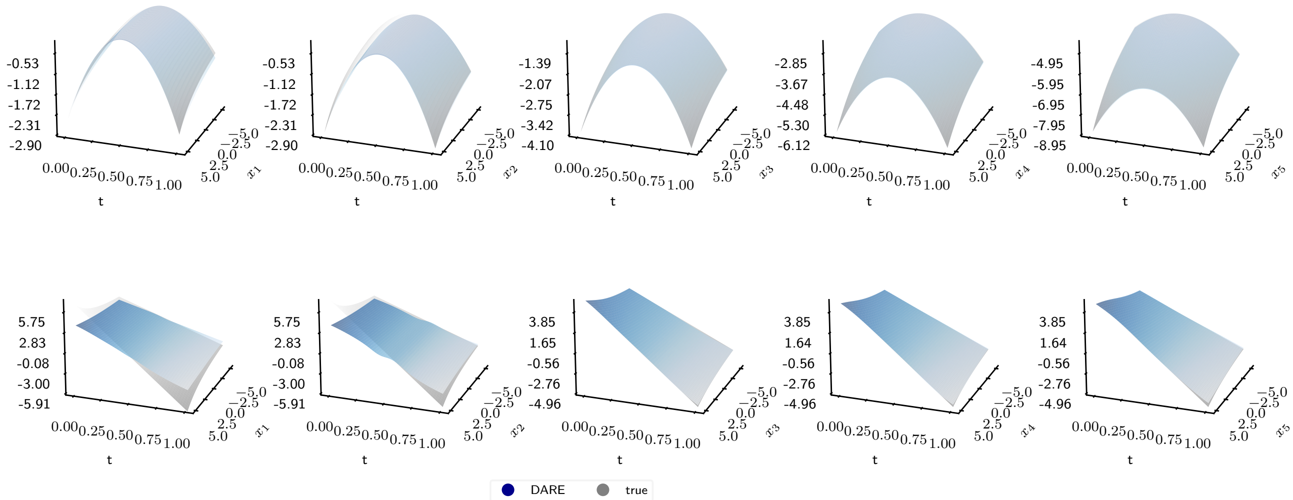


Figure 7. True and approximated (with `DARE`) value function (30) for $t \in [0, T]$ and $\boldsymbol{X} = X_1, X_2, X_3, X_4, X_5 \in [-5, 5]^5$. Each surface corresponds to the value function for time and one dimension in X, where the value of the system in all other dimensions is fixed to $x_i = 0$.

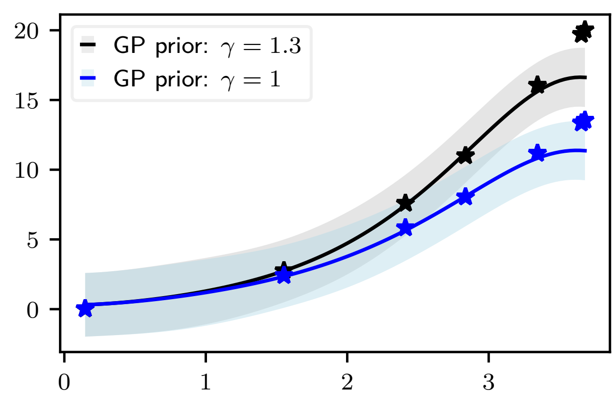


Figure 8. Two GPs fitted to $f(u) = u^{1+\gamma_i}$ for $\gamma_0 = 1.3$ and $\gamma_1 = 1$.

H. Reinforcement Learning Benchmarks

This section discusses the use of A2C as a benchmark for reinforcement learning in Section 6.2.

H.1. The A2C benchmark in Section 6.2

We use the implementation of `A2C` in (Mnih et al., 2016) from StableBaselines3 (Raffin et al., 2021). We use default parameters except for the number of steps necessary to update the policy, which we modify to account for the high stochasticity in the decision-making task. We found it is beneficial to process 10 episodes or more before updating. With this, it becomes clear that the time discretization played an important role in the performance of `A2C`, which is supported by the findings in (Tallec et al., 2019). For this reason, we use a standard technique for RL in continuous-time environments to augment performance, which is that during training, we hold the actions of the `A2C` agent constant for n time steps. At test time, the agent is allowed to act at all time steps. According to Appendix H.1, the performance of `A2C` only becomes competitive when actions in training are held constant for 100 time steps, that is, the agent makes only 10 actions in Figure 3.

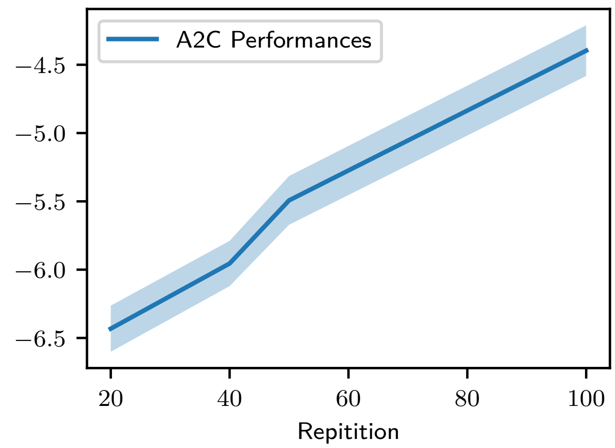


Figure 9. `A2C` performances in the setting of Section 6.2 for varying action repetitions, with performance estimation error.

In the LQG example, competitive performance is achieved likely due to the true optimal control being affine, so that linear interpolation between sparsely sampled optimal actions can lead to a decent approximation. In examples where the control is nonlinear, a finer time discretization is likely needed to approximate the true optimal control well.

In our experimental setup of Section 6.2, we also evaluated the `PPO` algorithm from StableBaselines3 (Raffin et al., 2021) as summarized in Table 5. While the algorithm demonstrated satisfactory performance, we observed several limitations impacting its suitability for our specific task. Firstly, `PPO` exhibited less stability compared to other baselines, with a noticeable sensitivity to hyperparameter configurations. This characteristic necessitated a more meticulous and often trial-and-error approach to hyperparameter tuning, which was less efficient in our context. Additionally, we encountered a fundamental challenge with the `PPO` algorithm similar to that experienced with the A2C approach, pertaining to the breakdown of reinforcement learning strategies in continuous time environments. To mitigate this, we were compelled to adopt a strategy of repeating actions, analogous to our approach with A2C.

Table 5. Performance of PPO in Section 6.2 for various hyper parameters.

MINI BATCH	512	256	256	100	64
STEP P. UPDATE	8192	8192	8192	1000	2048
NUM. EPOCH	100	10	100	10	10
ACT. REPIT.	100	100	10	100	10
PERF. (MEAN)	-4.22	-4.68	-5.46	-4.35	-6.61
PERF. (STD DEV)	5.48	5.27	4.97	5.51	6.38

H.2. Function-valued uncertainty

In Section 6.3, our decision to exclude an RL benchmark was informed by the fact that we use a GP to model uncertainty. In Section 6.2, uncertainty about the drift was parametric and could be incorporated as a state variable for the RL agent. Hence, the RL agent could be trained on samples of different drifts and learn the optimal policy for all drifts in the sampled region. In Section 6.3, the GP estimate could not be incorporated directly as a state variable for the RL agent, because the GP is a non-parametric function approximator. Tailoring an RL algorithm to optimize over a set of non-parametric cost functions was out of the scope of this paper.