CARL: Control-Augmented Reinforcement Learning for Decision-Making with Stochastic Factors

Sinong Geng ¹  Houssam Nassif²  Zhaobin Kuang³  Max Reppen⁴  Ronnie Sircar⁵

Abstract
We study finite-horizon dynamic decision-making in the presence of uncontrolled system variables, referred to as stochastic factors. Existing solutions fall into two distinctive categories: (i) Reinforcement learning employs end-to-end policy learning with flexible factor representation, but does not precisely model the dynamics of the problem; (ii) control-based methods, in contrast, take advantage of explicitly modeled dynamics via model fitting (calibration) but is constrained on factor representation. We propose CARL (control-augmented RL), a decision-making framework that interpolates between the two extremes by simultaneously conducting policy learning and model fitting. We prove the convergence of CARL and provide performance guarantees via a finite-sample error bound. On three diverse tasks ranging from linear-quadratic control, portfolio optimization, and epidemic spread management, we observe that CARL can outperform four leading methods for these tasks.

1. Introduction
We study finite-horizon dynamic decision-making in the presence of stochastic factors (a.k.a. factors)—variables that cannot be controlled or affected. Such decision-making problems are central in economics and finance (Dixit & Pindyck, 1994; Duffie, 2001; Merton et al., 1973). Models for factors exist in numerous applications (Fama & French, 2015; Liu et al., 2020), including the Nobel prize winning asset pricing work Fama & French (1992). As an example: stock prices are influenced by factors such as proprietary trading signals and economic indices, which cannot be affected by trading actions from one specific agent.

While reinforcement learning (RL) methods are generally applicable to decision-making with factors, there exist various classical stochastic optimal control models specifically designed for factors, many of which are in the form of stochastic differential equations (SDEs; Cox et al., 1985; Fouque et al. 2011; Hull & White 1990; Vasicek 1977). We aim to augment RL methods with control models and leverage the best of both worlds for a method with flexible factor modeling, good generalization performance, and an easy solving procedure.

However, RL and control are fundamentally different approaches to this problem in terms of both procedures and modeling: RL, like policy gradient descent, conducts end-to-end policy learning in one step, with model-free discrete-time setups (Sutton & Barto, 2018). Optimal control theory provides methods for finding optimal policies to parametric and (often) continuous-time models (Fleming & Rishel, 1975). Their implementation relies on model fitting to specify the parameters on which the policy depends. As a result, when dealing with factors, RL and control works are mostly isolated and seldom borrow strength from each other.

On the one hand, RL methods like policy gradient descent (often being model-free) directly optimize the performance objective for policy learning (Hou et al., 2020; Sutton et al., 1999; Silver et al., 2014). These methods learn a general representation of factors and rely little on domain knowledge. However, since RL does not explicitly model factors, such a method is unstable and tends to overfit in low-signal-to-noise settings. This is especially true in financial applications where the data is notoriously noisy (Aït-Sahalia & Jacod, 2014). On the other hand, control tools like the Hamilton–Jacobi–Bellman (HJB) equation can derive the optimal policy from the assumed SDE models (Fleming & Soner, 2006; Merton, 1992). For decision-making, one conducts model fitting by optimizing the prediction objective. In these methods, the factors and their dynamics are fully specified by domain knowledge without representation learning. As a result, such methods are constrained to low-dimensional, arguably oversimplified situations (Fan & Yao, 2017; Gourieroux & Jasiak, 2018). With the increased availability of finer-scale data where multi-dimensional factors are prevalent, control faces the risk of model misspecification.

¹Equal contribution ¹Department of Computer Science, Princeton University ²Amazon ³Google ⁴Questrom School of Business, Boston University ⁵ORFE, Princeton University.

In submission.
We propose a control-augmented RL (CARL) framework leveraging both policy learning and model fitting as demonstrated in Figure 1. Our solution includes a novel neural stochastic factor model (NSFM) combining classical SDE control models with flexible representation for factors. Further, CARL jointly optimizes the performance objective and the prediction objective for an integration of policy learning and model fitting. Conventionally, the two objectives do not share parameters: the performance objective is a function of policy parameters, while the prediction objective is a function of control model parameters. With policy function forms derived from control theory under NSFMs, we bridge this gap and enable parameter sharing. We prove that such a strategy balances the optimization bias and sample variance for better performance in Section 5. Empirically, we demonstrate the improved performance of CARL in Section 6.

Without specifically modeling factors, there exist other efforts combining policy learning and model fitting, like adaptive control, Bayesian RL, model-based RL, path-integral RL, non-PG-based RL and RL for trading (Appendix A; Aboussalah 2020; Chebotar et al. 2017b; De Prado 2018; Dixon et al. 2020; Fernandez et al. 2020; Ghavamzadeh et al. 2016; Lubars et al. 2021; Mehtab & Sen 2019; Wasser et al. 2015; Yang & Wang 2019). But none of them accomplishes a joint optimization like CARL. Note that CARL does not work for every existing control model for an arbitrary decision-making problem, but is applicable to a wide range of models studied using control in applied fields. We study (Appendix D) LQC (Sun & Yong, 2020), Kim–Omberg model (Kim & Omberg, 1996), Merton model (Merton, 1969), SIR model (Allen, 2017), and EVE model (Avanesyan et al., 2018).

2. Background

In this section, we first formally define the problem of decision-making with factors. Next, we review two common approaches for this problem: policy gradient RL (Section 2.1) and classical SDE control (Section 2.2). Finally, we outline CARL’s roadmap for combining RL and control in Section 2.3.

Decision-Making with Stochastic Factors We consider a finite-horizon dynamic system defined by a tuple \((S, \mathcal{X}, A, T, P, (h(\cdot), r(\cdot)))\), with \(S\) as the controlled state space, \(\mathcal{X}\) the factor space, \(A\) the action space, \(T \in (0, \infty)\) the terminal time, \(P\) the transition distribution, \(h(\cdot) : S \times A \to \mathbb{R}\) the running reward, and \(r(\cdot) : S \to \mathbb{R}\) the terminal reward. At time \(t \in [0, T]\), an agent takes an action \(A_t\) taking values in \(A\) after accounting for the controlled state \(S_t \in S\) and the factors \(X_t \in \mathcal{X}\), which are uncontrollable by \(A_t\). Such an action is chosen by following a time-dependent policy function \(\pi(\cdot) : [0, T] \times S \times \mathcal{X} \to A\). Our goal is to learn an optimal policy \(\pi^*\) that maximizes a performance objective measuring the overall performance.

2.1. Policy Gradient Methods for RL

A policy gradient (PG) method models the dynamics of the decision making problem by a Markov decision process (MDP) with flexible representations for factors. Under the MDP, it conducts end-to-end gradient-based policy learning to derive the optimal policy.

Markov Decision Process PG models the dynamics of \(S_t, X_t,\) and \(A_t\) with an MDP defined by \((S, \mathcal{X}, A, T, P, (h(\cdot), r(\cdot)))\). Importantly, PG does not explicitly model factors but simply treats them as part of the state variables. The policy function is formulated as a neural network with policy parameter \(\theta_{PR}\).

![Figure 1: Demonstration of RL, control and our CARL](image-url)

Figure 1: Demonstration of RL, control and our CARL
from trajectory data repetitively collected using the policy \( \pi(\cdot; \theta_P) \) (Appendix B). The consequence is that \( \theta_P \) can be updated by gradients to maximize \( J \). We summarize such a task as \( \max_{\theta_P} J(\theta_P) \). PG flexibly learns the representation of factors, without specifying a parametric dynamics. However, since the policy is parameterized as a complicated neural network, PG requires collecting many trajectories.

### 2.2. SDE Control

Unlike policy gradient methods, control methods rely on stochastic differential equations (SDEs) to model the dynamics. Optimal policies are then derived directly from the assumed model. To implement these policies in practice, the SDE is fit (calibrated) to trajectory data (a.k.a. model fitting). In what follows, we first provide the necessary background on SDEs and such resulting optimal policies. We then discuss the model fitting procedure.

#### SDEs with Stochastic Factors

SDEs extend ordinary differential equations to dynamical systems influenced by random fluctuations (see Appendix C or Øksendal 2013 for more details). The dynamics can be formulated as

\[
\begin{align*}
    dS_t &= f_1(S_t, X_t, A_t; \beta_1) \, dt + g_1(S_t, X_t, A_t; \beta_2) \, dW_t, \\
    dX_t &= f_2(X_t; \beta_3) \, dt + g_2(X_t; \beta_4) \, dW_t,
\end{align*}
\]

where \( W_t \) is a multi-dimensional independent Wiener process (a.k.a. Brownian motion) that characterizes random fluctuations; \( f_1, f_2, g_1, \) and \( g_2 \) are parametric functions determined by domain knowledge and parameterized by \( SDE \) parameters \( \theta^S := \{ \beta_i \}_{i=1}^4 \). Intuitively, (2) defines the transitions in an infinitesimal time step. The drift coefficients \( f_1(\cdot) \) and \( f_2(\cdot) \) characterize the deterministic part of the variable change, while the diffusion coefficients \( g_1(\cdot) \) and \( g_2(\cdot) \) model the influence of random fluctuations of \( dW_t \). Note that \( X_t \) appears in \( f_1 \) and \( g_1 \), while \( A_t \) is absent in the SDE of \( X_t \). In words, factors \( X_t \) affect the transition of state variables but cannot be affected by actions. With the assumed model, one can leverage control tools such as Hamilton–Jacobi–Bellman (HJB) equation formulations (with more details in Fleming & Rishel (1975)) to solve for the optimal policy.

#### Model Fitting

Implementing solutions found (by e.g. HJB equations) requires fitting a model like (2) to explain the trajectory data collected from the environment, by optimizing the prediction objective. In practice, one can maximize the likelihood according to (2) (Appendix C.3) or use other goodness-of-fit measures (Ait-Sahalia & Kimmel, 2010; Bishwal, 2007; Phillips, 1972). Informally, we denote the prediction objective as \( L \) and summarize the estimation process as \( \max_{\theta_S} L(\theta_S) \), for \( \theta_S \).

Generally speaking, control has good generalization performance by building on parametric models. However, when

#### 2.3. Combine RL and Control

The goal of CARL is to augment RL with control by combining policy learning and model fitting. To this end, we jointly optimize the performance objective and the prediction objective. However, such a combination is challenging since RL and control drastically differ in both modeling setups and learning strategies.

### 3. Neural Stochastic Factor Models

In this section, we introduce the neural stochastic factor model (NSFM), a new model that unifies the modeling flexibility of RL with much of the model structure of SDEs in control-type modeling. We provide details about NSFMs (Section 3.1) and show how NSFMs are instantiated in various decision making problems (Section 3.2).

#### 3.1. Modeling of NSFMs

Similar to classical control models, NSFMs use parametric SDEs to model the dynamics. However, unlike classical control models that use domain knowledge to pick the factors and dynamics, NSFMs leverage the rich features \( Y_t \) available in the data to learn the representation of factors in a fashion similar to RL. Specifically, NSFMs assume that the factors \( X_t \) are not a-priori known but can be represented by features \( Y_t \) of the form \( X_t = \phi(Y_t; \theta^R) \), where \( \phi \) is the representation function formulated by a neural network with the ‘true’ representation parameter \( \theta^R \). We use \( \theta^S \) and \( \theta^R \) as the true parameters and we denote by \( \theta_S \) and \( \theta_R \) their estimated counterparts. Formally, we define an NSFM to satisfy

\[
\begin{align*}
    dS_t &= f_1(S_t, X_t, A_t; \theta^S) \, dt + g_1(S_t, X_t, A_t; \theta^S) \, dW_t, \\
    dX_t &= f_2(X_t; \theta^S) \, dt + g_2(X_t; \theta^S) \, dW_t, \quad X_t = \phi(Y_t; \theta^R).
\end{align*}
\]
We now demonstrate policy learning and model fitting with CARL: Control-Augmented Reinforcement Learning for Decision-Making with Stochastic Factors. We apply NSFM to the Kim–Omberg model, leading to an NSFM-Kim–Omberg model. In detail, consider

$$dS_t = (BS_t + UA_t + X_t) dt + \sum_{j=1}^{2p} D_j A_t dW_t^j, \quad X_t = \phi(Y_t; \theta^R), \quad dX_t = \mu X_t dt + v dW_t,$$

(4)

where $B$, $U$, $\mu$, $v$, and $D_j$ are unknown $p \times p$ matrices. The performance objective is defined as $J(\theta_{PR}):= \mathbb{E}_{\theta_{PR}} \left[ J_T(\theta_{PR}) \right]$, with $Q$, $R$, and $G$ as known matrices with proper dimensions.

### 3.2. Examples of NSFMs

NSFM is widely applicable to many classical control models, some of which are well-known to be highly non-linear and challenging to solve. We detail two examples in this section: Kim–Omberg model from finance and linear quadratic control. More cases are studied in Section 6 and Appendix D.

**NSFM-Kim–Omberg** Kim–Omberg model (Kim & Omberg, 1996) is our major example. It is a standard model for portfolio optimization with predictable asset returns, which has been discussed extensively in the empirical literature (Welch & Goyal, 2008; Muhle-Karbe et al., 2017). We apply NSFM to the Kim–Omberg model, leading to an NSFM-Kim–Omberg model. In detail, consider $p$ assets with prices $Z_t = \{Z_{it}\}_{i=1}^p$ and a risk-free account with, for simplicity, zero interest rate. Factors affecting the asset prices are denoted by a multivariate random variable $X_t$. We denote by $Y_t$ the features available and aim to learn $X_t$ from $Y_t$; by $A_t = \pi(\cdot; \theta_{PR}) \in \mathbb{R}^p$ the action that represents the amount of capital invested in each risky asset; and by $S_t$ the wealth of the investor. With this notation, the SDE parameters can share the common parameter $\theta_{PR}$ to yield a new policy function

$$J(\theta_{PR}) = \mathbb{E}_{\theta_{PR}} \left[ \int_0^T h(S_t, A_t) dt + r(S_T) \right],$$

under the policy function $\pi(\cdot; \theta_{PR})$. Note that the expectation is over the true dynamics with the data generating $\theta^S$ and $\theta^p$. Importantly, in later in Section 4.1, we show that we can parameterize the policy function with the representation parameter $\theta_R$: $\theta_{PR} = (\theta_P, \theta_R)$. For model fitting, the prediction objective of NSFM has a different parameterization compared with ordinary SDE control in Section 2.2. Specifically, the prediction objective is now a function of both SDE parameter $\theta^S$ and the representation parameter $\theta_R$, since the representation function is now part of the model and needs to be learned in model fitting. This yields $L(\theta^S, \theta_R)$ as the prediction objective instead of $L(\theta^S)$. This fact that both the policy $\pi$ and the SDE depend on $\theta_R$ is referred to as parameter sharing. This sharing additionally reduces the complexity of the remaining parameters $\theta_P$, which can be smaller than $\theta_{PR}$ would be in an RL implementation.

### 4. Control-Augmented RL

We present CARL, which combines policy learning in PG-based RL and model fitting in SDE-based control. Specifically, CARL jointly maximizes a weighted average of the performance objective and the prediction objective: $\max(1-\lambda)J + \lambda L$, where $\lambda \in [0, 1]$ is a hyperparameter determining the trade-off between the two tasks. This is possible because we tackle two challenging problems that hinder joint optimization:

- On the one hand, there is no parameter sharing in a naive combination of policy learning and model fitting. We address this issue in Section 4.1. On the other hand, model-fitting of SDEs is a non-trivial task. We resolve this issue in Section 4.2. In Section 4.3 we provide formal definition and the benefits of our bifold learning objective and then detail the entire learning procedure in Section 4.4.

#### 4.1. Policy Learning

Recall that in a PG implementation the performance objective $J(\theta_{PR})$ in (1) depends on $\theta_{PR}$, which does not share parameters with the prediction objective $L(\theta^S, \theta_R)$. By using the NSFM structure of Section 3.1, these two objectives can share the common parameter $\theta_R$, which helps the two tasks to work together. In particular, we reparameterize $\theta_{PR} = (\theta_P, \theta_R)$ to yield a new policy function

$$dS_t = (BS_t + UA_t + X_t) dt + \sum_{j=1}^{2p} D_j A_t dW_t^j, \quad dX_t = \mu X_t dt + v dW_t,$$

(4)
We describe the model fitting of CARL designed to provide a new performance objective $J(\theta_P, \theta_R)$, where $\theta_P$ denotes policy parameters additional to $\theta_R$. This is possible thanks to a connection between the ground truth representation function $\phi(\cdot; \theta_R)$ and the optimal policy $\pi^*$. We state this last fact informally as Theorem 4.1 and develop a rigorous treatment in Appendix E.

**Theorem 4.1 (Informal Statement).** Under suitable technical conditions, there exists a function $F$ such that an optimal feedback policy $\pi^*$ of an NSFM can be written as $\pi^*(t, s, y) = F(t, s, \phi(y; \theta_R^*_R); \theta_P^*)$, where $\theta_P^*$ is an optimal policy parameter and $\theta_R^*_R$ is the true representation parameter.

From Theorem 4.1, we can rewrite a candidate policy function as $\pi(t, s, y; \theta_P, \theta_R) = F(t, s, \phi(y; \theta_R); \theta_P)$. This in turn yields the desired sharing in $J(\theta_P, \theta_R)$ through $\phi$. In addition, compared to model-free RL that formulates the policy function without constraints, Theorem 4.1 results in a much smaller candidate policy space, which substantially reduces sample complexity and simplifies computation. Theorem 4.1 is widely applicable to various NSFM s. The function $F$ in Theorem 4.1 is distinct in each problem and can often be analytically derived. As an example, we detail $F$ of NSFM-Kim–Omberg in Lemma 4.2, and the results for many other models are presented in Appendix D.

**Lemma 4.2.** In the NSFM-Kim–Omberg model defined in (3) with $p$ risky assets, given wealth $s$, stock price $z$, and feature $y$, the optimal policy at time $t$ is given by $\pi^*(t, s, z, y) = F(t, s, z, \phi(y; \theta_R^*_R); s)$ where $F(t, s, z, \phi(y; \theta_R^*_R); s)$ is a time-dependent function with an optimal policy parameter $\theta_R^*_R$.

The form of $F$ in Lemma 4.2 provides a structure that eases the policy learning in our framework. Importantly, the optimal policy can be factored into a time-dependent part $k(t)$, factor representation part $\phi(y; \theta_R)$, and the wealth $s$. This structure comes from analyzing the stochastic optimal control problem and is the quality of making SDEs central to the NSFM model.

While we can further explicitly specify the function form of $k(t)$, in practice, we could use a simple neural network to parameterize $k(t)$ (Appendix F.2). In general, we therefore do not need to derive an explicit closed form for the optimal policy. Instead, CARL can be applied and provides benefits as long as there is at least some knowledge about how $F$ relates to $\phi(y; \theta_R)$.

### 4.2. Model Fitting

We describe the model fitting of CARL designed to provide an easy-to-calculate prediction objective. An example under NSFM-Kim–Omberg is provided. One natural choice for model fitting is to use the likelihood as the prediction objective $L$. However, the derivation of the likelihood function of SDEs involves solving stochastic integrals, and thus is extremely challenging to calculate in the multi-dimensional case (see Appendix C.3). Instead, in CARL, it turns out that not all SDE parameters are important for deriving an well-performing policy. Consequently, we are able to conduct a simplified model fitting with a square-loss prediction objective, informally summarized in Theorem 4.3 (see Appendix E for a formal treatment).

**Theorem 4.3 (Informal Statement).** Under suitable technical conditions, we define $\Delta S_t := S_{t+\Delta t} - S_t$. Then, the true representation parameter $\theta_R^*$ satisfies $\theta_R^* \in \arg \max_{\theta_R} L(\theta_R)$, with

$$L(\theta_R) := -\min_{\theta_S} \| \Delta S_t - f_1(S_t, \phi(Y_t; \theta_R), A_t; \theta_S) \Delta t \|^2_2,$$

when $\Delta t > 0$ is small enough.

We discuss three implications of Theorem 4.3. First, the true representation parameter $\theta_R^*$ maximizes the proposed negative square loss. Therefore, we can maximize the empirical version negative square loss. Second, by (5), we can now focus on $L(\theta_R)$ instead of $L(\theta_S, \theta_R)$. Thanks to the often simple form of $f_1$ in Theorem 4.3, the minimization over $\theta_S$ in (5) can often be analytically solved given $\theta_R$. Finally, it is simple to optimize $L(\theta_R)$ as it is dependent on only $f_1$ in the NSFM. We venture NSFM-Kim–Omberg as an example and detail its $L(\theta_R)$ in Lemma 4.4.

**Lemma 4.4.** For NSFM-Kim–Omberg defined in (3), $L(\theta_R)$ is derived as:

$$L(\theta_R) := -\min_{\theta_S} \| \Delta \log(Z_t) - \phi(Y_t; \theta_R) \Delta t - \theta_S \|^2_2,$$

where $\Delta \log(Z_t) := \log(Z_{t+\Delta t}) - \log(Z_t)$ and $\theta_S$ is a $p \times 1$ parameter vector. Given any $\theta_R$, we can further derive

$$\arg \min_{\theta_S} \| \Delta \log(Z_t) - \phi(Y_t; \theta_R) \Delta t - \theta_S \|^2_2 = \mathbb{E} \left[ \phi(Y_t; \theta_R) \Delta t - \Delta \log(Z_t) \right].$$

According to Lemma 4.4, we can use the correlation between stock prices and factors to aid portfolio optimization. Nonetheless, it should be noticed that stock price prediction is an extremely challenging problem and is not a focus of this work. Instead, CARL seeks to take advantage of such correlations to an extent to derive a performant policy.

### 4.3. Bifold Learning Objective

To jointly implement policy learning and model fitting, we propose a bifold learning objective combining $J$ in Section 4.1 and $L$ in Section 4.2. Specifically, CARL solves the
following optimization \( \max_{\theta_P, \theta_R} H(\theta_P, \theta_R) \), where

\[
H(\theta_P, \theta_R) := (1 - \lambda) J(\theta_P, \theta_R) + \lambda L(\theta_R)
\]

\[
\begin{aligned}
\text{Performance} & := (1 - \lambda) J(\theta_P, \theta_R) \\
\text{Prediction} & := -\lambda \min_{\theta_S} \mathbb{E} \| \Delta S_t - f_t(S_t, \phi(Y_t; \theta_R, \mathbf{A}_t; \theta_S)) \|^2_2.
\end{aligned}
\]

Therefore, \( \theta_R \) is the shared parameter for performance and prediction: the learning of representation function benefits from both policy learning and model fitting.

The proposed bifold learning objective is designed to balance policy learning and model fitting. Ideally, with infinite data and perfect global optimization, optimizing each of the two terms is equivalent: the \( \theta_R \) providing the best prediction also leads to the optimal performance. In practice though, the choice of which of the two terms to optimize yields different solutions depending on the context, cf. (Bengio, 1997a;b; Kaastra & Boyd, 1996). On the one hand, only maximizing \( L \) provides little guarantees on the global optimality on the performance objective \( J \): we need to include \( J \) in the objective; one the other hand, the added \( L \) works as a regularization to \( J \) and leverages more information from the assumed model to improve sample complexity: it is beneficial to include \( L \) in the objective. This intuition is backed by rigorous theoretical guarantees in Section 5 and thorough empirical validation in Section 6.

4.4. Algorithm

We optimize \( H \) in (6) by a PG framework. The algorithm is summarized in Algorithm 1. Specifically, following Theorem 4.1, we first parameterize candidate policies as \( \pi(t, s, y) := F(t, s, \phi(y; \theta_R); \theta_P) \) (line 3 of Algorithm 1). Then, in each iteration, we implement the candidate policy with a time interval \( \Delta t \) and \( \mathbf{A}_t = \pi + \delta \varepsilon \). Interacting with the environment, we collect \( B \) trajectories including the observed states, features, and actions. We denote such a dataset by \( \mathbb{X} \) (Line 5 of Algorithm 1) and with this data we can estimate the gradient of \( H \) according to Lemma 2.1 (Line 6-8 of Algorithm 1).

5. Theoretical Analysis

We provide theoretical guarantees for CARL. First, since CARL seeks to optimize a highly non-convex objective (6), we show that CARL can indeed converge to a stationary point of the objective (Section 5.1). Second, with additional assumptions, we provide global properties of CARL’s performance (Section 5.2), which show the benefits of jointly conducting policy learning and model fitting. To simplify notation in this section, denote by \( \theta \) the tuple \( (\theta_P, \theta_R, \theta_S) \).

Algorithm 1 Control-Augmented RL (CARL)

1: Input: Hyperparameter \( \lambda \), learning rate \( \eta \), batch size \( B \), number of iterations \( N \), and the environment.
2: Output: \( \pi(\cdot; \theta_P, \theta_R) \).
3: Initialize a neural network \( \phi(\cdot; \theta_R) \) and accordingly the policy function \( \pi(\cdot; \theta_P, \theta_R) \) by Theorem 4.1.
4: for \( i \in [N] \) do
5: Generate a dataset \( \mathbb{X} \) of \( B \) trajectories using \( \pi(\cdot; \theta_P, \theta_R) \).
6: Estimate the gradient of \( J \) by Lemma 2.1 and Theorem 4.1.
7: Estimate the gradient of \( L \) according Theorem 4.3.
8: Combine two gradients with \( \lambda \) following (6).
9: Update \( \theta_P \) and \( \theta_R \) with learning rate \( \eta \).
10: end for
11: Return \( \pi(\cdot; \theta_P, \theta_R) \)

5.1. Convergence to Stationary Points

We begin by proving that Algorithm 1 converges to a stationary point of \( H \) where the gradient is zero. Such stationary points have benign local properties and this convergence is of great interests in the theoretical analysis of PG methods (Agarwal et al., 2021; Bhandari & Russo, 2019; Karimi et al., 2019; Papini et al., 2018; Wang et al., 2019; Xu et al., 2020).

Theorem 5.1. Let \( \theta_i \) denote the parameter after the \( i \)th iteration. Under common technical conditions (Appendix H.1) and with learning rate \( \eta = O\left(\frac{1}{\sqrt{N}}\right) \),

\[
\min_{i \in [N]} \mathbb{E} \left[ \| \nabla H(\theta_i) \|_2^2 \right] = O\left(\frac{1}{\sqrt{N}} + \frac{1}{B}\right)
\]

where \( N \) is the total number of iterations, \( B \) is the batch size, and \( \nabla H(\theta_i) \) is the gradient of \( H \) at \( \theta_i \).

From Theorem 5.1, the expected norm of the gradient of \( H \) at \( \theta_i \) converges to zero as \( N \) and \( B \) increase, suggesting that Algorithm 1 converges to a stationary point of \( H \).

5.2. Global Properties on Performance

We next provide a finite-sample upper bound between CARL’s performance and the global optimal performance. Importantly, we show that only carrying out policy learning \( (\lambda = 0) \) or model fitting \( (\lambda = 1) \) will lead to a suboptimal performance bound, thus demonstrating the benefits of CARL’s joint optimization procedure. Below we discuss the assumptions, present our theoretical results, and provide interpretations.

Assumptions Due to the nonconvex nature of the deep PG framework used in CARL, global theoretical analysis inevitably requires additional assumptions (Polyak, 1963;
Bhandari & Russo, 2019; Jin et al., 2019; Ma, 2020). Similarly, we follow a set of standard assumptions (Wang et al., 2019) and consider a projected version of Algorithm 1 (Appendix I). In addition, we consider a locally strongly concave parameter region of $H(\theta)$ with stationary point $\tilde{\theta}$. Similar local curvature assumptions are commonly used to analyze non-convex problems (Bach et al., 2017; Loh, 2017).

**Theorem 5.2.** Let $\bar{\theta}_i$ be the estimation after the $i$th iteration, $\bar{\theta} = \frac{1}{N} \sum_{i=1}^{N} \theta_i$ the average estimation, and $\tilde{\theta}$ a stationary point of $H$. With standard assumptions (Appendix I.1) and $\eta$ as the learning rate, there exists a constant $C > 0$, such that:

$$
\mathbb{E} \left[ (J(\theta^*) - J(\bar{\theta}))^2 \right] 
\leq \frac{\eta(1 - \lambda)^2 C}{B} + \frac{C \eta \lambda^2 (1 + \log(N))}{BN} 
+ \frac{(H(\theta^*) - H(\tilde{\theta}))^2}{(1 - \lambda)^2} + C(1 + \eta) \frac{1}{\eta N}.
$$

Here (I) bounds the error caused by the variance of the gradient estimation in Algorithm 1; (II) corresponds to the error caused by convergence to a local optimum $\tilde{\theta}$ instead of the global optimum $\theta^*$.

Theorem 5.2 shows the value of joint optimization over the two objectives: there exists a $\lambda \in (0, 1)$ providing a smaller upper bounds than $\lambda = 0$ or 1 (Lemma I.8). In detail, when $\lambda = 0$, one can show that the variance bound of the gradient estimation (I) is maximized. Only maximizing $J$ leads to bigger variance bound and is sample-inefficient. Secondly, when $\lambda = 1$, the error bound (II) associated with selecting a locally optimal point goes to infinity. So, we cannot effectively bound the final performance $J(\theta)$ by only working with $L(\theta)$. This is not surprising since optimizing over $L(\theta)$ may lead to a local optimum which, without further assumptions, has no guarantees regarding $J(\theta)$. Finally, there exists a $\lambda \in (0, 1)$ minimizing the upper bound, which may provide Further, this upper bound implies faster convergence in the order of $N$, with further assumptions on the relationship among $B$ and $N$ and $H(\theta^*) - H(\tilde{\theta})$. In our experiments (Section 6.1), $\lambda$ is tuned as a hyperparameter. We observe this fact empirically: that $\lambda \in (0, 1)$ provides better performance.

**6. Experiments**

We demonstrate the performance of CARL on a variety of applications ranging from LQC, to portfolio optimization, and to epidemic spread management. We compare CARL against up to four alternative methods—from baselines to state-of-the-arts—that combine RL and control in various manners.

**6.1. Synthetic Experiments**

We show that CARL can deliver superior empirical performance for decision-making with factors. To compare the methods, we average the achieved performance objective values ($J$) over different trajectories as the metric.

**Protocol** We simulate environments from four models and implement CARL and other methods (described below) to compare their performance. The four models (Section 3.2 and Appendix D) are: NSFM-LQC, NSFM-Kim–Omberg, Merton, and NSFM-EVE. In particular, NSFM-EVE is an instance of the stochastic volatility problems, which are notoriously difficult to solve. We simulate environments with dimension 10, 20, and 30. For each method, we vary the learning rate and other method-specific hyperparameters with early stopping. With each method-hyperparameter-environment combination, we repeat training and testing ten times (see Appendix J).

**Methods** We compare six methods that differ in terms of how they combine RL and control (Table 1): (i) **PG**: policy gradient descent is the state-of-the-art model-free RL method for problems with continuous states and actions; (ii) **Control**: this represents a fully control-based method, which does not conduct representation learning for factors; (iii) **E2C**: exiting methods (Lee et al., 2019; Watter et al., 2015) in the RL literature combine RL and control in an iterative fashion instead of the joint optimization manner like CARL; Embed to Control (E2C) is a representative since it best fits NSFM-LQC; (iv) **SDE-RL**: a state-of-the-art method in the control literature (Bu et al., 2020; Wang & Zhou, 2020) on combining RL and control on the consid-

---

**Table 1: Different methods with their characteristics**

<table>
<thead>
<tr>
<th>Methods</th>
<th>RL</th>
<th>Control</th>
<th>Joint Opt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Control</td>
<td>✖</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>E2C</td>
<td>✔</td>
<td>✔</td>
<td>✖</td>
</tr>
<tr>
<td>SDE-PG</td>
<td>✖</td>
<td>✔</td>
<td>✖</td>
</tr>
<tr>
<td>Oracle</td>
<td>✖</td>
<td>✖</td>
<td>✔</td>
</tr>
<tr>
<td>CARL</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Table 2: Average performance objective values under selected hyperparameters for NSFM-LQC**

<table>
<thead>
<tr>
<th># Variables</th>
<th>B</th>
<th>CARL (Ours)</th>
<th>E2C</th>
<th>SDE-PG</th>
<th>PG</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>50</td>
<td>5028</td>
<td>4980</td>
<td>5019</td>
<td>4409</td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>4947.705</td>
<td>3752</td>
<td>−3 × 10^5</td>
<td>−1 × 10^6</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>−1187.874</td>
<td>−8 × 10^5</td>
<td>−3 × 10^5</td>
<td>−4 × 10^5</td>
</tr>
</tbody>
</table>

---

**Figure 3: Total rewards for all configurations in the 20-variable NSFM-LQC**
### Table 3: Average performance after tuning for NSFM-Kim–Omberg, Merton, and NSFM-EVE

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average Performance</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NSFM-Kim–Omberg</td>
<td>Merton</td>
<td>NSFM-EVE</td>
<td></td>
</tr>
<tr>
<td>CARL (Ours)</td>
<td>23.708</td>
<td>21.998</td>
<td>23.597</td>
<td></td>
</tr>
<tr>
<td>PG</td>
<td>22.286</td>
<td>21.013</td>
<td>22.257</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>23.441</td>
<td>23.468</td>
<td>23.548</td>
<td></td>
</tr>
<tr>
<td>SDE-PG</td>
<td>19.655</td>
<td>23.425</td>
<td>23.583</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4: Average utility after tuning for real-world stock trading

<table>
<thead>
<tr>
<th>Sectors</th>
<th>Average Utility</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CARL (Ours)</td>
<td>23.757</td>
<td>21.315</td>
<td>10.474</td>
<td>18.441</td>
</tr>
<tr>
<td>PG</td>
<td>23.609</td>
<td>13.928</td>
<td>23.512</td>
<td>12.813</td>
</tr>
<tr>
<td>Control</td>
<td>23.456</td>
<td>18.1905</td>
<td>13.091</td>
<td>18.201</td>
</tr>
</tbody>
</table>

Figure 4: Average cost and number of infected under the selected hyperparameters

6.3. Robust Analysis via Epidemic Spread Management

We conduct robust analysis for CARL (see Appendix L). We show that even with a wrongly specified model, CARL can provide decent performance. Specifically, we study the susceptible-infected-recovered (SIR) model with factors for epidemic spread management ((Gray et al., 2011; Allen, 2017; Mahrouf et al., 2021)). In the model, the numbers of susceptible, infected, and recovered individuals are the state variables that are all assumed to follow a non-LQC SDE with factors. We simulate the environment following NSFM-SIR but then implement CARL with NSFM-LQC, which is a wrongly specified model for this problem. While we examine synthetic data, our study has significant practical implications, since approximating complicated dynamics with LQC is a common technique. By considering vaccination and treatment as actions, we aim to minimize a cost function. We compare CARL with PG and No Action (as not taking any vaccination or treatment actions). The achieved average cost and the numbers of infected individuals are report in Figure 4. We see that CARL achieves almost the same (only slightly higher) cost compared with PG. These phenomena suggest that CARL is applicable to this synthetic SIR problem and can be robust against model misspecification.

7. Conclusion

We propose CARL, a novel framework for finite-horizon decision-making under SDEs with stochastic factors. We demonstrate CARL’s characteristics both theoretically and empirically.
References


Appendix

A. Related Methods

In this section, we discuss related methods in both stochastic optimal control, reinforcement learning, and empirical risk minimization.

A.1. Stochastic Optimal Control

Existing strategies for solving decision-making problems using stochastic optimal control (Fleming & Rishel, 1975; Fleming & Mitter, 1982; Fleming & Soner, 2006; Yong & Zhou, 1999) can be loosely summarized as performing three steps:

1. Choosing the model for the dynamics (the type of ODE, SDE, MDP, etc).
2. Estimating the parameter of the selected model (which is also referred to as model fitting, model identification, or calibration).
3. Solving for the optimal policy under the estimated model.

Formally speaking, stochastic optimal control denotes refers to methods for the third step, where both the model and parameters are already specified.

Finding and estimating an appropriate model for stochastic optimal control requires significant domain knowledge. For instance, in finance, the modeler must specify both which feature are relevant and how they affect stock prices (Fama & French, 1992). If not every relevant factor is correctly specified, optimal control can hardly lead to good performances. As a result, in stock trading, control methods would hand pick three to five economic indices as the factors and assume they follow a simple (often linear) SDE. But indeed trading can benefit from much richer datasets including related option prices, technical indicators, and interest rates (Aboussalah, 2020; De Prado, 2018; Dixon et al., 2020; Mehtab & Sen, 2019).

Further, even with a correctly specified model and factors, likelihood-based estimation for SDE control models can be very challenging (Phillips & Yu, 2009) as demonstrated in Section C. As a result, methods like A¨ıt-Sahalia (2008); Ait-Sahalia & Kimmel (2010) seek to replace the exact likelihood with other likelihood-based objective functions, while maintaining theoretical guarantees. However, the proposed objective function needs to be derived for each specific problem, and the derivation can be challenging. Other methods like Fasen (2013); Holý & Tomanová (2018) rely on more specific parametric or low-dimensional setups. To alleviate these issues, our framework extends the existing SDE-based stochastic optimal control models by allowing for a flexible and generalized definition of stochastic factor dynamics. Further, we simultaneously conduct policy learning and model fitting in an RL manner, with a square-loss objective that avoids the calculation of an exact likelihood.

A.2. Reinforcement Learning

Unlike stochastic optimal control methods, RL aims to conduct the aforementioned three steps by (i) relying more on data (ii) in an end-to-end fashion. Methods like model-free RL assume no parametric forms on the dynamics, and directly learn the optimal policy while explicitly learning the model (step 1) and estimating the parameters (step 2).

Non-Policy-Gradient RL There exists many flavours of RL algorithms other than PG based methods, like Q iteration methods and policy iteration methods (Sutton & Barto, 2018). However, the problems considered in this manuscript are with continuous state and actions. In these cases, both Q iteration and policy iteration need additional, more complicated techniques to handle. In contrast, PG can naturally handle continuous states and actions with simple procedures. Therefore, we build CARL on a PG framework in this paper while we note that other RL methods may be also suitable.

Continuous-Time Model-Free RL Continuous-time model-free RL (Wang et al., 2018a; Doya, 2000; Munos, 2006) also leverages continuous-time models, which is similar to CARL. While often built on SDEs, continuous-time model-free RL does not leverage or assume any SDE structure, and thus struggles with the common open questions in model-free RL like poor stability and sample complexity.
Path-Integral RL  Path integral methods stem from the theoretical result that the value function of a type of continuous-time decision-making problems can be expressed in closed form as a Feynman-Kac path integral (Fleming & Rishel, 1975; Kappen, 2005). A series of control/RL methods follow the rationale of optimizing the policy to maximize such an integral. Specifically, Theodorou et al. (2010) propose an open-loop control strategy; Kappen & Ruiz (2016) build RL with importance sampling; Chebotar et al. (2017a;b); Stulp & Sigaud (2012) combine path integral with other model-based or model-free RL methods. However, the core derivation only holds for decision-making satisfying (see Equation (1) in Kappen & Ruiz (2016)):

$$dS_t = f_t(S_t, A_t) \, dt + g(S_t) \, dW_t,$$

where action $A_t$ can not affect the randomness of the state transition. Such an assumption is limiting, and does not hold for either portfolio optimization nor linear quadratic control, both studied in this manuscript. For portfolio optimization, how to allocate the wealth in order to minimize the risk is key to a successful policy.

Model-based RL  RL under specified dynamics is an active research area. Methods like Bradtke (1993); Basei et al. (2020); Fazel et al. (2018); Tu & Recht (2018); Umenberger et al. (2019) focus on the linear-quadratic setting without factors and develop RL methods with improved empirical and theoretical results. Fernandez et al. (2020) propose an RL method with a linear-quadratic region. Yang & Wang (2019) study decision-making under dynamics with linear transitions. These methods consider discrete-time models and do not study factors. We empirically study one representative (SDE-RL) of these methods in Section 6.

Representation learning and Partial Observability in RL  Our proposed procedure is related to representation learning and partial observability in RL, if we consider $X_t$ as the hidden variables or the representation in these works. Some existing works use auxiliary supervision to learn the representation of state variables (Dadashi et al., 2019; Gelada et al., 2019; Lange & Riedmiller, 2010). These methods do not specify a model and thus can not benefit from the SDE information. Another line of work develops RL by learning a partially observed Markov decision process. The learned model is either used to generate data for RL or to analytically derive the optimal policy (Chebotar et al., 2017a; Hafner et al., 2019; Kim et al., 2019; Watter et al., 2015). From a modeling perspective, these methods are not applicable to SDEs with factors. Methodologically, these methods learn the representation/dynamics by only fitting the model through likelihood maximization, without a joint optimization in CARL.

Bayesian RL  Our proposed framework is also related to Bayesian models (Ghavamzadeh et al., 2016; Rawlik et al., 2012), if we treat the learned representation of factors as the hidden variable. Strictly formulating an NSFM as a Bayesian model requires assumptions specifying the conditional distributions, and thus requires more domain knowledge. The optimization of Bayesian methods is also more challenging.

RL for Stock Trading  Various of efforts have been made on applying continuous-time or discrete-time model-free RL to stock trading (Corazza & Bertoluzzo, 2014; Hambly et al., 2021; Nechchi, 2016; Nan et al., 2020; Xiong et al., 2018). However, these methods focus more on the feature selection or framework construction: methodologically, they do not take benefit from control models. In Section 6.2, we pick features according to their results, and select PG as one representative for such methods.

A.3. Empirical Risk Minimization

Another related area is Empirical Risk Minimization (ERM) (Vapnik, 1992). ERM studies the minimization of an objective function using the averages over training data to construct an empirical loss function. Recent work connected ERM with simulation-based and data-based offline decision-making methods (Reppen & Soner, 2020). More specifically, when the random input is observable and unaffected by actions, and a training set is available, the decision-making problem can be formulated as an ERM problem. As a result, while CARL is applicable to decision-making in general, for portfolio optimization the procedure may be reformulated as an ERM extension.

B. Policy Gradient RL

In this section, we provide extended results on PG in Section 2.1. Specifically, we first prove Lemma 2.1, and then provide implementation details of PG.
B.1. Proof of Lemma 2.1

To start with, we review the policy gradient theorem in Sutton & Barto (2018):

**Lemma B.1** (Stochastic Policy Gradient Theorem). Let $\tau := \{t_i, s_{t_i}, x_{t_i}, a_{t_i} \mid i \in [m]\}$ be a trajectory with $m$ observations at time points $t_1, t_2, \cdots, t_m$. Under mild differentiable conditions,

$$
\nabla_{\theta} J(\theta_{PR}) \propto \mathbb{E}_{\theta} \left[ J(\tau; \theta_{PR}) \sum_{i=1}^{m} \frac{\partial \log P(A_{t_i} = a_{t_i} \mid S_{t_i} = s_{t_i}; \theta_{PR})}{\partial \theta_{PR}} \right],
$$

where $J(\tau; \theta_{PR})$ denotes one sample value of the performance objective under parameter $\theta_{PR}$.

According to the definition of policy in Section 2.1, the conditional distribution Then, Lemma 2.1 is proved by taking the likelihood of Gaussian distribution of $A_{t_i} = a_{t_i} | S_{t_i} = s_{t_i}$ is Gaussian. Therefore, by taking the likelihood of $A_{t_i} = a_{t_i} | S_{t_i} = s_{t_i}$ into Lemma B.1, we finish the proof.

B.2. PG Algorithm

With Lemma 2.1, we now review the policy gradient algorithm. In practice, one can collect trajectory data to estimate the gradient $\nabla_{\theta_{PR}} J(\theta_{PR})$ by $\nabla_{\theta_{PR}} J(\theta_{PR}) := \mathbb{E}_{\theta_{PR}} \left[ J(\tau; \theta_{PR}) \sum_{i=1}^{m} \frac{\partial \|A_{t_i} - \pi(\cdot; \theta_{PR})\|^2}{\partial \theta_{PR}} \right]$, with $\mathbb{E}_{\theta_{PR}}$ denoting the sample average on the path dataset collected under policy $\pi(\cdot; \theta_{PR})$. Note that the gradient updates maximizing $\mathbb{E}[J(\tau; \theta)]$ is equivalent to minimizing

$$
\mathbb{E}_{\theta_{PR}} \left[ R(\tau) \sum_{i=1}^{m} \|a_{t_i} - \pi(\cdot; \theta_{PR})\|^2 \right],
$$

which corresponds to solving a weighted least square problem.

C. A Primer on SDEs

In this section, we provide necessary background knowledge on SDEs and stochastic calculus.

C.1. Formulation

Stochastic differential equations are a generalization of ordinary differential equations to dynamic systems influenced by random fluctuations. The structure of the randomness can in principle be quite general, such as with jump processes where the state evolution is no longer continuous (Tankov, 2003). Although our method can be generalized to all SDEs, we restrict ourselves to practical settings where the source of randomness is a Brownian motion.

Let $W_t$ be a multi-dimensional independent standard Brownian motion. For a random process $S_t$, an SDE is typically expressed using a differential form as

$$
dS_t = f(S_t) \, dt + g(S_t) \, dW_t, \text{ or } S_t = S_0 + \int_0^t f(S_s) \, dt + \int_0^t g(S_s) \, dW_s,
$$

(7)

where $f(\cdot)$ and $g(\cdot)$ are functions of $S_t$. The stochastic integral $\int_0^t g(S_s) \, dW_t$ is the accumulation of influence to the state due to the noise. We refer the reader to (Karatzas & Shreve, 1987) for details on the construction of stochastic integrals and SDE theory. Important here is that Equation (7) defines the transition of $S_t$ in an infinitesimal time step. The drift coefficient $f(S_t)$ characterizes the deterministic part of the change of $S_t$, and the diffusion coefficient $g(S_t)$ models the randomness in the transition of $S_t$.

C.2. Examples

As concrete examples, we discuss two families of SDEs widely used in finance, economics, and biology: (Merton et al., 1973; Vasicek, 1977; Bartoszek et al., 2017; Blomberg et al., 2020; Rohlf et al., 2014) Geometric Brownian motion (GBM) and Ornstein–Uhlenbeck (OU) processes. The OU structure appears in both applications below, and the financial application
CARL: Control-Augmented Reinforcement Learning for Decision-Making with Stochastic Factors

uses GBM as a base, but extends it with OU drift coefficients. The two types of SDEs are given by

\[
\text{GBM: } \frac{dS_t}{S_t} = \mu \, dt + \sigma \, dW_t, \quad \text{OU: } dS_t = \mu S_t \, dt + \sigma \, dW_t,
\]

where \( \frac{dS_t}{S_t} := \left\{ \frac{dS_{t+1}}{S_{t+1}} \right\} \) denotes the component-wise division of \( S_t \), and the matrices \( \mu \) and \( \sigma \) define the drift and diffusion coefficients.

We refer the interested reader to \((\text{Fleming & Soner, 2006})\) for more information on these topics. We now briefly formulate two classic stochastic optimal control models for decision-making with stochastic factors. The stochastic factors appear as the drift coefficients of other state variables and are themselves modeled as SDEs.

C.3. Likelihood of SDEs

The likelihood of SDEs is derived in a sequential manner, and can be very hard to calculate. As an example, for an SDE formulated as

\[
dS_t = f(S_t; \theta) \, dt + g(S_t; \theta) \, dW_t,
\]

given a dataset \( \mathbb{X} := \{ t_i, s_{t_i} \mid i \in [m] \} \), the likelihood is derived as

\[
P(\mathbb{X}; \theta) \propto m^{-1} \sum_{i=1}^{m-1} \frac{dF_{S_{t_{i+1}} \mid S_{t_i} = s_{t_i}; \theta}}{ds} \bigg|_{s = s_{t_{i+1}}},
\]

where \( F_{S_{t_{i+1}} \mid S_{t_i} = s_{t_i}; \theta} \) denotes the cumulative distribution function of the conditional random variable \( S_{t_{i+1}} \mid S_{t_i} = s_{t_i}; \theta \). According to the SDE (8), the conditional random variable can be derived as

\[
(S_{t_{i+1}} \mid S_{t_i} = s_{t_i}; \theta) = s_{t_i} + \int_{t_i}^{t_{i+1}} dS_t = s_{t_i} + \int_{t_i}^{t_{i+1}} f(S_t; \theta) \, dt + \int_{t_i}^{t_{i+1}} g(S_t; \theta) \, dW_t,
\]

whose CDF can be very hard to derive or calculate.

C.4. Itô’s Formula

Itô’s Formula is a fundamental analytical tool for SDEs, and crucial for their analysis. We only provide a simple version here, which is sufficient for our analysis. A more general version and the rigorous proof and assumptions can be found in \((\text{Karatzas & Shreve, 1987, Theorem 3.3})\).

**Lemma C.1** (Itô’s Formula). Consider a twice differentiable function \( G \), and \( S_t \) following

\[
dS_t = f(S_t) \, dt + g(S_t) \, dW_t.
\]

Then, we have

\[
dG(t, S_t) = \left\{ \frac{\partial G}{\partial t} + \left( \frac{\partial G}{\partial S_t} \right)^\top f(S_t) + \frac{1}{2} \text{Tr} \left[ g(S_t)^\top \frac{\partial^2 G}{\partial S^2_t} g(S_t) \right] \right\} dt + \left( \frac{\partial G}{\partial S_t} \right)^\top g(S_t) \, dW_t.
\]

C.5. Properties of Brownian Itô’s Integrals

**Lemma C.2.** For a suitable bounded process \( S_t \), the Itô integral \( \int_0^t S_t \, dW_t \) satisfies:

\[
\mathbb{E} \left[ \int_0^t S_t \, dW_t \right] = 0, \quad \mathbb{E} \left[ \left( \int_0^t S_t \, dW_t \right)^2 \right] = \mathbb{E} \left[ \int_0^t S_t^2 \, dt \right].
\]

The rigorous constructions and derivations for the Itô integral can be found in \((\text{Karatzas & Shreve, 1987})\).
D. Applications of CARL

CARL is applicable to many classical control models, including some highly nonlinear and difficult ones. In this section, we discuss NSFM-LQC (Appendix D.1), the Merton model (Appendix D.2), NSFM-Kim–Omberg (Appendix D.3), NSFM-EVE (Appendix D.4), and NSFM-SIR (Appendix D.5).

D.1. NSFM-LQC

D.1.1. Modeling

In linear-quadratic control (LQC) (Sun & Yong, 2020), the multivariate state variable $S_t$ follows an Ornstein–Uhlenbeck process:

$$dS_t = (BS_t + UA_t) dt + \sum_{j=1}^{p} D_j A_t dW^j_t,$$

where $B$, $U$, $\mu$, $v$, and $D_j$ are unknown matrices with appropriate dimensions. With $A_t = \pi(\cdot)$ following the policy $\pi$ we aim to solve

$$\max \pi \mathbb{E}_\pi \left[ \int_0^T \left( (QS_t)^\top S_t + (RA_t)^\top A_t \right) dt + (GS_T)^\top S_T \right],$$

with $Q$, $R$, and $G$ as known matrices with appropriate dimensions, and $T$ is terminal time. Further, apply the modeling strategy in Section 3.1, we derive NSFM-LQC:

$$dS_t = (BS_t + UA_t + X_t) dt + \sum_{j=1}^{p} D_j A_t dW^j_t,$$

$$X_t = \phi(Y_t; \theta^*_R), \quad dX_t = \mu X_t dt + v dW_t,$$

where $X_t$ being the stochastic factor affecting the drift of state variable $S_t$.

D.1.2. Policy Learning and Model Fitting

In this section, we detail the policy learning and model fitting for NSFM-LQC.

First, we detail the policy learning of NSFM-LQC. Specifically, we derive $F$ in Theorem 4.1 of NSFM-LQC in Lemma D.1.

**Lemma D.1.** For the NFSM-LQC defined above, at time point $t$ given state variable $x$, and feature $y$, the optimal policy satisfies $\pi^*(t, s, z, y) = F(t, s, z, y; \theta^*_R; \theta^*_P)$ with

$$F(t, s, z, y; \theta^*_R; \theta^*_P) = k_1(t; \theta^*_P)s + k_2(t; \theta^*_P)\phi(y; \theta^*_R),$$

where $k_1(\cdot; \theta^*_P) : [0, T] \to \mathbb{R}^{p \times p}$ and $k_2(\cdot; \theta^*_P) : [0, T] \to \mathbb{R}^{p \times p}$ are two time-dependent functions.

Then, following Theorem 4.3, we derive the prediction objective $L$ for NSFM-LQC in Lemma D.2.

**Lemma D.2.** For NSFM-LQC defined above, $L(\theta_R)$ is derived as:

$$L(\theta_R) := \min_{\theta_S = (C_1, C_2)} \mathbb{E} \left[ \|S_t + \Delta_t - C_1 S_t - C_2 A_t\|^2_2 | S_t, A_t \right],$$

where $\Delta S_t := S_{t+\Delta t} - S_t$ and $\theta_S$ is a $p \times 1$ parameter vector.

The proof is provided in Appendix F.1.

D.2. Merton Model and Portfolio Optimization

Merton model (Merton, 1969) is one of the most classic setup for portfolio optimization. It studies the allocation of capital across a set of financial assets in order to maximize the profits and minimize the risks.
D.2.1. Modeling

We now provide the modeling details of the Merton model, and also the common setup for portfolio optimization. Originally, Merton model, does not have factors. Consider $p$ assets with prices $Z_t = \{Z_{it}\}_{i=1}^p$ and a risk-free money market account with, for simplicity, zero interest rate of return like cash. The dynamics for asset prices is formulated as

$$\frac{dZ_t}{Z_t} = \mu dt + \sigma dW_t,$$

with $\frac{dZ_t}{Z_t} := \left\{ \frac{dZ_{it}}{Z_{it}} \right\}$

The parameters $\mu, \sigma$ are $p \times p$ matrices, with $\sigma$ denoting the volatility of assets. Further, we use $S_t$ to denote the wealth at time point $t$, which is our controllable state variable. Under the famous and widely used self-financing assumption (Björk, 2009), have

$$dS_t = A_t \frac{dZ_t}{Z_t} = A_t \mu dt + \sigma dW_t.$$

Utility Function An investor’s goal is to maximize the expected utility of capital $U(Z_T)$ at some future time point $T$:

$$\max_\pi E_\pi [U(S_T)].$$

Negative values in the policy output are allowed, meaning the agent can short any asset. A common choice of utility is constant relative risk aversion (CRRA) $U(s) := \frac{1}{\gamma} s^{\gamma}$, with $\gamma < 1$ as a pre-selected risk aversion coefficient. (When $\gamma = 0$, $U(s) := \log(s)$ for convention.) A small $\gamma$ corresponds to a large penalization for risk-taking behavior. A demonstration of CRRA can be found in Figure 5. Note that with a small $\gamma$, the utility value does not increase much with more wealth, but decreases a lot with a less wealth.

![Figure 5: Demonstration of CRRA utility](image)

It should be noticed that the Merton model does not have stochastic factors. When implementing CARL to the synthetic Merton environment (Section 6.1), we assume/simulate fake factors $X_t$ from features $Y_t$:

$$dX_t = \mu X_t dt + v dW_t,$$

$$X_t = \phi(Y_t; \theta^*_R).$$

Note that the generated $X_t$ does not affect stock prices, and the true optimal policy should ignore $X_t$.

D.2.2. Policy Learning and Model Fitting

In this section, we detail the policy learning and model fitting for NSFM-LQC.

First, we detail the policy learning of NSFM-LQC. Specifically, we derive $F$ in Theorem 4.1 of NSFM-LQC in Lemma D.3.

Lemma D.3 (Policy Learning for Merton Model). For a Merton model defined in (9), the optimal policy and the function $F$ in Theorem 4.1 follows

$$\pi^*(t, s, z, y) = F(t, s, z, \phi(y; \theta^*_R); \theta^*_p) = Cs,$$

where $C$ is a $p \times 1$ vector.
Lemma D.3 is classic results in financial math and was proposed in Merton (1969).

Therefore, in Merton model, the optimal policy is independent of time, stock prices, features, and factors. The optimal investment strategy is to keep a constant portion of wealth in each asset along the time.

According to the Merton model formulation, there exist no factors affecting the evolution of stock prices. Therefore, there does not exist model fitting designed for Merton in CARL.

D.3. NSFM-Kim–Omberg

Kim–Omberg model (Kim & Omberg, 1996) is a standard model for portfolio optimization with predictable asset returns, which has been discussed extensively in the empirical literature (Welch & Goyal, 2008; Muhle-Karbe et al., 2017).

D.3.1. Modeling

We now provide the modeling details of the Kim–Omberg model, and then extend it an NSFM-Kim–Omberg model.

Consider $p$ assets with prices $Z_t = \{Z_{it}\}_{i=1}^p$ and a risk-free account with, for simplicity, zero interest rate. The SDE parameters $\sigma$, $v$, and $\mu$ are $p \times p$ matrices, with $v$ denoting the volatility of assets. Then, the dynamics of stock prices is formulated as

$$d\frac{Z_t}{Z_t} = X_t dt + \sigma dW_t,$$

$$dX_t = \mu X_t dt + v dW_t,$$

where $X_t = \phi(Y_t; \theta^*_R)$, and $\phi$ is a function of features $Y_t$ and parameters $\theta^*_R$.

Similar as the Merton model, the wealth $S_t$ follows

$$\frac{dS_t}{dt} := A_t \frac{dZ_t}{Z_t} = A_t X_t dt + A_t \sigma dW_t.$$

We apply NSFM (Section 3.1) to the Kim–Omberg model, leading to an NSFM-Kim–Omberg model:

$$\frac{dS_t}{dt} := A_t \frac{dZ_t}{Z_t} = A_t X_t dt + A_t \sigma dW_t,$$

$$dX_t = \mu X_t dt + v dW_t,$$

where $Y_t$ denotes the features available and we aim to learn $X_t$ from $Y_t$. Again, an investor’s goal is to maximize the expected constant relative risk aversion (CRRA) utility at future terminal time $T$: $J(\theta_{PR}) := \mathbb{E}_{\theta_{PR}} \left[ \frac{1}{\gamma} S_T^\gamma \right]$.

D.3.2. Policy Learning and Model Fitting

The policy learning and model fitting details of NSFM-Kim–Omberg are discussed in Lemma 4.2 and Lemma 4.4. The proof can be found in Appendix F.2.

D.4. NSFM-EVE

EVE model (Avanesyan et al., 2018), but considers a more challenging stochastic volatility setting, where the volatility of asset prices is also affected by stochastic factors.

D.4.1. Modeling

We now detail the modeling of EVE and then extend it to an NSFM-EVE model. Consider $p$ assets with prices $Z_t = \{Z_{it}\}_{i=1}^p$, and two independent $d_W$-dimensional Brownian motions $W_t$ and $W_t^\perp$. There exists a $p \times 1$ stochastic factor $X_t$ not
We consider a more concrete example of EVE satisfying the formulation and assumptions in Appendix D.4.1. Specifically, we use $A_i$, with $S_t$ as the wealth, EVE models asset prices via

$$
\frac{dZ^i_t}{Z^i_t} = \mu_i(X_t; \beta_1)dt + \sum_{j=1}^{d_W} \sigma_{ij}(X_t; \beta_2) dW^j_t, \quad i = 1, 2, \ldots, p,
$$

$$
dX_t = (M^T X_t + \omega) dt + \kappa(X_t; \beta_3)^T dB_t,
$$

$$
B_t = \rho^T W_t + L^T W_t^\perp,
$$

(12)

\[\mu, \sigma, \text{and } \kappa \text{ are parametric functions with parameter } \{\beta_i\}_{i=1}^3; M, \rho, L \text{ are } p \times p \text{ matrices}; \omega \text{ is a } p \times 1 \text{ vector. The SDE parameters include all the parameter matrices and } \beta \text{'s}. \]

Then, we define

\[\text{The conditions in Assumption D.4 are necessary for the process of } Y_t \text{ to be } [0, \infty)^k \text{-valued and affine. Under these conditions, the SDE in (12) has a unique weak solution which is affine and takes values in } [0, \infty)^k \text{ (Filipović & Mayerhofer, 2009). Further, the EVE model requires the following two assumptions:}
\]

**Assumption D.4.** $M$ has non-negative off-diagonal entries and $\omega \in [0, \infty)^k$. Further, we assume that there exist $\lambda(y), \Lambda$ and $L$, and $N$ such that $\mu(\cdot), \sigma(\cdot), \kappa(\cdot), \rho$ satisfy

$$
\lambda(y)^\top \lambda(y) = \mu(y)^\top \sigma(y)^{-1}(\sigma(y)^{-1})^\top \mu(y) = \Lambda^\top y
$$

$$
\kappa(y)^\top \kappa(y) = \text{diag}(L_1 y_1, L_2 y_2, \ldots, L_k y_k), \text{ with } L_1, L_2, \ldots, L_k \geq 0
$$

(13)

The conditions in Assumption D.4 are necessary for the process of $Y_t$ to be $[0, \infty)^k$-valued and affine. Under these conditions, the SDE in (12) has a unique weak solution which is affine and takes values in $[0, \infty)^k$ (Filipović & Mayerhofer, 2009). Further, the EVE model requires the following two assumptions:

**Assumption D.5** (Assumption 2.2 in Avanesyan (2021)). The functions $\mu : \mathbb{R}^p \to \mathbb{R}^n, \sigma : \mathbb{R}^p \to \mathbb{R}^{d_W \times n}$ are continuous. Moreover, the columns of $\rho$ belong to the range of left-multiplication by $\sigma(y)$ for all $y \in \mathbb{R}^p$.

**Assumption D.6** (EVE Condition in Avanesyan (2021)). For some $p \in [0, 1]$,

$$
\rho^\top \rho = p I_p.
$$

Note that when $p = 1$, $\rho$ is a vector and thus we define $p := \rho^\top \rho$.

**D.4.2. Concrete Example**

We consider a more concrete example of EVE satisfying the formulation and assumptions in Appendix D.4.1. Specifically, we use $D_\mu$, $D_\sigma$, and $D_\Lambda$ to denote diagonal matrices. Further, let $D(y)$ denote the diagonal matrix whose diagonal is $y$. Also, we use $y^{\odot k}$ for any $k \in \mathbb{R}$ to denote the component-wise power operation (Hadamard power).

Then, we define

$$
\mu(y) := D_\mu y^{\odot \frac{1}{2}}
$$

$$
\sigma(y) := D_\sigma D(y)
$$

$$
\kappa(y) := \rho^{-1} D_\kappa D(y^{\odot \frac{1}{2}})
$$

Then, we have

$$
\lambda(y) = D_\sigma^{-1} D_\mu y^{\odot \frac{1}{2}}.
$$

Further, we pose

$$
\rho^\top \rho = \rho \rho^\top = I,
$$

$$
A = 0.
$$
We further define the accumulated number of infected individuals as \( I_t \). The proof of Lemma D.7 and Lemma D.8 can be found in Appendix F.4.

\[
\begin{align*}
\lambda(y) &= (\sigma(y)^{-1})^T \mu(y) = D_\sigma^{-1} y^{\frac{\beta}{2}} \text{ and } \lambda(y)^T \lambda(y) = D_\sigma^{-2} y, \\
\kappa(y)^T \kappa(y) &= D_\kappa D(y^{\frac{\gamma}{2}})(\rho^{-1})^T \rho^{-1} D_\kappa D(y^{\frac{\gamma}{2}}) = D_\kappa^2 D(y), \\
\Gamma \kappa(y)^T \rho^T \lambda(y) &= \Gamma D(y^{\frac{\gamma}{2}}) D_\kappa (\rho^{-1})^T \rho^{-1} D_\sigma^{-1} y^{\frac{\gamma}{2}} = \Gamma D_\kappa D_\sigma^{-1} y.
\end{align*}
\]

Such a setup is shown to satisfy (12).

D.4.3. Policy Learning and Model Fitting

In this section, we detail the policy learning and model fitting for NSFM-EVE.

First, we detail the policy learning of NSFM-EVE. Specifically, we derive \( F \) in Theorem 4.1 of NSFM-EVE in Lemma D.7.

**Lemma D.7.** In the NSFM-EVE model defined above with \( p \) risky assets, given wealth \( s \), stock price \( z \), and feature \( y \), the optimal policy at time \( t \) is given by \( \pi^*(t, s, z, y) = F(t, s, z, \phi(y; \theta^*_R); \theta^*_p) \) with

\[
F(t, s, z, \phi(y; \theta^*_R); \theta^*_p) = k(t; \theta^*_p) \phi(y; \theta^*_R)s,
\]

where \( k(\cdot; \theta^*_p) : [0, T] \to \mathbb{R}^{p \times p} \) is a time-dependent function with parameter \( \theta^*_p \).

For model fitting, we follow Theorem 4.3, and derive the prediction objective \( L \) for NSFM-LQC in Lemma D.8.

**Lemma D.8.** For NSFM-EVE defined above, \( L(\theta_R) \) is derived as:

\[
L(\theta_R) := -\min_{\theta_s = (C_1, C_2)} \mathbb{E} \left\| \Delta \log(Z_t) - C_1 \phi(Y_t; \theta_R)^\gamma \Delta t - C_2 \phi(Y_t; \theta_R)^\gamma \Delta t \right\|_2^2,
\]

where \( C_1 \) and \( C_2 \) are \( p \times p \) diagonal matrices.

The proof of Lemma D.7 and Lemma D.8 can be found in Appendix F.4.

D.5. NSFM-SIR

SIR studies the spreading of disease (Gray et al., 2011; Allen, 2017; Mahrouf et al., 2021).

D.5.1. Modeling

We now detail the modeling of SIR and then extend it to an NSFM-EVE model. The numbers of susceptible, infected, and recovered individuals are assumed to follow a nonlinear dynamics. Specifically, consider \( S_t, I_t \), and \( R_t \) all taking values in \([0, \infty)\). We use a \( 2 \times 1 \) random vector \( Y_t \) to representing the available features in the data, and a \( 2 \times 1 \) random vector \( X_t \) as the stochastic factors. We consider a two dimensional action variable \( A_t \), where at time point \( t \), \( A_{1,t} \) denotes the treatment and \( A_{2,t} \) the vaccination. Then, NSFM-SIR is defined as:

\[
\begin{align*}
\frac{dS_t}{dt} &= -\frac{X_{1,t}}{N} S_t I_t - 1000 A_{2,t}, \\
\frac{dI_t}{dt} &= \frac{X_{1,t}}{N} S_t I_t - X_{2,t} I_t - 1000 A_{1,t}, \\
\frac{dR_t}{dt} &= X_{2,t} I_t + 1000 A_{2,t}, \\
X_t &= \phi(Y_t), \\
dX_t &= (M^T X_t + \omega) dt + \kappa(X_t; \beta_3)^T dB_t.
\end{align*}
\]

The coefficients are selected as the estimated values of COVID-19 using the data of Algeria in Lounis & Bagal (2020).

We further define the accumulated number of infected individuals as \( I'_t \) with

\[
\frac{dI'_t}{dt} = \frac{X_{1,t}}{N} S_t I_t - X_{2,t} I_t
\]
We aim to conduct decision-making to minimize the following cost function:

\[
\min_{\pi} \mathbb{E} \left[ \int_0^T A_t' A_t dt + I_t' \right],
\]

which is designed to simultaneously minimize the number of infected individuals and the amount of the actions taken.

We start with \( S_0 = 40000000 \), and \( I_0 = 10000 \), as suggested by Lounis & Bagal (2020).

**D.6. Policy Learning and Model Fitting**

To apply CARL, we approximate NSFM-SIR with an NSFM-LQC. The derivation of the exact form \( F \) for NSFM-SIR is deferred to future work.
E. Extended Results for Theorem 4.1 and Theorem 4.3

In this section, we formalize and prove Theorem 4.1 and Theorem 4.3. We first list the definition and assumptions required in Appendix E.1. Then, we formally present our main theoretical results in Appendix E.2. We finally provide the lemmas required and the proof in Appendix E.3 and Appendix E.4.

E.1. Definitions and Assumptions for Theorem 4.1 and Theorem 4.3

In order to provide and prove the formal version of Theorem 4.1 and Theorem 4.3, we require the following definitions and assumptions.

E.1.1. Definitions

To start with, we redefine the problem of NSFM. Let \((\Omega, \mathcal{F}, \mathbb{P})\) be a complete filtered probability space with \(\mathcal{F} = \{\mathcal{F}_t\}_{t \in [0,T]}\). We define a multi-dimensional independent standard Brownian motion \(W_t\) on \(\mathcal{F}\) such that \(\mathcal{F}\) is the natural filtration generated by \(W_t\). Then, the decision-making problem of NSFM can be formulated as

\[
\max J(\pi) \text{ with } J(\pi) := \mathbb{E} \left[ \int_0^T h(S_t, a_t)dt + r(S_t) \mid \mathcal{F}_0 \right].
\]

\[
dS_t = f_1(S_t, X_t, A_t; \theta^*_S) \, dt + g_1(S_t, X_t, A_t; \theta^*_S) \, dW_t, \\
dX_t = f_2(X_t; \theta^*_S) \, dt + g_2(X_t; \theta^*_S) \, dW_t, \\
X_t = \phi(Y_t; \theta^*_R).
\]

With (14) provide the following definitions.

**Definition E.1.** We use \(\Xi_t\) to denote the random variable consisting of both \(S_t\) and \(X_t = \phi(Y_t; \theta^*_R)\):

\[
\Xi_t := [S_t^T, \phi(Y_t)^T]^T,
\]

where \(\phi\) is the true representation function for the data generation. We further use \(\xi := [s^T, \phi(y; \theta^*_R)^T]^T\) to denote the assignment of \(\Xi\).

By Definition E.1, we can rewrite the coefficients in (14) as \(f_1(\xi, a), g_1(\xi, a)\) and \(g_2(\xi)\), where the SDE parameter \(\theta^*_S\) is also omitted. Next, we define the notation for the conditional random variable.

**Definition E.2.** For \(T \geq t' > t \geq 0\), we use \(\Xi_{t', \xi}(t')\) to denote the random variable \(\Xi_{t'}, \xi\) conditional on \(\Xi_t = \xi\) and the policy \(\pi\).

We then define the value function.

**Definition E.3 (Value Function).** The value function of (14) is defined as

\[
V(t, s, y) := \max_a \mathbb{E} \left\{ \int_0^T h(S_t, a_t)dt + r(S_t) \mid S_0 = s, Y_0 = y \right\}.
\]

Note that the value function only depends on \(y\) through \(x = \phi(y; \theta^*_R)\) according to the dynamics (14), and it can thus be written as a function of \((t, s, \phi(y; \theta^*_R))\). In particular, with abuse of notation, we also write \(V(t, \xi)\).

It can been noticed that \(V(t, \xi)\) is related to the performance objective in (14) by

\[
\max J(\pi) = \mathbb{E}_{\xi} V(0, \xi).
\]

Further, as a key concept in stochastic calculus and optimal control, we define the infinitesimal generator, which carries the information about the process of value function over time.

**Definition E.4 (Infinitesimal Generator).** Under (14), the infinitesimal generator is defined as

\[
\mathcal{H} \left( t, \xi, a, \frac{\partial V(t, \xi)}{\partial \xi}, \frac{\partial^2 V(t, \xi)}{\partial \xi^2} \right) := \frac{\partial V(t, s, \phi(y))}{\partial t} + \left[ f_1(\xi, a) \right] \frac{\partial V(t, \xi)}{\partial \xi} + \frac{1}{2} \Sigma(\xi, a) \frac{\partial^2 V(t, \xi)}{\partial \xi^2},
\]

where

\[
\Sigma(\xi, a) \cdot \vartheta := \frac{\partial^2 V(t, \xi)}{\partial \xi^2},
\]

and \(\vartheta \) is a vector of size \(n \times 1\).
with \( : \) denoting the Frobenius inner product and
\[
\Sigma(\xi, a) := \begin{pmatrix} g_1(\xi, a) \\ g_2(\xi) \end{pmatrix}, \quad \begin{pmatrix} g_1(\xi, a) \\ g_2(\xi) \end{pmatrix}^T.
\]

E.1.2. Assumptions

To conduct analytical analysis on the optimal policy, we require the following assumptions.

**Assumption E.5.** We assume that the SDEs for \( S_t \) and \( X_t \) in (14) have unique strong solution.

The specific conditions for Assumption E.5 to hold can be found in (Karatzas & Shreve, 1987).

**Assumption E.6.** The value function \( V(t, \xi) \) is continuously differentiable in \( t \) and twice continuously differentiable in \( \xi \).

Assumption E.6 may be weakened in the theory of viscosity solutions. We refer the interested readers to Fleming & Soner (2006). Also, we require the following commonly used assumption for maximum likelihood estimation.

**Assumption E.7.** For any trajectory \( \tau := \{t_i, s_i, x_i, a_i \mid i \in [m] \} \) under NSFM (14), with \( \phi^* \) as any candidate representation function, we assume that
\[
|E \log P(\tau; \phi^*)| < \infty.
\]

E.2. Formal Expression of Theorem 4.1 and Theorem 4.3

We now provide the formal version of Theorem 4.1 and Theorem 4.3.

**Theorem E.8** (Formal Version of Theorem 4.1 and Theorem 4.3). Under the definitions and assumptions in Section E.1, there exists a function \( F \) such that an optimal feedback policy \( \pi^* \) of an NSFM (14) can be written as \( \pi^*(t, s, y) = F(t, s, \phi(y; \theta^*_R)) \) and

[Policy Learning (Theorem 4.1)] With \( V, H, \) and \( \xi \) defined in Definition E.3 and E.4, \( F \) satisfies
\[
F(t, s, \phi(y)) = \sup_a \left\{ h(s, a) + H \left( t, \xi, a, \frac{\partial V(t, \xi)}{\partial \xi}, \frac{\partial^2 V(t, \xi)}{\partial \xi^2} \right) \right\},
\]
where \( V \) is the solution to the following Hamilton–Jacobi–Bellman equation
\[
\frac{\partial V(t, \xi)}{\partial t} = \sup_a \left\{ H \left( t, \xi, a, \frac{\partial V(t, \xi)}{\partial \xi}, \frac{\partial^2 V(t, \xi)}{\partial \xi^2} \right) \right\},
\]
with the terminal condition
\[
V(T, s, y) = r(s).
\]

[Model Fitting (Theorem 4.3)] By discretizing the NSFM in (14) with an explicit Euler manner, \( \phi(\cdot; \theta^*_R) \) satisfies
\[
\phi(\cdot; \theta^*_R) \in \arg \min_{\phi} \min_{\theta_R} \mathbb{E} \| \Delta S_t - f_1(S_t, \phi(Y_t), A_t; \theta_S) \Delta t \|^2_2,
\]
when \( \Delta t \) is small enough.

Note that, in practice, we will parameterize \( \phi \) by \( \phi(\cdot; \theta_R) \) and \( F \) by \( F(\cdot; \theta_RW) \) leading to
\[
\pi(t, s, y; \theta_{WR}) = F(t, s, \phi(y; \theta_R); \theta_R),
\]
like Theorem 4.1 and Theorem 4.3. By Theorem E.8, we only need to study candidate policies under a function class corresponding to the solutions of HJB PDEs. Such a function class can be quite constrained, and ease the learning process as shown in Section 6. Also, Theorem E.8 requires the policy to admit an NSFM, by minimizing a square loss. Note that the explicit form of \( H \) vary over different SDEs and problems studied. Various of results can be used for the derivation under specific settings (Fouque et al., 2011; Ait-Sahalia & Kimmel, 2010; Soner & Touzi, 2002; Fleming & Soner, 2006).
We prove Theorem E.8 by considering the policy learning part (Appendix E.4.1) and the model fitting part (Appendix E.4.2) separately.

**E.4. Proof of Theorem E.8**

We prove Theorem E.8 by considering the policy learning part (Appendix E.4.1) and the model fitting part (Appendix E.4.2) separately.

**E.4.1. Proof of Policy Learning**

Now we prove the policy learning part of Theorem E.8. To start with, we apply Lemma E.9 with $T \geq t' \geq t \geq 0$:

$$V(t, \xi) = \sup_\pi \mathbb{E} \left[ \int_t^{t'} h(S_s, A_s) ds + V(t', \Xi_{t', \xi}(t')) \mid \mathcal{F}_t \right],$$

(16)

Next, we consider the last component of the right hand side of (16). Under Assumption E.6, $V$ is smooth enough and satisfies the assumptions of Itô’s formula. Then, by Itô’s formula,

$$V(t', \Xi_{t', \xi}(t')) = V(t, \xi) + \int_t^{t'} \mathcal{H} \left( s, \Xi_{t', \xi}(s), A_s, \frac{\partial V(s, \Xi_{t', \xi}(s))}{\partial \xi}, \frac{\partial^2 V(s, \Xi_{t', \xi}(s))}{\partial \xi^2} \right) ds$$

$$+ \int_t^{t'} \left( \frac{\partial V(s, \Xi_{t', \xi}(s))}{\partial \xi} \right) ^\top g_\mathcal{X}(\Xi_{t', \xi}(s), A_s) dW_s,$$

(17)

where $g_\mathcal{X}$ is the diffusion coefficient of $\mathcal{X}_t$. We do not explicitly write out the formulation of $g_\mathcal{X}$, since it will be omitted by the following analysis. Next, we take expectation on the both side of (17):

$$\mathbb{E}[V(t', \Xi_{t', \xi}(s) \mid \mathcal{F}_t)] = V(t, \xi) + \mathbb{E} \left[ \int_t^{t'} \mathcal{H} \left( s, \Xi_{t', \xi}(s), A_s, \frac{\partial V(s, \Xi_{t', \xi}(s))}{\partial \xi}, \frac{\partial^2 V(s, \Xi_{t', \xi}(s))}{\partial \xi^2} \right) ds \right]$$

$$+ \int_t^{t'} \left( \frac{\partial V(s, \Xi_{t', \xi}(s))}{\partial \xi} \right) ^\top f_\mathcal{X}(\Xi_{t', \xi}(s), A_s) dW_s \mid \mathcal{F}_t$$

$$= \mathbb{E} \left[ \int_t^{t'} \mathcal{H} \left( s, \Xi_{t, \xi}(s), A_s, \frac{\partial V(s, \Xi_{t, \xi}(s))}{\partial \xi}, \frac{\partial^2 V(s, \Xi_{t, \xi}(s))}{\partial \xi^2} \right) ds \mid \mathcal{F}_t \right],$$

where the last equality is due to the fact that an Itô integral is a martingale. (See Lemma C.2.) Then, by taking the results into (16):

$$V(t, \xi) = V(t, \xi) + \sup_\pi \left\{ \mathbb{E} \left[ \int_t^{t'} h(S_{t, \xi}(s), A_s) ds \mid \mathcal{F}_t \right] \right\}$$

$$+ \mathbb{E} \left[ \int_t^{t'} \mathcal{H} \left( s, \Xi_{t, \xi}(s), A_s, \frac{\partial V(s, \Xi_{t, \xi}(s))}{\partial \xi}, \frac{\partial^2 V(s, \Xi_{t, \xi}(s))}{\partial \xi^2} \right) ds \mid \mathcal{F}_t \right] \},$$

which suggests

$$0 = \sup_\pi \left\{ \mathbb{E} \left[ \int_t^{t'} h(S_s, A_s) ds \mid \mathcal{F}_t \right] \right\}$$

$$+ \mathbb{E} \left[ \int_t^{t'} \mathcal{H} \left( s, \Xi_{t, \xi}(s), A_s, \frac{\partial V(s, \Xi_{t, \xi}(s))}{\partial \xi}, \frac{\partial^2 V(s, \Xi_{t, \xi}(s))}{\partial \xi^2} \right) ds \mid \mathcal{F}_t \right] \}. $$
We then divide the equation by $t' - t$ and let $t' - t$ go to zero:

$$0 = \sup_a \left[ h(s, a) + \mathcal{H} \left( t, \xi, a, \frac{\partial V(s, \xi)}{\partial \xi}, \frac{\partial^2 V(s, \xi)}{\partial \xi^2} \right) \right],$$

which is the HJB PDE for NSFM. Remember $\xi = [s^T, \phi(y; \theta^*_R)]^T$. Thus, the optimal policy can be written as $\pi^* = F(t, s, \phi(y; \theta^*_R))$ with

$$F(t, s, \phi(y)) = \arg \sup_a \mathcal{H}(s, a) + \mathcal{H} \left( t, \xi, a, \frac{\partial V(s, \xi)}{\partial \xi}, \frac{\partial^2 V(s, \xi)}{\partial \xi^2} \right).$$

By the classical verification analysis in Fleming & Rishel (1975), we can show that the optimal policy of NSFM does satisfy $\pi^*(t, s, y) = F(t, s, \phi(y; \theta^*_R))$.

### E.4.2. Proof of Model Fitting

Now we justify the square-loss objective for the model fitting of Theorem E.8. To start with, we discretize the dynamics in (14) with time interval $\Delta t$ by Euler method and yield

$$\Delta S_t = f_1(S_t, X_t, A_t; \theta^*_S) \Delta t + g_1(S_t, X_t, A_t; \theta^*_S) \Delta W_t, \quad (18)$$

By the characteristics of Brownian motion, we have

$$S_{t+\Delta t} | S_t, Y_t \sim N(f(S_t, \phi(Y_t; \theta^*_R); \theta^*_S), g(S_t, \phi(Y_t; \theta^*_R); \theta^*_S)), \quad (19)$$

In words, the conditional distribution of $S_{t+\Delta t}$ is normal with expectation and covariance depending on $S_t$ and $Y_t$.

Next, we prove

$$\theta^*_R, \theta^*_S \in \arg \min_{\theta_R, \theta_S} \mathbb{E} \left[ ||S_{t+\Delta t} - f(S_t, \phi(Y_t; \theta^*_R); \theta_S)||^2 \right],$$

where $\mathbb{E}$ is on $S_{t+\Delta t}, S_t,$ and $Y_t$.

According to (19),

$$\mathbb{E}(S_{t+\Delta t} | S_t, Y_t) = f(S_t, \phi(Y_t; \theta^*_R); \theta^*_S). \quad (20)$$

Then, we consider one specific dimension of $S_{t+\Delta t}$ denoted as $S_{t+\Delta t}^j$ for $j \in [p]$. Then,

$$\mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S))^2 \right] = \mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S))^2 \right] + \mathbb{E}\left[ (f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S) - f_j(S_t, \phi(Y_t; \theta_R); \theta_S))^2 \right]$$

$$+ 2\mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S))(f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S) - f_j(S_t, \phi(Y_t; \theta_R); \theta_S)) \right]$$

Further, we focus on the last component:

$$\mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S))^2 \right] = \mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S))^2 \right] | S_t, Y_t] = 0,$$

where the last equality is due to (20). Therefore, we can derive

$$\mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta_R); \theta_S))^2 \right] = \mathbb{E}\left[ (S_{t+\Delta t}^j - f_j(S_t, \phi(Y_t; \theta_R); \theta_S))^2 \right]$$

$$+ \mathbb{E}\left[ (f_j(S_t, \phi(Y_t; \theta^*_R); \theta_S) - f_j(S_t, \phi(Y_t; \theta_R); \theta_S))^2 \right],$$

which is minimized when $\theta = \theta^*$. Further, by summing over $j$, we conclude

$$\theta^*_R, \theta^*_S \in \arg \min_{\theta_R, \theta_S} \mathbb{E} \left[ ||S_{t+\Delta t} - f(S_t, \phi(Y_t; \theta_R); \theta_S)||^2 \right],$$
F. Derivation of Theorem E.8 for Different CARL Applications

Previously in Appendix D, we provide the procedures for the application of CARL on NSFM-LQC, Merton NSFM-Kim–Omberg, NSFM-EVE, and NSFM-SIR. Now, we provide rigorous derivation of such procedures by applying Theorem E.8 to these models.

F.1. NSFM-LQC

F.1.1. Assumptions for Lemma D.1 and Lemma D.2

In general, we require the same assumptions as in Section E.1, which can be summarized as threefold:

1. The defined SDE has the unique strong solution.
2. The value function exists, and is smooth enough.
3. The corresponding differential Riccati equation has a solution.

The specific conditions on the model coefficients for the three assumptions to be true are well studied under various settings (Sun & Yong, 2020; Yong & Zhou, 1999)

F.1.2. Technical Lemmas for Lemma D.1 and Lemma D.2

Lemma F.1. Consider a stochastic linear-quadratic control problem

\[ dS_t = (BS_t + UA_t) dt + \sum_{j=1}^{p} D_j A_t dW_j^t, \]

with the objective function

\[ \max \mathbb{E}_\pi \left[ \int_0^T \left( (QS_t)^\top S_t + (RA_t)^\top A_t \right) dt + (GS_T)^\top S_T \right], \]

where Q, R, and G are known matrices with proper dimensions. Under the assumptions in Section F.1.1, the optimal policy can be derived as

\[ \pi^*(t, s) = \Lambda(t, K(t))^{-1} U^\top K(t)s, \]

where

\[ \Lambda(t, K(t)) = R + \sum_{j=1}^{p} (D_j^\top k(t)D_j). \]

Also, K(t) solves the differential Riccati equation:

\[ K(t) = -e^{B^\top (T-t)} Ge^{B(T-t)} - \int_t^T e^{B^\top (T-s)} K(s)^\top U(A(s, K(s))^\top)^{-1} U^\top K(s) e^{B^\top (T-s)} ds, \]

with

\[ K(T) = 0. \]

Proof. Lemma F.1 is well established and can be found in various of literature (Sun & Yong, 2020; Yong & Zhou, 1999). \( \square \)

F.1.3. Proof of Lemma D.1 and Lemma D.2

We review the NSFM for LQC:

\[ dS_t = (BS_t + UA_t + X_t) dt + \sum_{j=1}^{p} D_j A_t dW_j^t, \]

\[ X_t = \phi(Y_t), \]

\[ dX_t = \mu X_t dt + \nu dW_t. \]
By taking \( \Xi_t \) (the combination of \( S_t \) and \( X_t \)) as the state variables, we can reformulate the problem as a classic LQC problem. As slight abuse of notion, we have

\[
d\Xi_t = B\Xi_t + UA_t + \sum_{j=1}^{p} (D_j A_t + \beta_t) dW^j_t,
\]

with all the coefficients redefined. Then, we can apply Lemma F.1 for Lemma D.1.

The derivation of Lemma D.2 is straightforward by applying Theorem E.8.

F.2. NSFM-Kim–Omberg

F.2.1. Definitions and Assumptions for Lemma 4.2 and Lemma 4.4

Definition F.2. As slight abuse of notion, we use \( s \) to denote an assignment of the capital \( S_t \), \( y \) to denote an assignment of \( Y_t \), and \( x \) as an assignment of the factors \( X_t := \phi(Y_t) \), with \( \phi \) as the true representation function for data generation.

For portfolio optimization, we first define the value function.

Definition F.3 (Value Function). The value function for portfolio optimization is

\[
V(t, s, y) := \max_{\pi} \mathbb{E}_\pi \left[ \frac{1}{\gamma} S_T^\top | S_0 = s, Y_0 = y \right].
\]

Definition F.4 (Stabilizability). For any pair of matrices \( A \in \mathbb{R}^{p \times p} \) and \( B \in \mathbb{R}^{p \times p} \), \((A, B)\) is stabilizable if there exists a \( p \times p \) matrix \( K \) such that \( A + BK \) is a stability matrix, i.e., all characteristic roots of \( A + BK \) have negative real parts.

Definition F.5 (Detectability). For any pair of matrices \( A \in \mathbb{R}^{p \times p} \) and \( B \in \mathbb{R}^{p \times p} \), \((A, B)\) is detectable if for any solution \( x(t) \) of \( \dot{x} = Ax + Bx(t) \equiv 0 \), we have \( \lim_{t \to \infty} x(t) = 0 \).

We require the following assumption on the studied portfolio optimization problem:

Assumption F.6. Trading is continuous without transaction costs, fees, or taxes. Also, the investor is a price taker and does not possess enough market power to influence prices. Further, the dynamics of the portfolio are self-financing such that:

\[
dS_t = \pi_t (dS_t - \frac{dS_t}{S_t}).
\]

Further, to guarantee the analytical form of the optimal policy, we require the following assumptions.

Assumption F.7. The matrix \( \Sigma := \sigma \sigma^\top \) is non-singular.

Assumption F.8. The matrix pair \( \left( \frac{1}{\gamma - 1} \Sigma^{-1} \sigma v^\top - I, \sqrt{\frac{1}{\gamma - 1}} v \right) \) is stabilizable, and \( \left( \frac{1}{\gamma - 1} \Sigma^{-1} \sigma v^\top - I, \sqrt{\frac{1}{\gamma - 1}} \sigma^{-1} \right) \) is detectable.

F.2.2. Technical Lemmas

Lemma F.9 (Theorem 4.3 in Behr et al. (2019)). Consider \( A, B, D, \) and \( X_0 \in \mathbb{R}^{p \times p} \). Consider a differential Riccati equation:

\[
\frac{dX(t)}{dt} = A^\top X(t) + X(t)A - X(t)BB^\top X(t) + D^\top D, \quad X(0) = X_0.
\]

Let \((A, B)\) be controllable, and \((A, D)\) be detectable. Define \( X_\infty \) as the unique symmetric positive stabilizing solution of the corresponding algebraic Riccati equation. Let \( \hat{A} := A - BB^\top X_\infty \) and \( X_L \) be the unique symmetric positive semidefinite solution to the Lyapunov equation:

\[
\hat{A}X_L + X_L\hat{A}^\top + BB^\top = 0.
\]

Then, the solution to (21) can be derived as

\[
X(t) = X_\infty - e^{t\hat{A}} (X_\infty - X_0) \left[ I - (X_L - e^{t\hat{A}} X_L e^{t\hat{A}}^\top) (X_\infty - X_0) \right]^{-1} e^{t\hat{A}}.
\]
Proof. See Theorem 4.3 in Behr et al. (2019).

F.2.3. Auxiliary Results for Lemma 4.2

Lemma F.10. Under the definitions and assumptions in Section E.1, and Section F.2.1. The optimal policy for (3) satisfies:

\[ \pi^*(t, s, y) = \frac{1}{1 - \gamma} (\sigma^\top)^{-1} \left[ \phi(y; \theta^*_R) + \sigma^\top K_1(t) \phi(y; \theta^*_R) \right] s, \]

where \( K_1(t) \) is a \( p \times p \) deterministic time-dependent matrix, and \( K_2(t) \) a \( p \times 1 \) vector.

Proof. By Theorem E.8, we can write both the value function and the optimal policy function using \( t, S_t, \) and \( X_t = \phi(Y_t) \):

\[ \pi^*(t, S_t, Y_t) = F(t, S_t, X_t), \]

with \( X_t := \phi(Y_t) \).

By Assumption F.6, we derive the dynamics of the wealth:

\[ dS_t = \pi(t, S_t, X_t)^\top X_t dt + \pi(t, S_t, X_t)^\top \sigma dW_t. \]

For the ease of presentation, when possible, we omit the arguments for the policy function \( \pi(t, s, y) \) and the value function \( V(t, s, y) \) and write them as \( \pi \) and \( V \). Then, following the derivations similar as Section E.4.1, we can derive the HJB PDE for portfolio optimization under an NSFM.

\[ \frac{\partial V}{\partial t} + \max_{\pi} \left[ \pi^\top x \frac{\partial V}{\partial s} + x^\top \frac{\partial V}{\partial x} + \frac{1}{2} \pi^\top \Sigma \pi^2 V + \pi^\top \sigma v^\top \frac{\partial^2 V}{\partial s \partial x} + \text{tr} \left( \frac{\partial^2 V}{\partial s^2} vv^\top \right) \right] = 0, \]

with the terminal condition

\[ V(T, s, x) = \frac{1}{\gamma} s^\gamma. \]

Solving the maximization in (22) we can derive

\[ \pi^* = -\Sigma^{-1} \frac{1}{\partial^2 V / \partial s^2} \left[ \left( \frac{\partial V}{\partial z} \right)^\top x + \sigma v^\top \frac{\partial V}{\partial z} x \right]. \] (23)

Taking (23) into (22), we have

\[ \frac{\partial V}{\partial t} + x^\top \frac{\partial V}{\partial x_t} + \frac{1}{2} \text{tr} \left( \frac{\partial^2 V}{\partial x_t^2} vv^\top \right) - \frac{1}{2} \left( \frac{\partial V}{\partial x_t} \right)^2 x^\top \Sigma^{-1} x \]

\[ - x^\top \Sigma^{-1} \sigma v^\top \frac{\partial V}{\partial x_t} \frac{\partial^2 V}{\partial s \partial x} - \frac{\partial^2 V}{\partial s^2} vv^\top \frac{\partial^2 V}{\partial s \partial x} = 0. \] (24)

To solve the HJB PDE, we conjecture that

\[ V(t, z, x) = \frac{1}{\gamma} z^\gamma \exp \left( K_3(t) + K_2(t)^\top x + \frac{1}{2} x^\top K_1(t)x \right), \] (25)

where \( K_3(t) \) is a scalar, \( K_2(t) \) a vector, and \( K_1(t) \) a symmetric matrix. Taking (25) into the terminal condition, we have

\[ K_1(T) = 0, \]
\[ K_2(T) = 0, \]
\[ K_3(T) = 0, \]
Taking (26) into the HJB PDE and using the trick of trace, we derive

\[
\begin{align*}
\frac{\partial V}{\partial s} &= s^{\gamma-1} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right), \\
\frac{\partial^2 V}{\partial s^2} &= (\gamma - 1)s^{\gamma-2} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right), \\
\frac{\partial^2 V}{\partial s \partial x} &= s^{\gamma-1} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) [K_2 + K_1 x], \\
\frac{\partial V}{\partial x} &= \frac{1}{\gamma} s^{\gamma} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) [K_2 + K_1 x], \\
\frac{\partial^2 V}{\partial x^2} &= \frac{1}{\gamma} s^{\gamma} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right), \\
\frac{\partial}{\partial t} &= \frac{1}{\gamma} s^{\gamma} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) \left[ \frac{\partial K_3}{\partial t} + \frac{\partial K_2^T}{\partial t} x + \frac{1}{2} x^T \frac{\partial K_1}{\partial t} x \right].
\end{align*}
\]

Taking (26) into the HJB PDE and using the trick of trace, we derive

\[
\begin{align*}
\frac{1}{\gamma} s^{\gamma} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) \left[ \frac{\partial K_3}{\partial t} + \frac{\partial K_2^T}{\partial t} x + \frac{1}{2} x^T \frac{\partial K_1}{\partial t} x \right] \\
+ \frac{1}{\gamma} s^{\gamma} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) x^T A^T [K_2 + K_1 x] \\
+ \frac{1}{2} s^{\gamma} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) \text{tr} \left\{ v^T K_2 K_2^T v + 2K_2^T v v^T K_1 x + x^T K_1 v v^T K_1 x + v^T K_1 v \right\} \\
- \frac{s^{\gamma-1} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right)}{\gamma - 1} x^T \Sigma^{-1} x \\
- \frac{s^{\gamma-1} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right)}{\gamma - 1} s^{\gamma-2} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) x^T \Sigma^{-1} x \\
- \frac{s^{\gamma-1} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right)}{\gamma - 1} [K_2 + K_1 x] \\
- \frac{s^{\gamma}}{2} \exp \left( K_3 + K_2^T x + \frac{1}{2} x^T K_1 x \right) \frac{(K_2 + K_1 x)^T (K_2 + K_1 x)}{\gamma - 1} = 0.
\end{align*}
\]

Then, we factorizing some components and derive:

\[
\begin{align*}
\left( \frac{\partial K_3}{\partial t} + \frac{\partial K_2^T}{\partial t} x + \frac{1}{2} x^T \frac{\partial K_1}{\partial t} x \right) + x^T (K_2 + K_1 x) \\
+ \frac{1}{\gamma} \text{tr} \left\{ v^T K_2 K_2^T v + 2K_2^T v v^T K_1 x + x^T K_1 v v^T K_1 x + v^T K_1 v \right\} - \frac{\gamma}{\gamma - 1} x^T \Sigma^{-1} x \\
- \frac{x^T \Sigma^{-1} \sigma v^T}{\gamma - 1} (K_2 + K_1 x) - \frac{\gamma}{2} \frac{(K_2 + K_1 x)^T (K_2 + K_1 x)}{\gamma - 1} = 0.
\end{align*}
\]

By the characteristic of trace, we have

\[
\begin{align*}
\text{tr} \left\{ v^T K_2 K_2^T v + 2K_2^T v v^T K_1 x + x^T K_1 v v^T K_1 x + v^T K_1 v \right\} \\
= \text{tr} \left\{ v^T K_2 K_2^T v + v^T K_1 v \right\} + 2K_2^T v v^T K_1 x + x^T K_1 v v^T K_1 x.
\end{align*}
\]
Then, by grouping the components with the same order of \( x_t \), we can derive

\[
\frac{\partial K_3}{\partial t} + \frac{1}{2} \text{tr} \left\{ v^T K_2 K_2^T v + v^T K_1 v \right\} - \frac{\gamma}{2(\gamma - 1)} (K_2^T vv^T K_2) \\
+ \left[ \frac{\partial K_2}{\partial t} + K_2^T + K_1 v v^T K_1 - \frac{\gamma}{\gamma - 1} (\Sigma^{-1} \sigma v K_2 + K_1 vv^T K_2) \right] x \\
+ \frac{1}{2} x^T \left[ \frac{\partial K_1}{\partial t} + K_1 v v^T K_1 + 1 + K_1 \right] \\
- \frac{\gamma}{\gamma - 1} (K_1 vv^T K_1 + \Sigma^{-1} + \Sigma^{-1} \sigma v^T K_1 + K_1 v \sigma^T \Sigma^{-1}) x^T = 0.
\]

To satisfy the PDE, we derive

\[
\frac{\partial K_3}{\partial t} + \frac{1}{2} \text{tr} \left\{ v^T (K_2 K_2^T + K_1) v \right\} - \frac{\gamma}{2(\gamma - 1)} (K_2^T vv^T K_2) = 0,
\]

\( K_3(T) = 0 \),

for \( K_3 \);

\[
\frac{\partial K_2}{\partial t} + K_1 v v^T K_2 + K_2 - \frac{\gamma}{\gamma - 1} (\Sigma^{-1} \sigma v K_2 + K_1 vv^T K_2) = 0,
\]

\( K_2(T) = 0 \),

for \( K_2 \);

\[
\frac{\partial K_1}{\partial t} + K_1 v v^T K_1 + 1 + K_1 \\
- \frac{\gamma}{\gamma - 1} (K_1 vv^T K_1 + \Sigma^{-1} + \Sigma^{-1} \sigma v^T K_1 + K_1 v \sigma^T \Sigma^{-1}) = 0,
\]

\( K_1(T) = 0 \),

for \( K_1(T) \). Under Assumption F.7 and Assumption F.8, the solutions to \( K_1(t) \) exist, and \( K_2(t) \equiv 0 \), suggesting that the conjecture is correct.

Now, we take (25) into (23) and derive

\[
\pi^*(t, z, y) = \frac{1}{1 - \gamma} (\sigma^T)^{-1} \left[ \phi(y; \theta_R^*) + \sigma^T (K_1 \phi(y; \theta_R^*)) \right],
\]

which finishes the proof.

F.2.4. AUXILIARY RESULTS FOR LEMMA 4.4

**Lemma F.11.** Under the NSFM-Kim–Omberg (3), with factors \( X_t = \{X_{it}\}_{i=1}^p \) the stock price \( Z_t = \{Z_{it}\}_{i=1}^p \) satisfies

\[
d \log(Z_{it}) = \left( X_t^i - \frac{1}{2} \sigma_i \sigma_i^T \right) dt + \sigma_i dW_t,
\]

where \( \sigma_i \) is the \( i^{th} \) row vector of \( \sigma \).

**Proof.** We apply Itô’s formula (Lemma C.1) to the \( i^{th} \) component of \( \log(\tilde{S}_t) \).

F.3. Proof for Lemma 4.2 and Lemma 4.4

With \( \phi \) as the true representation function for data generation, by Lemma F.10, we conclude that the optimal policy can be written as

\[
\pi^*(t, z, y) = k(t) \phi(y; \theta_R^*) z,
\]
with
\[ k(t) = I + (\sigma \sigma^T)^{-1} \sigma v^T K_1(t)(\sigma \sigma^T), \]
which proves Lemma 4.2.

Then, we apply Theorem E.8 to the log asset price SDE in Lemma F.11 and prove Lemma 4.4.

F.3.1. Function Form of $K_1(t)$

We can further derive the function from of $K_1(t)$ corresponding to the $k(t)$ in Lemma 4.2:

**Theorem F.12.** Under Assumption F.8, there exists symmetric matrices $\{M_i\}_{i=1,2,3}$ and a matrix $C$, such that the $K_1(t)$ in Lemma F.10 satisfies
\[ K_1 = M_1 - [C^T M_2 (C^T) + M_3]^{-1}. \]

**Proof.** Note that $K_1(t)$ satisfies a differential Riccati equation. Thus, we review the general result regarding the solution to a differential Riccati equation in Lemma F.9. Applying Lemma F.9 to (27), we finish the proof.

However, in practice, we notice that it is not necessary to use such a explicit form for $k(t)$: directly parameterizing with a simple neural network can provide decent performances.

F.4. NSFM-EVE

F.4.1. Technical Lemmas for Lemma D.7 and Lemma D.8

**Lemma F.13** (Summary of (Avanesyan, 2021)). The optimal policy for the portfolio optimization under the setting described in Section D.4.1 fulfills
\[ \sigma(y_t)\pi^*_t = \frac{1}{\gamma} \left[ \lambda(y_t) + q \rho \kappa(y_t) \Phi_t \right]. \]
$\Phi_t$ is a $k$-dimensional time-varying coefficient, whose components solve a system of Riccati ODEs.

**Proof.** The proof follows Avanesyan (2021).

F.4.2. Proof for Lemma D.7 and Lemma D.8

The proof of Lemma D.7 directly follows Lemma F.13. The proof of Lemma D.8 follows the same rationale as Section F.2.
### G. Extended Results for Discretization Error and Exploration Error

Note that CARL learns a policy conducted in discrete time with exploration, while leveraging the information from the continuous-time model, SDEs. Therefore, both the theoretical analysis in Section 5, and the empirical analysis in Section 6 focuses on the performance of CARL in such a discrete-time stochastic-policy setting. One question is that how well the learned policy performs compared to the continuous-time optimal policy without exploration. We provide the following results.

#### G.1. Definitions and Assumptions

To start with, we define the policy function set and the value function for policies with time interval $\Delta t$ and exploration.

**Definition G.1.** We use $\mathcal{F}_{\Delta t, \delta}$ to denote the policy function class with time interval $\Delta t$ and the exploration parameterized by $\delta$ for exploration. We define $\mathcal{F}_{\Delta t}$ to denote the policy function class with time interval $\Delta t$ but without the exploration.

**Definition G.2.** We define value functions corresponding to $\Delta t$ and $\delta$ for (14):

$$V_{\Delta t, \delta}(t, s, y) := \max_{\pi \in \mathcal{F}_{\Delta t, \delta}} \mathbb{E} \left\{ \int_0^T h(S_t, A_t) dt + r(S_t) \mid S_0 = s, Y_0 = y \right\},$$

$$V_{\Delta t}(t, s, y) := \max_{\pi \in \mathcal{F}_{\Delta t}} \mathbb{E} \left\{ \int_0^T h(S_t, A_t) dt + r(S_t) \mid S_0 = s, Y_0 = y \right\}.$$  

Note that the policy functions in $\mathcal{F}_{\Delta t, \delta}$ and $\mathcal{F}_{\Delta t}$ are piecewise constant over time. Therefore, Prophet indeed corresponds to $V_{\Delta t, \delta}$ instead of $V$.

**Assumption G.3.** The discrete-time value function $V_{\Delta t, \delta}$ is Lipschitz continuous in $\delta$, with $C_\delta > 0$.

**Assumption G.4.** The action space is a compact set.

**Assumption G.5.** The coefficients $f_1, f_2, g_1, g_2$ in (14), and $h$ and $r$ in the objective function, are all Lipschitz continuous in $\xi$.

#### G.2. Main Results

Prophet aims to learn the optimal policy with discrete-time interactions with the environment and exploration. In other words, the optimal performance Prophet can achieve is indeed $V_{\Delta t, \delta}$ instead of $V$. Now, we bound the difference of the two.

**Theorem G.6.** Consider the definitions and assumptions in Section G.1 and dynamics described in (14). For any $t \in [0, T]$, $s$, and $y$, there exists constant $C_{\Delta t} > 0$ and $C_\delta > 0$ such that

$$0 \leq V(t, s, y) - V_{\Delta t, \delta} \leq C_{\Delta t}(\Delta t)^{1/4} + C_\delta \delta.$$ 

Therefore, when $\Delta t$, and $\delta$ are small enough, the effect of discretization and exploration is trivial. Combined with the results in Section 5, we argue that Prophet achieves a performance close to the optimal performance of continuous-time policies.

**Proof.** The proof directly follows Lemma G.7.

**Lemma G.7** (Theorem 2.1 in Jakobsen et al. (2019)). Consider the definitions and assumptions in Section G.1 and dynamics described in (14). For any $t \in [0, T]$, $s$, and $y$, there exists constant $C_{\Delta t} > 0$ such that

$$0 \leq V(t, s, y) - V_{\Delta t} \leq C_{\Delta t}(\Delta t)^{1/4}.$$ 

Combined with Assumption G.3, we can finish the proof. \[\square\]
H. Extended Results for Theorem 5.1

H.1. Definitions and Assumptions for Theorem 5.1

Definition H.1. Define the objective function as

\[ H(\theta) := (1 - \lambda)J(\theta) + \lambda L(\theta). \]

Definition H.2 (Policy Gradient Estimation). For a dataset \( \mathcal{X} = \{\tau_n\}_{n=1}^B \) with \( B \) trajectories. We use \( \nabla_{\theta} H(\theta) := \frac{\sum_{n=1}^B \nabla H(\tau_n; \theta)}{B} \) to denote the estimated empirical gradient in Section 1, where \( \nabla H(\tau_n; \theta) \) denotes the gradient estimation built on \( \tau_n \). The detailed procedure can be found in Section 4.4.

Assumption H.3. \( \nabla_{\theta} H(\theta) \) is \( C_L \)-Lipschitz continuous.

Assumption H.4. The variance of \( \nabla_{\theta} H(\tau, \theta) \) is bounded by \( C_{\text{var}} \).

Assumption H.5. There exists constant \( C_N \) such that

\[ H(\theta^*) - H(\theta_0) \leq C_N \]

These assumptions are widely used in existing analysis (Papini et al., 2018; Karimi et al., 2019; Agarwal et al., 2021; Bhandari & Russo, 2019; Wang et al., 2019; Xu et al., 2020).

H.2. Technical Lemmas for Theorem 5.1

Lemma H.6. With Assumption H.4, and according to the definition of \( \nabla_{\theta} H(\theta) \), we have

\[ \mathbb{E}[\nabla_{\theta} H(\theta)] = \nabla_{\theta} H(\theta), \]

\[ \mathbb{E} \left[ \left\| \nabla_{\theta} H(\theta) - \nabla_{\theta} H(\theta) \right\|^2 \right] = O \left( \frac{1}{B} \right). \]

Proof. It can be seen that \( \nabla_{\theta} H(\theta) \) is the sample average of \( \nabla_{\theta} H(\theta) \). For ease of presentation:

\[ \nabla_{\theta} H(\theta) = \frac{\sum_{n=1}^B \nabla_{\theta} H(\tau_n, \theta)}{B}, \]

where \( B \) is the batch size in Algorithm 1, and \( \nabla_{\theta} h^i(\theta) \) is one sample of \( \nabla_{\theta} H(\theta) \). Therefore,

\[ \mathbb{E}[\nabla_{\theta} H(\theta)] = \nabla_{\theta} H(\theta), \]

\[ \text{Var} \left[ \nabla_{\theta} H(\theta) \right] = \frac{\text{Var} \left[ \nabla_{\theta} H(\tau_n, \theta) \right]}{B}. \]

By the Law of Total Expectation, we have

\[ \mathbb{E} \left[ \left\| \nabla_{\theta} H(\theta) - \nabla_{\theta} H(\theta) \right\|^2 \right] = \left\| \mathbb{E}[\nabla_{\theta} H(\theta)] - \nabla_{\theta} H(\theta) \right\|^2 + \text{Var} \left[ \nabla_{\theta} H(\theta) \right] = \frac{\text{Var} \left[ \nabla_{\theta} H(\tau, \theta) \right]}{B}. \]

By Assumption H.4, we finish the proof.

H.3. Proof of Theorem 5.1

Following the updating rule in Algorithm 1, we have

\[ \theta_{i+1} - \theta_i = \eta \nabla_{\theta} H(\theta_i), \]

where \( H \) and \( \nabla_{\theta} H(\theta_i) \) are defined in Definitions H.1 and H.2. We conduct a second-order Taylor expansion of \( H(\theta) \) at \( \theta_i \):

\[ H(\theta_{i+1}) - H(\theta_i) = \nabla_{\theta} H(\theta) \top (\theta_{i+1} - \theta_i) + \frac{1}{2} (\theta_{i+1} - \theta_i) \top \nabla_{\theta}^2 H(\theta') (\theta_{i+1} - \theta_i), \]
where \( \theta' := k\theta_i + (1 - k)\theta_{i+1} \) with \( k \in [0, 1] \). By Assumption H.3, we can conclude that
\[
\frac{1}{2}(\theta_{i+1} - \theta_i)\top \nabla_\theta^2 H(\theta')(\theta_{i+1} - \theta_i) \geq -C_L/2 \| \theta_{i+1} - \theta_i \|_2^2.
\]
Thus,
\[
H(\theta_{i+1}) - H(\theta_i) \geq \eta \nabla_\theta H(\theta)\top \nabla_\theta H(\theta_i) - C_L/2 \| \theta_{i+1} - \theta_i \|_2^2.
\]
(29)
Note that according to Lemma H.6,
\[ E[\nabla_\theta H(\theta)] = \nabla_\theta H(\theta). \]
Thus, the first component of the right hand side of (29)
\[
\nabla_\theta H(\theta_i)\top \nabla_\theta H(\theta_i) = - (\nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i))^\top \nabla_\theta H(\theta_i) + \nabla_\theta H(\theta_i)^\top \nabla_\theta H(\theta_i)
\]
\[
\geq - \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|^2 / 2 - \| \nabla_\theta H(\theta_i) \|^2 / 2 + \| \nabla_\theta H(\theta_i) \|^2 / 2,
\]
where the second inequality is by Cauchy inequality. Taking the result back to Equation (29),
\[
H(\theta_{i+1}) - H(\theta_i) \geq \eta \| \nabla_\theta H(\theta_i) \|^2 / 2 - \eta \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|^2 / 2 - C_L/2 \| \theta_{i+1} - \theta_i \|_2^2.
\]
Incorporating Equation (28),
\[
H(\theta_{i+1}) - H(\theta_i) \geq (\eta - C_L\eta^2) / 2 \| \nabla_\theta H(\theta_i) \|^2 / 2 - \eta \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|^2 / 2.
\]
By taking \( \eta = \frac{1}{\sqrt{C_L}} \), we get
\[
(1 - C_L/\sqrt{N})/2 \| \nabla_\theta H(\theta_i) \|^2 / 2 \leq \sqrt{N} \| H(\theta_{i+1}) - H(\theta_i) \| + \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|^2 / 2.
\]
Therefore, with \( N \) large enough, \( (1 - C_L/\sqrt{N}) \geq \frac{1}{2} \), such that
\[
\| \nabla_\theta H(\theta_i) \|^2 \leq 4\sqrt{N} \| H(\theta_{i+1}) - H(\theta_i) \| + 2 \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|^2.
\]
(30)
By applying the average technique and the expectation to both side of Equation (30), we derive
\[
\min_{i \in [N]} \mathbb{E} \left[ \| \nabla_\theta H(\theta_i) \|^2 / 2 \right] \leq \frac{\sum_{i=0}^{N-1} 4\sqrt{N} \mathbb{E} [H(\theta_{i+1}) - H(\theta_i)]}{N} + 2 \frac{\sum_{i=0}^{N-1} \mathbb{E} \left[ \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|^2 \right]}{N},
\]
\[
\leq \frac{4}{\sqrt{N}} \mathbb{E} [H(\theta_N) - H(\theta_0)] + O \left( \frac{1}{B} \right),
\]
where the last inequality is derived by Lemma H.6. Combined with Assumption H.5, we finish the proof.
I. Extended Results for Theorem 5.2

I.1. Assumptions for Theorem 5.2

Assumption I.1. We consider a slightly different estimation procedure detailed in Algorithm 2. In words, (i) the learning/optimization process is conducted in a bounded parameter space $\theta \in \mathcal{B}$, such that the value function $H(\theta)$ is locally $M$-strongly concave in $\mathcal{B}$ with a local maximal point $\theta$; (ii) the gradient of $L(\tau; \theta_i)$ at the $i^{th}$ iteration is calculated using the data collected in previous iterations.

Assumption I.2. For $\theta \in \mathcal{B}$, the variance of the derivative function can be bounded via

$$\text{Var} \left[ \nabla_\theta J(\tau, \theta_i) \right] \leq C_{\text{var}} \text{ and } \text{Var} \left[ \nabla_\theta L(\tau, \theta_i) \right] \leq C_{\text{var}}.$$  

Algorithm 2 Control-Augmented RL (CARL)

\begin{enumerate}
\item \textbf{Input:} Hyperparameter $\lambda$, learning rate $\eta$, batch size $B$, number of iterations $N$, the environment and a parameter space $\mathcal{B}$.
\item \textbf{Output:} $\pi(\cdot; \theta_P, \theta_R)$.
\item Initialize a neural network $\phi(\cdot; \theta_R)$ and accordingly the policy function $\pi(\cdot; \theta_P, \theta_R)$ by Theorem 4.1.
\item \textbf{for} $i \in [N]$ \textbf{do}
\item \hspace{1em} Generate a dataset $\mathbb{X}$ of $B$ trajectories using $\pi(\cdot; \theta_P, \theta_R)$.
\item \hspace{1em} Estimate the gradient of $J$ by Lemma 2.1 and Theorem 4.1 using the data in the current iteration.
\item \hspace{1em} Estimate the gradient of $L$ according Theorem 4.3 using the data collected in the previous iterations.
\item \hspace{1em} Combine two gradients with learning rate $\eta$, and then project the achieved update to $\mathcal{B}$.
\item \textbf{end for}
\item \textbf{Return} $\pi(\cdot; \theta_P, \theta_R)$
\end{enumerate}

I.2. Technical Lemmas for Theorem 5.2

Lemma I.3. With $\eta \leq \frac{1}{C_L}$ and Assumption H.3,

$$\sum_{i=0}^{N-1} \left\| \nabla_\theta H(\theta_i) \right\|_2^2 \leq \mathbb{E} \left[ \frac{2 (H(\theta_{N+1}) - H(\theta_0))}{\eta - C_L \eta^2} \right] + \sum_{i=0}^{N-1} \text{Var} \left( \nabla_\theta H(\theta_i) \right) \frac{1}{1 - C_L \eta}.$$  

\textbf{Proof.} By Assumption H.3, we have

$$H(\theta_{i+1}) - H(\theta_i) \geq \eta \nabla_\theta H(\theta_i) \nabla_\theta H(\theta_i) - C_L/2 \| \theta_{i+1} - \theta_i \|_2^2,$$  

(31)

where $h$ and $\nabla_\theta H(\theta_i)$ are defined in Definition H.1. By Lemma H.6,

$$\mathbb{E} \left[ \nabla_\theta H(\theta) \right] = \nabla_\theta H(\theta).$$

Therefore,

$$\nabla_\theta H(\theta_i) \nabla_\theta H(\theta_i) = -\left( \nabla_\theta H(\theta_i) - \nabla_\theta J(\theta_i) \right) \nabla_\theta H(\theta_i) + \nabla_\theta J(\theta_i) \nabla_\theta H(\theta_i)$$

$$\geq -\left\| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \right\|_2^2 / 2 - \| \nabla_\theta H(\theta_i) \|_2^2 / 2 + \left\| \nabla_\theta H(\theta_i) \right\|_2^2 / 2$$

$$= -\left\| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \right\|_2^2 / 2 + \left\| \nabla_\theta H(\theta_i) \right\|_2^2 / 2.$$

With Equation (31),

$$H(\theta_{i+1}) - H(\theta_i) \geq (\eta - C_L \eta)^2 / 2 \left\| \nabla_\theta H(\theta_i) \right\|_2^2 - \eta \| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|_2^2 / 2.$$  

If $\eta \leq \frac{1}{C_L}$,

$$\frac{2 (H(\theta_{i+1}) - H(\theta_i))}{\eta - C_L \eta^2} + \frac{\| \nabla_\theta H(\theta_i) - \nabla_\theta H(\theta_i) \|_2^2}{1 - C_L \eta} \geq \left\| \nabla_\theta H(\theta_i) \right\|_2^2.$$
Next, we take the expectation on both sides, and derive
\[
E \left[ \| \nabla_\theta H(\theta_i) \|^2 \right] \leq E \left[ \frac{2 (H(\theta_{i+1}) - H(\theta_i))}{\eta - C_L \eta^2} \right] + \frac{\text{Var} [ \nabla_\theta H(\theta_i) ]}{1 - C_L \eta}.
\]

**Lemma I.4.** Say \( H \) is strongly convex with \( M \), then
\[
|H(\theta) - H(\tilde{\theta})| = (\theta - \tilde{\theta})^T \frac{\nabla^2 h(\tilde{\theta})}{2} (\theta - \tilde{\theta}) \geq \frac{M}{2} \| \theta - \tilde{\theta} \|^2.
\]

**Lemma I.5.** The performance objective function is Lipschitz continuous with \( C_L > 0 \):
\[
(J(\theta) - J(\theta'))^2 \leq C_L \| \theta - \theta' \|^2.
\]

**Lemma I.6.** Under Assumption I.2, the summation of the variance of the gradient estimation can be bounded by
\[
\sum_{i=1}^{N} \text{Var} (\nabla_\theta H(\theta_i)) \leq \frac{(1 - \lambda)^2 C_{\text{var}} N}{B} + \frac{\lambda^2 C_{\text{var}} (1 + \log(N))}{B}.
\]

**Proof.** The variance of the estimated gradient satisfies
\[
\text{Var} (\nabla_\theta H(\theta_i)) = \frac{(1 - \lambda)^2 \text{Var} (\nabla_\theta J(\tau, \theta_i))}{B} + \frac{\lambda^2 \text{Var} (\nabla_\theta L(\tau, \theta_i))}{B(i - 1)} \leq \frac{(1 - \lambda)^2 C_{\text{var}} N}{B} + \frac{\lambda^2 C_{\text{var}} (1 + \log(N))}{B(i - 1)}.
\]

Note that, according to Algorithm 2, the gradient of \( L \) at the \( i \)-th iteration uses the data in the past \( (i - 1) \) iterations, which accounts for the \( i - 1 \) in the last component of (32). Next, we sum over \( i \):
\[
\sum_{i=1}^{N} \text{Var} (\nabla_\theta H(\theta_i)) \leq \frac{(1 - \lambda)^2 C_{\text{var}} N}{B} + \frac{\lambda^2 C_{\text{var}} (1 + \log(N))}{B}.
\]

**Lemma I.7.** Let \( \theta_i \) be the estimated \( \theta \) after the \( i \)-th iteration of Algorithm 2. Define \( \bar{\theta} = \frac{\sum_{i=0}^{N} \theta_i}{N} \). With Assumption I.1, we have
\[
H(\bar{\theta}) - E[H(\bar{\theta})] \leq \frac{\| \theta_0 - \bar{\theta} \|^2}{2 \eta N} + \frac{\eta}{2N} \left[ \frac{2 (H(\theta_N) - H(\theta_0))}{\eta - C_L \eta^2} \right] + \frac{\eta (1 - \lambda)^2 C_{\text{var}}}{2B} + \frac{\eta \lambda^2 C_{\text{var}} (1 + \log(N))}{2BN}.
\]

**Proof.** By the strong concavity of \( h \) in Assumption I.1,
\[
\nabla_\theta H(\theta_i) (\tilde{\theta} - \theta_i) \geq H(\tilde{\theta}) - H(\theta_i) + \frac{m}{2} \| \theta_i - \tilde{\theta} \|^2.
\]
Further, since \( \theta_{i+1} \) is the projection of \( \theta_i + \eta \nabla_\theta H(\theta_i) \) to \( B \), the projection satisfies
\[
\| \theta_i + \eta \nabla_\theta H(\theta_i) - \tilde{\theta} \|^2 \geq \| \theta_{i+1} - \tilde{\theta} \|^2.
\]
We take expectations on both sides of Equation (35) and derive:

\[
\|\theta_i - \hat{\theta}\|^2 - \|\theta_{i+1} - \hat{\theta}\|^2 \geq \|\theta_i - \hat{\theta}\|^2 - \|\theta_i + \eta \nabla_{\theta} H(\theta_i) - \hat{\theta}\|^2 \\
= -\eta \nabla_{\theta} H(\theta_i)^T (2\theta_i + \eta \nabla_{\theta} H(\theta_i) - 2\hat{\theta}) \\
= -\eta^2 \|\nabla_{\theta} H(\theta_i)\|^2 - 2\eta (\theta_i - \hat{\theta})^T \nabla_{\theta} H(\theta_i).
\]  

(34)

We reorder Equation (34) and derive

\[
-(\theta_i - \hat{\theta})^T \nabla_{\theta} H(\theta_i) \leq \frac{1}{2\eta} \left(\|\theta_i - \hat{\theta}\|^2 + \eta^2 \|\nabla_{\theta} H(\theta_i)\|^2 - \|\theta_{i+1} - \hat{\theta}\|^2\right).
\]

Taking the result back to Equation (33):

\[
H(\hat{\theta}) - H(\theta_i) \leq (\nabla_{\theta} H(\theta_i) - \nabla_{\theta} H(\theta_i))(\hat{\theta} - \theta_i) \\
+ \frac{1}{2\eta} \left(\|\theta_i - \hat{\theta}\|^2 - \|\theta_{i+1} - \hat{\theta}\|^2 + \eta^2 \|\nabla_{\theta} H(\theta_i)\|^2\right) - \frac{m}{2} \|\theta_i - \hat{\theta}\|^2,
\]

(35)

where in the last inequality, we omit \( \frac{m}{2} \|\theta_i - \hat{\theta}\|^2 \) for an upper bound.

By Lemma H.6, we have

\[
E \left((\nabla_{\theta} H(\theta_i) - \nabla_{\theta} H(\theta_i))(\hat{\theta} - \theta_i)\right) = 0.
\]

We take expectations on both sides of Equation (35) and derive:

\[
H(\hat{\theta}) - H(\theta_i) \leq \frac{1}{2\eta} \|\theta_i - \hat{\theta}\|^2 - \frac{1}{2\eta} \|\theta_{i+1} - \hat{\theta}\|^2 + \frac{1}{2} \eta \|\nabla_{\theta} H(\theta_i)\|^2.
\]

By averaging over \( i \), we get

\[
H(\hat{\theta}) - E[H(\theta_i)] \leq \frac{\|\theta_0 - \hat{\theta}\|^2}{2\eta N} + \frac{\eta}{2N} \sum_{i=0}^{N-1} E \left[\|\nabla_{\theta} H(\theta_i)\|^2\right].
\]

Finally, we apply Lemma I.3, and Lemma I.6

\[
H(\hat{\theta}) - E[H(\theta_i)] \leq \frac{\|\theta_0 - \hat{\theta}\|^2}{2\eta N} + \frac{\eta}{2N} E \left[\frac{2 (H(\theta_{N+1}) - H(\theta_0))}{\eta - C_L \eta^2}\right] + \frac{\eta}{2N} \sum_{i=0}^{N} \text{Var} \left(\nabla_{\theta} H(\theta_i)\right) \frac{1}{1 - C_L \eta}
\]

\[
\leq \frac{\|\theta_0 - \hat{\theta}\|^2}{2\eta N} + \frac{\eta}{2N} E \left[\frac{2 (H(\theta_{N+1}) - H(\theta_0))}{\eta - C_L \eta^2}\right]
\]

\[
+ \frac{\eta(1 - \lambda)^2 C_{\text{var}}}{2B} + \frac{\eta\lambda^2 C_{\text{var}} (1 + \log(N))}{2BN}.
\]

I.3. Proof of Theorem 5.2

To start with, we leverage the convergence result for Algorithm 2 detailed by Lemma I.7:

\[
H(\hat{\theta}) - E[H(\theta_i)] \leq \frac{\|\theta_0 - \hat{\theta}\|^2}{2\eta N} + \frac{\eta}{2N} E \left[\frac{2 (H(\theta_{N+1}) - H(\theta_0))}{\eta - C_L \eta^2}\right] + \frac{\eta(1 - \lambda)^2 C_{\text{var}}}{2B}
\]

\[
+ \frac{\eta\lambda^2 C_{\text{var}} (1 + \log(N))}{2BN}.
\]

(36)
Combined with Lemma I.4, we can further bound $\| \hat{\theta} - \tilde{\theta} \|^2$:

$$\| \hat{\theta} - \tilde{\theta} \|^2 \leq \frac{2}{m} \left| H(\hat{\theta}) - E[H(\tilde{\theta})] \right|$$

$$\leq \frac{1}{mN} \left( 2 \left( H(\theta_{N+1}) - H(\theta_0) \right) \right) + \frac{\eta(1 - \lambda)^2 C_{\text{var}}}{mB}$$

$$+ \frac{\eta\lambda^2 C_{\text{var}}(1 + \log(N))}{mBN},$$

where the first inequality is by Lemma I.4, and the second inequality is by Equation (36).

Finally, we apply Lemma I.5

$$\mathbb{E} [(J(\theta^*) - J(\tilde{\theta}))^2] \leq \frac{C_L}{m} \left[ \frac{\theta_0 - \tilde{\theta}}{m\eta N} \right]^2 + \mathbb{E} \left[ \frac{2C_L}{m\eta N(1 - C_L\eta)} \left( H(\theta_{N+1}) - H(\theta_0) \right) \right]$$

$$+ \frac{C_L \eta(1 - \lambda)^2 C_{\text{var}}}{mB}$$

$$+ \frac{C_L \eta\lambda^2 C_{\text{var}}(1 + \log(N))}{mBN}.$$  (37)

When $\eta$ is small enough with $B, N$ large enough, there exists $C > 0$ such that

$$\mathbb{E} [(J(\theta^*) - J(\tilde{\theta}))^2] \leq \frac{C(1 + \eta)}{\eta N} + \frac{C\eta(1 - \lambda)^2}{B} + \frac{C\eta\lambda^2(1 + \log(N))}{BN}.$$

One the other hand, by the definition of $h$, we have

$$0 \leq J(\theta^*) - J(\tilde{\theta}) \leq \frac{H(\theta^*) - H(\tilde{\theta})}{1 - \lambda}. $$  (38)

Combining Equations (37) and (38), we have

$$\mathbb{E} [(J(\theta^*) - J(\tilde{\theta}))^2] \leq \left( \frac{C}{B} + \frac{C\eta(1 - \lambda)^2}{BN} \right) \left( I \right) + \frac{C(1 + \eta)}{\eta N} \left( II \right).$$

I.4. Extended Discussions for Theorem 5.2

To start with, we prove Lemma I.8 for the effect of $\lambda$ on the proved upper bound.

**Lemma I.8.** If $B \leq \frac{\eta C}{H(\theta^*) - H(\tilde{\theta})}$, the upper bound in Theorem 5.2 is minimized by a $\lambda \in (0, 1)$.

**Proof.** It should be noticed that $\lambda = 1$ does not minimize the upper bound of Theorem 5.2. Therefore, the minimal point is either $\lambda = 0$ or some value $\lambda \in (0, 1)$.

Further, we take derivative of the upper bound:

$$\frac{\partial}{\partial \lambda} \left[ \frac{\eta(1 - \lambda)^2 C}{B} + \frac{C\eta\lambda^2(1 + \log(N))}{BN} + \frac{C(1 + \eta)}{\eta N} \right] \bigg|_{\lambda=0} < 0.$$

Therefore, there exists $\lambda \in (0, 1)$ minimizing the upper bound.

Lemma I.8 suggests that adding the prediction objective is helpful if $B$ is not too big. This makes sense: given a huge $B$, we would have sufficient data for a classic PG method and the sample complexity benefit of the prediction objective does not exist.
Intuitively, our algorithm seeks to find a policy that balances between performing well and predicting well. On the one hand, optimizing for \( L \) helps the learning algorithm to avoid policies that perform well on the training set (high \( J \)) but that provide poor explanation for the data generation process (low \( L \)). On the other hand, optimizing for \( J \) helps to avoid policies that appear to have a good explanation for the data generation process (high \( L \)) but perform poorly on the training data (low \( J \)).

Note that the optimal \( \lambda \) minimizing the **upper bound** in Theorem 5.2 depends on the specific function form of \( J(\theta) \) and \( L(\theta) \), more specifically the value of \( (H(\theta^*) - H(\hat{\theta}))^2 \). Bounding \( H(\theta^*) - H(\hat{\theta}) \) under different assumptions is indeed a key theoretical challenge for non-convex optimization, indicating the difference between the local optimal values and the global optimal values. There are different settings other than the one in Section 5.2, which correspond to different optimal \( \lambda \)'s:

- \( H(\theta^*) - H(\hat{\theta}) = 0 \) (Polyak, 1963; Bhandari & Russo, 2019): Under specific conditions, one can argue the local optimal value is equivalent to global optimal values. In that sense, (I) would dominate (II) (which is zero). Since MLE theoretically converges to a local optimal value more efficiently (asymptotic efficiency), the optimal \( \lambda \) is 1.

- \( (H(\theta^*) - H(\hat{\theta}))^2 = c > 0 \): Without further assumptions, \( (H(\theta^*) - H(\hat{\theta}))^2 \) cannot be bounded and will not even decay over \( B \) or \( N \). As a result, (II) would dominate (I), suggesting the optimal \( \lambda \) as 0.

- Decaying \( (H(\theta^*) - H(\hat{\theta}))^2 \): As the middle case between the two extreme ones, Wang et al. (2019) provides settings where \( H(\theta^*) - H(\hat{\theta}) \) decays to 0 with a fast speed regarding \( N \). Therefore, the optimal \( \lambda \) should balance (I) and (II), and lies in \((0, 1)\).

More specifically, if \( H(\theta^*) - H(\hat{\theta}) \leq O(N^{-3}) \) as shown under a series of assumptions (Wang et al., 2019), \( \lambda \in (0, 1) \) satisfying \( \left( \frac{\lambda}{1-\lambda} \right)^2 = O\left( \frac{N}{1+\log(N)} \right) \) provides a faster converging upper bound: \( \mathbb{E}[ (J(\theta^*) - J(\hat{\theta}))^2 ] \leq O\left( \frac{\log(N)^{1/4}}{N} \right) \).

Empirically, we treat \( \lambda \) as a hyperparameter, and conduct tuning for specific problems.
J. Extended Results for Synthetic Experiments

In this section, we provide extended results for synthetic experiments on NSFM-LQC (Appendix J.1), and synthetic experiments on Merton, NSFM-Kim–Omberg and NSFM-EVE (Appendix J.2).

J.1. Synthetic Experiments for NSFM-LQC

For synthetic NSFM-LQC we provide details about the protocol (Appendix J.1.1), related works (Appendix J.1.2), hyperparameter tuning, (Appendix J.1.3) implementation of competing methods (Appendix J.1.4), and extra experiments (Appendix J.1.5).

J.1.1. Protocols

We simulate three linear-quadratic environments with 5, 10, and 15 state variables and factors each, yielding a total dimension of 10, 20, and 30 respectively. For each method, we vary the learning rate and other method-specific hyperparameters. For each method-hyperparameter-environment combination, we repeat our training and testing procedures ten times. After training a policy for LQC, we deploy the trained policy in an environment, collect trajectories, and compute their total reward.

The data generation procedure follows:

1. Let $p \in \{5, 10, 15\}$, $\Delta t = 0.01$ and $T = 0.5$. In other words, each trajectory consists of 50 steps.
2. Randomly generate the matrices by
   \[
   B := I_p + 0.1 \epsilon_B, \\
   U := I_p + 0.1 \epsilon_U, \\
   D := 0.1 I_p, \\
   \mu := 1.4 I_p + 0.1 \epsilon_\mu, \\
   \nu := 0.1 I_p, \\
   Q := G := -10 I_p,
   \]
   with $I_p$ as the $p$-dimensional identity. \(\epsilon\)'s are random matrices whose components follow IID standard normal distributions.
3. Randomly generate $S_0 := 1_p + N(0, I_p)$, and $X_0 := 1_p + N(0, I_p)$.
4. Let $\phi(\cdot)$ be the component-wise exponential function, and thus generate $Y_0 = \phi(X_0)$.
5. Given an action $A$, generate the next step $S$ and $X$ by conditional distribution following an Euler discretization.
6. Repeat Step 5 for 50 steps to finish one trajectory and then collect the total reward.

After training a policy for LQC, we deploy the trained policy in an environment, collect trajectories, and compute their total reward.

J.1.2. Related Works

While not studying factors, methods leveraging the LQC setup exist in both control and and RL. LQC is first proposed and rigorously studied in the control community. The solutions follow a two-step procedure: first fit a model and then drive a policy by control tools (Sun & Yong, 2020). However, the first step requires deriving the likelihood of an SDE model, and the second step often ends up with solving a partial differential equation (PDE): both steps are challenging under a high-dimensional setting. From the RL side, efforts have been made to replace the two steps of optimal control by a specifically designed RL method (Bradtke, 1993; Bu et al., 2020). Such methods avoid calculating the likelihood and solving PDEs. Compare with model-free RL, these methods have stronger theoretical guarantees since the data generation process is assumed to strictly follow LQC (Basei et al., 2020). Further, efforts have been made to incorporate part of an LQC model in to RL, aiming to improve the sample efficiency (Fernandez et al., 2020; Li & Todorov, 2004; Tu & Recht, 2020).
CARL: Control-Augmented Reinforcement Learning for Decision-Making with Stochastic Factors

2018), robustness (Umenberger et al., 2019) and theoretical properties (Fazel et al., 2018). Our method differ from these methods in two perspectives. First of all, we explicitly model the factors and thus can provide better sample efficiency. Second, none of the existing methods jointly conduct policy learning and model fitting like CARL. We demonstrate the improved performances of such a combination both theoretically and empirically.

J.1.3. Hyperparameter Selection

To select the hyperparameters, we use the achieved total rewards as the validation set. Specifically, for each method, the hyperparameter achieving the highest average of total rewards are selected. With the selected hyperparameters, we retrain the methods with 10-time replications, and report the average of total rewards in Table 2.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Iterations $N$</td>
<td>1000</td>
</tr>
<tr>
<td>Time Interval $\Delta t$</td>
<td>0.01</td>
</tr>
<tr>
<td>Terminal Time $T$</td>
<td>0.5</td>
</tr>
<tr>
<td>Considered Learning Rates $\eta$</td>
<td>${0.3, 0.2, 0.1, 0.05, 0.01, 0.005, 0.001}$</td>
</tr>
<tr>
<td>Considered $\lambda$’s</td>
<td>${0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1}$</td>
</tr>
<tr>
<td>Number of Simulated Trajectories $B$</td>
<td>${50, 100}$</td>
</tr>
<tr>
<td>Number of Replications under Each Hyperparameter</td>
<td>10</td>
</tr>
<tr>
<td>Compute Resources</td>
<td>AWS ec2 m5ad.24xlarge</td>
</tr>
</tbody>
</table>

Table 5: Hyperparameters for synthetic experiments

J.1.4. Implementation Details for Competing Methods

We now provide more details regarding the competing methods.

- PG belongs to the category of model-free RL (Sutton et al., 1999; Sutton & Barto, 2018). PG does not explicitly learn the representation of stochastic factors, or leverage the SDE structure.

- SDE-PG is an RL method developed specifically for LQC, and thus falls into the category of RL for LQC (Wang et al., 2018b; Basei et al., 2020; Bu et al., 2020), which leverages the structure of SDEs. Specifically, under the framework of PG, LQC-PG uses the policy function form derived from the SDEs to parameterize the policy (also derived in Lemma F.1). LQC-PG does not consider stochastic factors.

- E2C first learns a locally-linear representation of the state variables, and then solve the decision-making as a discrete-time linear-quadratic control on the learned representations. The original E2C method is designed for a Bayesian setting, where $X_t | Y_t$ is random and the variational autoencoder is leveraged. However, by NSFM, $X_t = \phi(Y_t)$ is deterministic given $Y_t$ in the simulated dynamics. Therefore, to improve the performance of E2C and strike a fair comparison, we replace the variational autoencoder by a neural network representing $\phi$.

J.1.5. Extra Results

Results with 10 and 30 variables are reported in Figure 6.

J.2. Synthetic Experiments for Merton, NSFM-Kim–Omberg and NSFM-EVE

Merton, NSFM-Kim–Omberg and NSFM-EVE are models for portfolio optimization. For synthetic portfolio optimization, we provide details for protocol (Appendix J.2.1), data generation (Appendix J.2.2), extra penalty (Appendix J.2.3), and hyperparameter tuning (Appendix J.2.4).
CARL: Control-Augmented Reinforcement Learning for Decision-Making with Stochastic Factors

Figure 6: Total rewards for all configurations in histogram and boxplot

J.2.1. Protocol

For synthetic portfolio optimization, we consider three different models: NSFM-Kim–Omberg (Appendix D.3), Merton (Appendix D.2), and NSFM-EVE (Appendix D.4). Merton model considers a simplified setting with constant drifts for asset prices without factors, and is one of the most classical settings for portfolio optimization. Note that we still generate fake features unrelated to stock prices as the input for CARL. The hypothesis is that CARL would not be affected by the fake features and provide decent performances. EVE model is much more complicated, where factors also affect the volatility of asset prices. EVE model belongs to stochastic volatility problems which are notoriously challenging. For each setting, we build a synthetic environment with 5 assets with 5 features, following the SDE of the corresponding model. We consider 21-day trading, and generate 5000 trajectories for training, 5000 for validation, and 5000 for testing.

J.2.2. Data Generation

To generate synthetic environments, the drifts and the volatility are randomly picked but mimicking the historical stock price data, with an average annual return around 0.1 and average annual volatility around 0.15. Thus the daily return is around $0.1 \times \sqrt{252}$, and the daily volatility is around $0.15 \times \sqrt{252}$. The true representation function is selected as component-wise exponential operation. To generate data, we consider log prices, whose SDE follows the form of Lemma F.11. Then, we discretize the SDE following the explicit Euler method like (18), and generate data accordingly.

The specific configurations for data generation are provided below.

- Merton (Appendix D.2):
  - $\sigma$ is selected as a random matrix, whose components follow a uniform distribution in $[0.5 \times \frac{0.15}{\sqrt{252}}, 1.5 \times \frac{0.15}{\sqrt{252}}]$.
  - $v$ is selected as a random matrix, whose components follow a uniform distribution in $[-1.5 \times \frac{0.1}{\sqrt{252}} \times 5, 1.5 \times \frac{0.1}{\sqrt{252}} \times 5]$.
  - $\mu$ is selected as a diagonal matrix whose diagonal components follow a uniform distribution in $[0.9, 1]$.
  - The initial values of $\mathbf{X}$ are randomly generated from a uniform distribution on $[-2 \times \frac{0.1}{\sqrt{252}}, 2 \times \frac{0.1}{\sqrt{252}}]$.

Methods

Prophet
E2C
LQC−PG
PG
The initial values of asset prices are randomly generated from a uniform distribution in $[20, 30]$.

- **NSFM-Kim–Omberg (Appendix D.3):**
  - $\sigma$ is selected as a random matrix, whose components follow a uniform distribution in $[0.5 \times \frac{0.15}{\sqrt{252}}, 1.5 \times \frac{0.15}{\sqrt{252}}]$.
  - $\nu$ is selected as a random matrix, whose components follow a uniform distribution in $[-1.5 \times \frac{0.15}{\sqrt{252}} \times 5, 1.5 \times \frac{0.15}{\sqrt{252}} \times 5]$.
  - $\mu$ is selected as a diagonal matrix whose diagonal components follow a uniform distribution in $[0.9, 1]$.
  - The initial values of $X$ are randomly generated from a uniform distribution on $[-2 \times \frac{0.15}{\sqrt{252}}, 2 \times \frac{0.15}{\sqrt{252}}]$.
  - The initial values of asset prices are randomly generated from a uniform distribution in $[20, 30]$.

- **NSFM-EVE (Appendix D.4):**
  - $D_\sigma$ is a diagonal matrix whose diagonal components follow a uniform distribution on $[0.5 \times \frac{0.15}{100 \times \sqrt{252}}, 1.5 \times \frac{0.15}{100 \times \sqrt{252}}]$.
  - $D_\mu$ is a diagonal whose diagonal components follow a uniform distribution $[0.5 \times \frac{0.15}{\sqrt{252}} \times \frac{1}{100}, 1 \times \frac{0.15}{\sqrt{252}} \times \frac{1}{100}]$.
  - $M$ is a diagonal matrix, whose diagonal components follow a uniform distribution on $[-1, -0.9]$.
  - $\omega$ is a vector whose components follow a uniform distribution on $[0.5 \times 100, 0.1 \times 100]$.
  - $D_a$ is a diagonal matrix, whose diagonal components follow a uniform distribution on $[5 \times 0.5 \times \frac{0.15}{\sqrt{252}} \times \frac{1}{100}, 5 \times \frac{0.15}{\sqrt{252}} \times \frac{1}{100}]$.
  - The initial values of asset prices are randomly generated from a uniform distribution in $[0.1 \times \sqrt{252}, 2 \times \frac{0.15}{\sqrt{252}} \times \sqrt{252}]$.
  - The initial values of $X_t$ are randomly generated from a uniform distribution in $[\frac{0.15}{252} \times 100, 2 \times \frac{0.15}{252} \times 100]$.

Note that the design makes sure that the simulated price has approximately a yearly return as 0.1 and yearly volatility as 0.15.

### J.2.3. Extra Penalty

In practice, we add one penalty term for the portfolio optimization. Consider a trajectory with time interval $\Delta t$, $\tau := \{t_i, s_{t_i}, x_{t_i}, a_{t_i} \mid i \in [m]\}$. The penalty is defined as

$$
\delta \min_{C, b} \sum_{i=1}^{m} \left\| \phi(s_{t_i+1}) - C \phi(s_{t_i}) - b \right\|_2^2,
$$

where $C$ is a matrix and $b$ a vector of proper dimensions. We tune the hyperparameter $\delta > 0$ for the effect of the penalty. The intuition of (39) comes from the transition of $X_t$ in (4). We achieved improved performance when tuning $\delta$ in practice.

### J.2.4. Early Stopping and Hyperparameter Tuning

For better performance, we conduct early stopping for all the considered methods using the total utility per trajectory on the validation set with the patience as 5 steps. But for CARL, we also incorporate the prediction objective into the stopping strategy. More specifically, we update the saved model only when both the achieved validation utility gets higher, and the validation prediction loss gets smaller. The considered hyperparameters include the learning rate, $\lambda$, batch size, penalty parameter $\delta$, and $\lambda_X$. For each configuration, we conduct training for 10 times, and average the results. Then, we pick the configuration providing the best the average validation utility, and test it on the test data for the average test utility per trajectory. The tuning process is conducted using the software wandb (Biewald, 2020). We report the considered and the finally picked values in Table 6, 7 and 8.

### K. Extended Results of Real-World Stock Trading

#### K.1. Protocol

We consider 21-day stock trading in 3 different stock sectors using the daily stock price data from Yahoo finance between January 4, 2006 and April 1, 2020. More specifically, we use the adjusted close price as the daily trading price. For factors, we consider economic indexes, technical analysis indexes (generated by python package TA), and sector-specific features such as oil prices, gold prices, and related ETF prices, leading to around 30 factors for each sector. In each sector we select 10 stocks according to the availability and trading volume in the considered time range. The considered sectors, stocks, and the features are provided in Table 9.
CARL: Control-Augmented Reinforcement Learning for Decision-Making with Stochastic Factors

### Table 6: Hyperparameters of CARL for synthetic portfolio optimization

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Considered Values</th>
<th>Selected Values</th>
<th>Kim–Omberg Model</th>
<th>Merton Model</th>
<th>EVE Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>{0.0005, 0.001, 0.01, 0.1}</td>
<td>0.01</td>
<td>0.1</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>Batch Size</td>
<td>{100, 50}</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>$\lambda$</td>
<td>{0, 0.1, 0.5, 0.9, 1}</td>
<td>0.5</td>
<td>0.9</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>$\delta$</td>
<td>{0, 0.1, 0.5, 1}</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

### Table 7: Hyperparameters of PG for synthetic portfolio optimization

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Considered Values</th>
<th>Selected Values</th>
<th>Kim–Omberg Model</th>
<th>Merton Model</th>
<th>EVE Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>{0.0005, 0.001, 0.01, 0.1}</td>
<td>0.01</td>
<td>0.01</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>Batch Size</td>
<td>{200, 100, 50}</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
</tbody>
</table>

### K.2. Hyperparameter Tuning

The training, validation, testing data are constructed using rolling windows. Specifically, we take the first 1260-day (five years) data as the training data, the next 128 days (six months) for validation, and another 128 days following for testing. The considered hyperparameters and the selected ones are summarized in Table 10, 11 and 12.
### Table 8: Hyperparameters of SDE-PG for synthetic portfolio optimization

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Considered Values</th>
<th>Selected Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>{0.0005, 0.001, 0.01, 0.1}</td>
<td>0.01 0.01 0.01</td>
</tr>
<tr>
<td>Batch Size</td>
<td>{200, 100, 50}</td>
<td>50 100 50</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.1</td>
<td>0.1 0.1 0.1</td>
</tr>
</tbody>
</table>

Table 9: Selected stocks and features

<table>
<thead>
<tr>
<th>Sectors</th>
<th>Stocks</th>
<th>Features for Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td>APA, COP, CVX, HAL, HES, MRO, OKE, OXY, VLO, WMB</td>
<td>SP500 returns, MACD of stock prices, RSI of stock prices, oil prices, gasoline prices, US Dollar/USDX - Index - Cash (DX-Y.NYB)</td>
</tr>
<tr>
<td>Industrial</td>
<td>BA, CAT, DE, EMR, ETN, GE, HON, LMT, LUV, PNR,</td>
<td>SP500 returns, MACD of stock prices, RSI of stock prices, ETF prices including DIA, EXI, IYJ and VIS</td>
</tr>
<tr>
<td>Materials</td>
<td>APD, AVY, BLL, DD, ECL, FMC, IFF, IP, NEM, VMC</td>
<td>SP500 returns, MACD of stock prices, RSI of stock prices, gold prices, silver prices, ETF prices including IYM and VAW</td>
</tr>
</tbody>
</table>

### L. Extended Results for Robust Analysis

To simulate data for NSFM-SIR (Appendix D.5), the coefficients are selected as the estimated values of COVID-19 using the data of Algeria in Lounis & Bagal (2020). The configuration follows:

- The true representation function $\phi$ is selected to be the component-wise exponential operation.
- $v$ is selected to be a $2 \times 2$ random matrix with the first-row components following a uniform distribution in $[-1.5 \times 0.04553 \times 0.5, 1.5 \times 0.04553 \times 0.5]$, and the second-row components following a uniform distribution in $[-1.5 \times 0.0561215 \times 0.5, 1.5 \times 0.0561215 \times 0.5]$.
- $M$ is a $2 \times 1$ random matrix with first-row components following a uniform distribution in $[0.8 \times 0.04553, 0.9 \times 0.04553]$, and second-row components following a uniform distribution in $[0.8 \times 0.0561215, 0.9 \times 0.0561215]$.
- $\mu$ is a diagonal matrix with diagonal components following a uniform distribution in $[0.9, 1]$.
- The initial value of $X_t$ is $[0.5 \times 0.04553, 0.5 \times 0.0561215]$.
- $S_0 = 40000000$, and $I_0 = 36669$, as suggested by Lounis & Bagal (2020).
- Such a design makes sure that $X_t$ fluctuates around $[0.04553, 0.0561215]$, which is estimated by Lounis & Bagal (2020).

### L.1. Sensitivity Analysis

Under the setting of Figure 3, we implement Prophet with different $\lambda$’s. For each $\lambda$, we use the same set of hyperparameters with 10 replications. Figure 7 plots the average of the total rewards for each $\lambda$. Most of $\lambda \in (0, 1)$ provide better performance than $\lambda = 0$ or $\lambda = 1$, which is consistent with our analysis in Section 5.2: combining the objectives can achieve better performances than solely using either one.
### Table 10: Hyperparameters of CARL for real-world stock trading

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Considered Values</th>
<th>Selected Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>{0.0005, 0.001, 0.01, 0.1}</td>
<td>0.01 0.1 0.01</td>
</tr>
<tr>
<td>Batch Size</td>
<td>{200, 400}</td>
<td>400 200 50</td>
</tr>
<tr>
<td>(\lambda)</td>
<td>{0, 0.1, 0.5, 0.9, 1}</td>
<td>0.5 0.9 0.1</td>
</tr>
<tr>
<td>(\delta)</td>
<td>{0, 0.1, 0.5, 1}</td>
<td>0 0.5 1</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>0.1</td>
<td>0.1 0.1 0.1</td>
</tr>
</tbody>
</table>

### Table 11: Hyperparameters of PG for real-world stock trading

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Considered Values</th>
<th>Selected Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>{0.0005, 0.001, 0.01, 0.1}</td>
<td>0.001 0.01 0.1</td>
</tr>
<tr>
<td>Batch Size</td>
<td>{200, 400, 100, 50}</td>
<td>100 100 200</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>0.1</td>
<td>0.1 0.1 0.1</td>
</tr>
</tbody>
</table>

### Table 12: Hyperparameters of SDE-PG for real-world stock trading

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Considered Values</th>
<th>Selected Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>{0.0005, 0.001, 0.01, 0.1}</td>
<td>0.01 0.01 0.1</td>
</tr>
<tr>
<td>Batch Size</td>
<td>{200, 400, 100, 50}</td>
<td>400 200 400</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>0.1</td>
<td>0.1 0.1 0.1</td>
</tr>
</tbody>
</table>

**Figure 7:** Sensitivity Analysis on \(\lambda\)