A Machine Learning Approach to Adaptive Robust Utility Maximization and Hedging

Tao Chen* and Michael Ludkovski†

Abstract. We investigate the adaptive robust control framework for portfolio optimization and loss-based hedging under drift and volatility uncertainty. Adaptive robust problems offer many advantages but require handling a double optimization problem (infimum over market measures, supremum over the control) at each instance. Moreover, the underlying Bellman equations are intrinsically multi-dimensional. We propose a novel machine learning approach that solves for the local saddle-point at a chosen set of inputs and then uses a nonparametric (Gaussian process) regression to obtain a functional representation of the value function. Our algorithm resembles control randomization and regression Monte Carlo techniques but also brings multiple innovations, including adaptive experimental design, separate surrogates for optimal control and the local worst-case measure, and computational speed-ups for the sup-inf optimization. Thanks to the new scheme we are able to consider settings that have been previously computationally intractable and provide several new financial insights about learning and optimal trading under unknown market parameters. In particular, we demonstrate the financial advantages of adaptive robust framework compared to adaptive and static robust alternatives.

Key words. adaptive robust control, Gaussian process surrogates, portfolio optimization

1. Introduction.

Stochastic control formulations have been a core tool in financial mathematics for over 40 years. One fundamental challenge in applying stochastic models to practice is the issue of model risk, i.e. calibrating system dynamics to real life. In this article by "model risk" we mean uncertainty about the underlying probability measure $\mathbb{Q}$ representing the mis-specification between the assumed dynamics and the true probabilistic structure. More precisely, the uncertainty concerns some parameters θ, so that we have the parameterization $\mathbb{Q} = \mathbb{Q}^\theta$. Parameter uncertainty arises in any application, since the required model calibration necessarily leaves some residual inference error about the true parameter values. For example, in problems involving risky assets such as stocks, asset return μ and volatility σ are the two main parameters $\theta \equiv (\mu, \sigma)$ driving the investment decisions, but are notoriously difficult to calibrate.

As one remedy, there have been recently multiple proposals on robust extensions for the underlying stochastic control formulation. The basic strategy is to take a worst-case view among a collection of potential $\mathbb{Q}^\theta$'s. In the financial context, this conservative perspective stemming from unknown parameters is tempered by the idea of *learning* the dynamics. Historical data about the risky asset evolution can often be used to improve estimates about its drift and/or volatility. Combining these two concepts of robustness and learning, we adopt the *adaptive robust* framework recently proposed in Bielecki et al. [9]. This paradigm elegantly connects *static robust* approaches that do not allow belief updating with *adaptive* control that fully incorporates the latest estimate within a fixed-parameter setup. Because adaptive con-

*Department of Mathematics, University of Michigan, 530 Church St, Ann Arbor, MI 48109 (chenta@umich.edu).
†Department of Statistics and Applied Probability, University of California Santa Barbara, 5520 South Hall, Santa Barbara, CA 93106-3110 (ludkovski@pstat.ucsb.edu).

trol ignores dynamic updating, it is time-inconsistent. However, adaptive robust control also presents formidable numerical challenges as it features an expanded, multi-dimensional state space and non-trivial nonlinear optimization to find the optimal feedback control $u^*(t,x)$ and the value function $V(t,x)$. As a result, closed form solutions are ruled out, and standard PDE methods are not feasible beyond the most simple settings.

In this paper we propose and develop a novel algorithm for adaptive robust control. Our approach belongs to the class of Regression Monte Carlo (RMC) and Control Randomization (CR) strategies [10, 5, 12, 14, 26]. The key idea is to recursively construct a functional approximation $\hat{V}(t,\cdot)$ to $V(t,\cdot)$ which is then evaluated over a stochastic (non-gridded) mesh. However, in the context of robust control, existing techniques run into the double challenge of: (i) strong path-dependence of optimal state (X_t^*) on the control, preventing direct application of Regression Monte Carlo (see Section 4); (ii) the nested optimization due to considering multiple $\mathbb{Q}$'s is computationally intensive and requires proper approximation architecture to extract the optimal control $u^*(t,x)$. We employ machine learning techniques to overcome the above challenges, by constructing a non-parametric value function approximation [21, 7]. Specifically, we leverage tools from *statistical emulation* in Stochastic Simulation [11, 6] (and more broadly the Design and Analysis of Computer Experiments field [25]) by recasting the task of solving the Bellman equation as a statistical learning problem of fitting a surrogate (i.e. a statistical model) for $x \mapsto \hat{V}(t,x)$ [23, 18]. Our framework resembles recent results for Monte Carlo based solvers of Hamilton-Jacobi-Bellman equations [16, 2, 13]. However, unlike the above references that propose to apply Deep Neural Networks (DNN) for the function approximation step, we rather rely on Gaussian Process (GP) surrogates. GPs is a core machine learning technique which is well-suited for cases where simulation is expensive and therefore efficient surrogate training is needed (in contrast, DNNs are best suited for high-dimensional problems where maximum flexibility is needed and a lot of training data is available). Related use of GPs is in [3, 18].

Utilizing our numerical algorithm, we provide an extensive investigation into the adaptive robust approach to optimal investment and nonlinear hedging. These are two fundamental problems in financial modeling, yet few fully numerical algorithms have ever been available. Our analysis provides new insights into the interplay between robustifying beliefs, learning and investing/hedging. In particular, we investigate the dependence of the controls and the terminal wealth on risk- and robustness-level parameters.

The rest of the paper is organized as follows. The rest of Section 1 sets up the adaptive robust formulation and lays out the associated discrete-time Bellman recursions. Section 2 develops our numerical methodology and the Gaussian Process surrogate-based algorithm. The second half of the paper illustrates the algorithm on an Optimal Investment (Section 3) and Hedging (Section 4) case studies. Finally, Section 5 provides discussion on the numerical aspects and concludes with a list of additional enhancements.

1.1. The Adaptive Robust Bellman Equation. Let $(\Omega, \mathfrak{F})$ be a measurable space and $T > 0$ a fixed time horizon. Let $\mathcal{T} = \{t_k : k = 0, 1, 2, \ldots, K,\ t_K = T\}$ be the discrete time set and $\Theta \subset \mathbb{R}^p$ be a non-empty set, which plays the role of the global parameter space, so that for each $\theta \in \Theta$ there is a probability measure $\mathbb{Q}^\theta$ on $(\Omega, \mathfrak{F})$. We write $\mathbb{E}_\theta$ to denote the expectation operator corresponding to $\mathbb{Q}^\theta$. We consider a controlled stochastic Markov state

process $Y = \{Y_t^{\vec{u}},\ t \in \mathcal{T}\}$ taking values in the state space $\mathcal{Y} \subseteq \mathbb{R}^d$. We postulate that this process is observed, and we denote by $\mathbb{F} = (\mathcal{F}_t, t \in \mathcal{T})$ its natural filtration. The optimization problem involves the family $\mathcal{A}$ of admissible feedback control processes, which are $\mathbb{F}$–adapted processes $\vec{u} = \{u_t,\ t \in \mathcal{T}\}$ defined on $(\Omega, \mathfrak{F})$ with u_t taking values in a measurable space $\mathcal{U}$ and being the feedback control applied at time t.

Consider the objective of maximizing a reward functional

$$C(\vec{u}) := \sum_{k=0}^{K-1} g(t_k, Y_{t_k}^{\vec{u}}, u_{t_k}) + G(Y_T^{\vec{u}}),$$

with running rewards $g(t, y, u) : \mathcal{T} \times \mathcal{Y} \times \mathcal{U} \mapsto \mathbb{R}$ and terminal reward $G(y) : \mathcal{Y} \mapsto \mathbb{R}$ which are both measurable, square-integrable functions. In the two financial motivations we have $g(t, y, u) \equiv 0$, but since our algorithm is of independent interest, we continue with this slightly more general setup. In the classical setup, the dynamics of Y are described through a probability measure $\mathbb{P}$ and dynamic optimization of $\mathcal{V}(t_0, y_0) = \sup_{\vec{u}\in\mathcal{A}} \mathbb{E}^{\mathbb{P}}[C(\vec{u})|Y_0 = y_0]$ is reduced to the classical dynamic programming Bellman equation

$$\mathcal{V}(t_k, y) = \sup_{u\in\mathcal{U}} \left\{ g(t_k, y, u) + \mathbb{E}_{t_k}^{\mathbb{P}} \left[\mathcal{V}(t_{k+1}, Y_{t_{k+1}}^u) \right] (y) \right\}, \tag{1.1}$$

with $\mathcal{V}(T, y) = G(y)$ and $\mathbb{E}_{t_k}^{\mathbb{P}}[\cdot](y) \equiv \mathbb{E}^{\mathbb{P}}[\cdot|Y_{t_k} = y]$.

The pervasive challenge of prescribing $\mathbb{P}$-dynamics implies model mis-specification for determining the maximizer u^* in (1.1). *Robust* control injects consideration of multiple models directly into the optimization step via a generic min-max formulation. Therefore, the single $\mathbb{P}$ is replaced with many $\mathbb{Q}$'s. In the parametric setup we consider, model uncertainty is indexed via $\theta \in \Theta$ whose effect is through the dynamics

$$Y_{t_{k+1}}^u = T_{(Y)}(t_k, Y_{t_k}, u, \theta, \epsilon_{t_{k+1}}). \tag{1.2}$$

The mapping $T_{(Y)}$ captures the joint effect of the initial condition y, the control u applied at epoch t_k, the parameterized measure $\mathbb{Q}^\theta$, the stochastic shock $\epsilon_{t_{k+1}}$ and the time-dependence t_k. Note that (1.2) postulates that we may re-write the evolution of Y in terms of external, model-independent stochastic shocks; in the common setting we consider, ϵ_{t_k} are independent and identically distributed random variables (without loss of generality taken to be standard Gaussian in some $\mathbb{R}^q$).

For each $(t_k, y, u, \theta) \in \mathcal{T} \times \mathbb{R}^d \times \mathcal{U} \times \mathbf{\Theta}$, define a probability measure on $\mathbb{R}^d$:

$$Q(B \mid t_k, y, u, \theta) = \mathbb{Q}^\theta \left(T_{(Y)}(t_k, y, u, \theta, \varepsilon) \in B \right),$$

for any Borel measurable set $B \subseteq \mathbb{R}^d$. For every control process $\vec{u} \in \mathcal{A}$, every $Y_0 = y_0$, and every sequence $\vec{\theta} = \{\theta_k \in \Theta,\ k = 0, \ldots, K-1\}$, let $\mathbb{Q}_{y_0}^{\vec{u},\vec{\theta}}$ be the "pasted" probability measure on the canonical space $\mathsf{X}_{k=0}^K \mathbb{R}^d$ defined by:

$$\mathbb{Q}_{y_0}^{\vec{u},\vec{\theta}}(B_0 \times \cdots \times B_K) = \int_{B_0} \cdots \int_{B_K} \prod_{k=1}^{K} Q\left(dy_{t_k} \mid t_{k-1}, y_{t_{k-1}}, u_{t_{k-1}}, \theta_{k-1}\right) \delta_{y_0}(dy),$$

where δ is the Dirac measure. We restrict $\theta_k \in \Theta_k \in \mathcal{F}_{t_k}$ to certain adapted time-dependent subsets of Θ detailed below and define the set $\mathcal{Q}_{y_0}^{\vec{u}} = \{\mathbb{Q}_{y_0}^{\vec{u},\vec{\theta}},\ \theta_k \in \Theta_k,\ k = 0, \dots, K-1\}$.

The robust control setup then considers the dynamic optimization problem in the following form:

$$\mathcal{V}(t_0, y_0) = \sup_{\vec{u}\in\mathcal{A}} \inf_{\mathbb{Q}\in\mathcal{Q}_{y_0}^{\vec{u}}} \mathbb{E}^{\mathbb{Q}}\left[C(\vec{u}) \mid Y_0 = y_0\right],$$

which, for each $t_k \in \mathcal{T}$, admits the one-step Bellman representation

$$\mathcal{V}(t_k, y) = \sup_{u\in\mathcal{U}} \inf_{\mathbb{Q}\in\mathcal{M}(\Theta_k)} \left\{ g(t_k, y, u) + \mathbb{E}_{t_k}^{\mathbb{Q}}[\mathcal{V}(t_{k+1}, Y_{t_{k+1}}^u)](y) \right\} \tag{1.3}$$

$$= \sup_{u\in\mathcal{U}} \inf_{\theta\in\Theta_k} \left\{ g(t_k, y, u) + \int_{\mathbb{R}^q} \mathcal{V}(t_{k+1}, T_{(Y)}(t_k, y, u, \theta, z)) f_\epsilon(z) dz \right\}, \tag{1.4}$$

where the second line follows from (1.2), $f_\epsilon(\cdot)$ is the density of the random variable $\epsilon_{t_{k+1}}$, and $\mathcal{M}(\Theta_k)$ is the family of probability measures on Ω corresponding to the adversarial parameter set Θ_k. The expression (1.3) can be seen as a min-max game between nature that picks $\mathbb{Q}^\theta$ and the controller who counteracts with her control $u = u_{t_k}$. In information-theoretic language, (1.3) is a game against the adversary who selects $\mathbb{Q}^\theta$ among the Knightian uncertainty set $\mathcal{M}(\Theta_k)$ [15]. Note that Θ_k is understood to be dynamic, and in particular can depend on Y_{t_k}.

The *adaptive robust* control as developed by Bielecki et al. [9], couples the Bayesian updating procedure to the set of measures in (1.3). Specifically, we consider

$$\Theta_k = \Theta(t_k, \bar{\theta}_{t_k}),$$

for a given measurable $\Theta(\cdot,\cdot) : \mathcal{T} \times \mathbb{R}^p \mapsto \mathcal{B}(\mathbb{R}^p)$ interpreted as the robustified belief set defined by the sufficient statistic $\bar{\theta}_{t_k}$. As a canonical example, $\Theta(t_k, \bar{\theta}_{t_k})$ is the posterior credible interval at some level α (say the 95% CI) centered around the unbiased point estimate $\bar{\theta}_{t_k}$. This interpolates *dynamic adaptive* control, where $\mathcal{M}_k = \mathbb{Q}^{\bar{\theta}_{t_k}} \Leftrightarrow \Theta_k = \{\bar{\theta}_{t_k}\} \Leftrightarrow \alpha = 0.5$ (no uncertainty but learning), with the robust versions that do not allow for learning, $\mathcal{M}_k = \{\mathbb{Q}^\theta : \theta \in \bar{\Theta}\} \Leftrightarrow \Theta_k \equiv \bar{\Theta}$ independent of t_k. (We should also mention the *myopic adaptive* formulation where one plugs-in $\bar{\theta}_{t_k}$ into the static model that treats θ as a parameter. This is in fact the most common applied usage which myopically separates learning from control.)

The learning algorithm is meant to reduce the uncertainty about the true probabilistic structure driving Y. If one assumes that there exists $\tilde{\theta}$ such that $Y_{t_{k+1}} = T_{(Y)}(t_k, y, u, \tilde{\theta}, \epsilon_{t_{k+1}})$ for all k and the learning and estimation are asymptotically perfect, $\bar{\theta}_{t_k} \to \tilde{\theta}$ as $k \to \infty$ then it is also reasonable to expect $\Theta_k \to \{\tilde{\theta}\}$ and the adaptive robust method brings a consistently conservative extension of Bayesian adaptive control. However, the framework can also handle more general situations, such as when Θ_k never converges to a singleton, or when no $\tilde{\theta}$ exists. Thus, the adaptive robust can handle non-stationary models where the dynamics change over time (as might be imagined happens in real markets, leading to a process $\tilde{\theta}(t_k)$); in such settings $\bar{\theta}_{t_k}$ represents the best *current* guess about $\tilde{\theta}(t_k)$. So the size of Θ_k need not go to zero.

Returning to the control objective, since $\bar{\theta}_{t_k}$ now affects the inner optimization in (1.4), it is augmented to the system state

$$X_t := (Y_t, \bar{\theta}_t) \qquad \text{with state space} \quad \mathcal{X} = \mathcal{Y} \times \Theta \subseteq \mathbb{R}^{d+p}.$$

The dynamics of $(X_{t_k}) = \{X_t^{\vec{u}},\ t \in \mathcal{T}\}$ consist of the autonomous dynamics of (Y_{t_k}), as well as the filtering equations that govern the evolution of $(\bar{\theta}_{t_k})$. We summarize them by the mapping

$$X_{t_{k+1}}^{x,u} = \mathbf{T}(t_k, x, u, \theta, \epsilon_{t_{k+1}}), \quad x := (y, \bar{\theta}) \in \mathcal{X},$$

which aggregates the previous dynamics $T_{(Y)}$ of the state process with the filtering equations for $\bar{\theta}_{t_k} \mapsto \bar{\theta}_{t_{k+1}}$. Note that those joint dynamics are degenerate since a single noise term $\epsilon_{t_{k+1}}$ simultaneously drives both $Y_{t_{k+1}}$ and $\bar{\theta}_{t_{k+1}}$. As a result, there is a strong dependence between Y_{t_k} and $\bar{\theta}_{t_k}$, and more generally Θ_{t_k}, which is to be contrasted with the static robust framework where Y_{t_k} and Θ are independent. The resulting adaptive robust Bellman equation which is the main object of analysis in this article is then

$$V(t_k, x) = \sup_{u \in \mathcal{U}} \inf_{\theta \in \Theta(t_k, \bar{\theta}_{t_k})} \mathbb{E}\left[g(t_k, y, u) + V\left(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon_{t_{k+1}})\right)\right]. \tag{1.5}$$

Note that $\mathbb{E}$ now simply denotes expectation over $\epsilon_{t_{k+1}} \in \mathbb{R}^q$ which is the sole stochastic, model-independent component above; the parameter dependence and model risk are encoded in $\mathbf{T}$. The solution entails applying backward induction over $t = t_{K-1}, \ldots, 0$ to compute $V(t_k, \cdot)$.

1.2. Solving the Bellman Equation. To implement the adaptive robust framework, we must solve (1.5) which practically reduces to (1.4) with the state-dependent uncertainty set $\Theta(t_k, \cdot)$. If we define the operators $\mathscr{F}, \mathscr{M}$ via

$$\mathscr{M}(t_k, x = (y, \bar{\theta}))[\mathscr{F}] := \sup_{u \in \mathcal{U}} \inf_{\theta \in \Theta(t_k, \bar{\theta})} \mathscr{F}(u, \theta; t_k, x)$$
$$\mathscr{F}(u, \theta; t_k, x)[V] := \mathbb{E}\left[g(t_k, y, u) + V(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon_{t_{k+1}}))\right]$$

then the Bellman recursion is $V(t_k, x) = \mathscr{M}(t_k, x) \circ \mathscr{F}(\cdot, \cdot, t_k, x)[V]$. $\mathscr{F}$ is the *propagation operator* that computes the continuation value based on taking conditional expectations of step-ahead value function. $\mathscr{M}$ is the *optimization operator* that computes the optimal control at (t_k, x), recording the optimizers $(\theta^*(t_k, x), u^*(t_k, x)) = \arg\sup_{u \in \mathcal{U}} \inf_{\theta \in \Theta(t_k, \bar{\theta})}\{\cdot\}$. $\theta^*(t_k, x)$ is the worst-case parameter (i.e. beliefs) for the model dynamics given $(t_k, y, \bar{\theta})$ and $u^*(t_k, x)$ is the robustified feedback control in state x. The difference $\theta^*(t_k, (y, \bar{\theta}))$ vs $\bar{\theta}$ is precisely the precaution taken by the controller against model mis-specification.

Three challenges must be handled to solve (1.5):

- Continuous state space: because the state space $\mathcal{X}$ of X is continuous and multi-dimensional, some discretization is required to handle it computationally. This discretization however implies that $V(t_k, \cdot)$ (i.e. $\mathscr{F}(\cdot, \cdot; t_k, x)[V]$) will not be computed for *all* potential x, but only for some finite subset. Therefore, space discretization must be accompanied by *interpolation* in order to be able to evaluate $V(t_k, x)$ for arbitrary $x \in \mathcal{X}$ in subsequent steps.
- Integration: the integral over $\epsilon_{t_{k+1}}$ in (1.4) requires approximation given that the integrand is not analytically available. This implies that the true conditional operator $\mathbb{E}$ is to be replaced with an approximation $\hat{E}$, i.e. $\mathscr{F}$ is replaced with $\hat{F}$.
- Optimization: finding the optimizers $\theta^*(t_k, x)$ and $u^*(t_k, x)$ is generally not possible analytically, so another approximation is required to carry out these optimizations numerically. This implies that we replace $\mathscr{M}$ with an approximation $\hat{M}$.

In low dimensions, the standard technique for Bellman equations are PDE-based solvers that discretize $\mathcal{X}$. However, the augmentation of beliefs $\bar{\theta}$ to the Y-state necessarily leads to multi-dimensional setups, essentially ruling out PDE-based strategies that are practical only in dimension ≤ 3. At the same time, simulation-based strategies from conventional regression Monte Carlo are also difficult to apply to (1.5):

- The control $\vec{u}_t$ intrinsically affects the evolution of the state $(Y_t^{\vec{u}})$ and hence prevents direct *simulation* of the overall (X_t) as is done in the standard RMC paradigm.
- Parametric representation of an approximate value function $\hat{V}(t_k, \cdot)$ (e.g. in terms of polynomials in x) is challenging in dimension $d + p > 2$ and brings the concern of overfitting/underfitting;
- Because we face the additional step of inner optimization over θ, the representation of $x \mapsto \hat{V}(t_k, x)$ is critical to yield financially reasonable/accurate estimates of $\theta^*(t_k, x)$, which is another limitation of classical parametric approximations;
- One usually learns $\hat{V}$ by constructing a grid of x-values (akin to finite-difference approaches for PDEs); however such gridding is extremely inefficient for $d + p > 2$ and essentially impossible for $d + p > 4$.

As a result, fully numerical approaches to robust stochastic control have been traditionally viewed as intractable and there remains a large gap in actual use of robust models for financial modeling. In this article we resolve these challenges, making contributions along two directions. On the algorithmic side, we propose a new, machine-learning-inspired algorithm for (1.4). The key concept is to employ a non-parametric value function approximation strategy [21, 7], namely a Gaussian process surrogate. In addition, we borrow concepts from the Design and Analysis of Computer Experiments field [25] to construct the underlying stochastic meshes. This methodology is quite general, and could also be easily modified to tackle other robust control formulations (e.g strong robust), and other contexts (e.g robust optimal stopping). Moreover, it mitigates the issue of scalability to higher dimensions. Related extensions would be treated in separate sequels, and we see a lot of further potential for such blending of RMC and machine learning. On the finance side, we present two detailed case studies of using adaptive robust paradigm in optimal investment and loss-based hedging. In particular, the latter problem, see (4.5) below, was essentially out of reach until now. Employing our algorithm, we are able to give a comprehensive investigation of the resulting strategy and its sensitivity to different model parameters. We also demonstrate that the adaptive robust framework outperforms the common (simpler) alternatives of static robust and adaptive control.

1.3. Motivation: Adaptive Robust Optimal Investment. The problem of optimal investment in risky financial instruments dates back to Markowitz and Merton and has been extensively studied over the past 50 years. Asset return μ and volatility σ are the two main parameters driving the investment decisions, but are notoriously difficult to calibrate. Therefore, there is strong interest in methods that explicitly take into account parameter uncertainty, while allowing one to partially learn asset dynamics.

Let r be the constant risk-free interest rate. We assume a fixed time grid $t_k = t_0 + k\Delta t$. The excess log-return of a risky asset S between t_k and t_{k+1} is Gaussian with mean $\mu\Delta t$ and variance $\sigma^2\Delta t$. Both μ and σ are treated as unknown, yielding the framework of (1.3) with

$\theta \equiv (\mu, \sigma)$. Thus, the dynamics of the controlled wealth process $Y^{\vec{u}}$ under $\mathbb{Q}^{\mu,\sigma}$ are given by

$$Y^u_{t_{k+1}} = Y_{t_k}(1 + r\Delta t + u(e^{\mu\Delta t + \sigma\sqrt{\Delta t}\epsilon_{t_{k+1}}} - r\Delta t - 1)), \qquad \epsilon_{t_{k+1}} \sim \mathcal{N}(0,1) \quad \text{i.i.d,} \tag{1.6}$$

where $u \in [0,1] =: \mathcal{U}$ is the proportion of portfolio wealth invested in the risky asset. Above we restrict the investor from shorting $u < 0$ or leveraging $u > 1$ her risky position. The objective is to maximize expected utility of terminal wealth

$$\mathbb{E}\left[U(Y^{\vec{u}}_T)\right] \to \max_{\vec{u}}!$$

for a utility function $U(\cdot)$. The augmented state x is three-dimensional: $x = (y, \bar{\mu}, \bar{\sigma})$ and the Bellman equation for $V(t_k, x)$ is

$$V(t_K, x) = U(y); \tag{1.7}$$

$$V(t_k, x) = \sup_{u\in[0,1]} \inf_{(\mu,\sigma)\in\Theta(t_k,\bar{\mu},\bar{\sigma})} \mathbb{E}\left[V(t_{k+1}, \mathbf{T}(t_k, x, u, (\mu,\sigma), \epsilon_{t_{k+1}}))\right] \tag{1.8}$$

where the expectation is over the standard Gaussian increment $\epsilon_{t_{k+1}} \sim \mathcal{N}(0,1)$.

To specify the transition map $\mathbf{T}$, we recall the learning of the asset dynamics. Given past observations of Y, the respective MLE estimators of the drift μ and the volatility σ are denoted by $\bar{\theta}_{t_k} = (\bar{\mu}_{t_k}, \bar{\sigma}_{t_k})$ and under $\mathbb{Q}^{\mu,\sigma}$ satisfy for $k = 0, 1, \ldots$ the recursions [8]

$$\bar{\mu}_{t_{k+1}} = \frac{k+1}{k+2}\bar{\mu}_{t_k} + \frac{1}{k+2}\big(\mu + \frac{\sigma}{\sqrt{\Delta t}}\epsilon_{t_{k+1}}\big), \tag{1.9}$$

$$\bar{\sigma}^2_{t_{k+1}} = \frac{k+1}{k+2}\bar{\sigma}^2_{t_k} + \frac{k+1}{(k+2)^2}\left((\bar{\mu}_{t_k} - \mu)\sqrt{\Delta t} - \sigma\epsilon_{t_{k+1}}\right)^2. \tag{1.10}$$

Note that the above are time-dependent, capturing the idea that learning slows over time as the information set grows. As k increases, the weight of the prior $(\bar{\mu}_{t_k}, \bar{\sigma}_{t_k})$ rises and the role of the latest shock $\epsilon_{t_{k+1}}$ declines.

In sum, the dynamics of $x_{t_k} = (y_{t_k}, \bar{\mu}_{t_k}, \bar{\sigma}_{t_k})$ under $\mathbb{Q}^{\mu,\sigma}$ are prescribed by (1.6)-(1.9)-(1.10), all driven by the 1-D exogenous factor $\epsilon_{t_{k+1}}$. They are summarized by the map

$$\begin{aligned}\mathbf{T}(t_k, (y, \bar{\mu}, \bar{\sigma}), u, (\mu,\sigma), z) := \Big(&y(1 + r\Delta t + u(e^{\mu\Delta t + \sigma\sqrt{\Delta t}z} - r\Delta t - 1)),\\ &\frac{k+1}{k+2}\bar{\mu} + \frac{1}{k+2}(\mu + \frac{\sigma}{\sqrt{\Delta t}}z), \sqrt{\frac{k+1}{k+2}\bar{\sigma}^2 + \frac{k+1}{(k+2)^2}((\bar{\mu} - \mu)\sqrt{\Delta t} - \sigma z)^2}\Big),\end{aligned} \tag{1.11}$$

where μ, σ are treated as external parameters.

Assuming there is a true $\tilde{\theta}$, the uncertainty set $\Theta(t_k, \cdot)$ is described through the (1-α)-confidence region for $\tilde{\theta}$ for some confidence level $\alpha \in (0,1)$. In the case where only the drift μ is uncertain and σ is given, $\Theta_\alpha(t_k, \bar{\mu}_{t_k})$ is an interval centered on $\bar{\mu}_{t_k}$,

$$\Theta_\alpha(t_k, \bar{\mu}_{t_k}) := \left[\bar{\mu}_{t_k} - \frac{\sigma}{\sqrt{(k+1)\Delta t}}q_{\alpha/2},\ \bar{\mu}_{t_k} + \frac{\sigma}{\sqrt{(k+1)\Delta t}}q_{\alpha/2}\right], \tag{1.12}$$

where q_α denotes the α-quantile of a standard normal distribution. In the 2-D case where both μ and σ are uncertain, it is shown in [8] that the $(1-\alpha)$-confidence region for $(\tilde{\mu}, \tilde{\sigma})$ at time t_k is an ellipsoid given by

$$\Theta_\alpha(t_k, \bar{\mu}_{t_k}, \bar{\sigma}_{t_k}) := \left\{ (\mu, \sigma) \in \mathbb{R}^2 \ : \ \frac{(k+1)\Delta t}{\bar{\sigma}^2_{t_k}} (\mu - \bar{\mu}_{t_k})^2 + \frac{k+1}{2\bar{\sigma}^4_{t_k}} (\sigma^2 - \bar{\sigma}^2_{t_k})^2 \leq \kappa \right\}, \tag{1.13}$$

where κ is the $(1-\alpha)$-quantile of the $\chi^2(2)$ distribution with two degrees of freedom.

When the utility function is of the CRRA power type, $U(y) = \frac{y^{1-\gamma}}{1-\gamma}$, then the problem (1.7)-(1.8) is separable with respect to the current wealth y: $V(t_k, y, \bar{\mu}, \bar{\sigma}) = y^{1-\gamma} \tilde{V}(t_k, \bar{\mu}, \bar{\sigma}^2)$ with the dimension-reduced recursion $\tilde{V}(t_K, \bar{\mu}, \bar{\sigma}) \equiv \frac{1}{1-\gamma}$, and for $0 \leq k \leq K-1$

$$\begin{aligned} \tilde{V}(t_k, \bar{\mu}, \bar{\sigma}^2) = \sup_{u \in [0,1]} \inf_{(\mu,\sigma) \in \Theta(t_k, \bar{\mu}, \bar{\sigma})} \mathbb{E}\Big[& (1 + r\Delta t + u(e^{\mu \Delta t + \sigma \sqrt{\Delta t} \epsilon_{t_{k+1}}} - r\Delta t - 1))^{1-\gamma} \times \\ & \tilde{V}\big(t_{k+1}, \frac{k+1}{k+2}\bar{\mu} + \frac{1}{k+2}(\mu + \frac{\sigma}{\sqrt{\Delta t}} \epsilon_{t_{k+1}}), \frac{k+1}{k+2}\bar{\sigma}^2 + \frac{k+1}{(k+2)^2}((\bar{\mu} - \mu)\sqrt{\Delta t} - \sigma \epsilon_{t_{k+1}})^2\big) \Big]. \end{aligned} \tag{1.14}$$

We will solve (1.14) in Section 3. Another case study which also relies on $\theta = (\mu, \sigma)$ is considered in Section 4.

2. Methodology. In describing our algorithm below, we make use of the following notational shorthands: true quantities that we wish to approximate, are denoted with a *, e.g. u^* for optimal control or θ^* for adaptive worst-case parameters; approximations are denoted with a hat $\hat{\cdot}$; intermediate statistical samples that drive the approximations are denoted with a check $\check{\cdot}$ and are indexed with superscript n reflecting the empirical Monte Carlo sample. We also employ a bit of `pseudo-code` terminology to denote generic computational building blocks that are available in high-level modeling environments like `R`, `Matlab` or `Python`.

Our algorithm is based on value function approximation. This is achieved by evaluating the right hand side of (1.5) at a collection of inputs $x_{t_k}^{1:N}$ and then constructing a statistical surrogate $x \mapsto \hat{V}(t_k, \cdot)$. At a high level, the solver consists of a `fit`–`predict`–`optimize` loop over time based on the recursion

$$\hat{V}(t_k, x) = \sup_{u \in \mathcal{U}} \inf_{\theta \in \Theta(t_k, \bar{\theta}_{t_k})} \left\{ g(t_k, y, u) + \hat{E}\left[\hat{V}(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \cdot)) \right] \right\}, \tag{2.1}$$

where `predict` is needed to evaluate the $\hat{V}(t_{k+1}, \cdot)$'s and `fit` to construct $\hat{V}(t_k, \cdot)$. For the latter, we use the machine learning tool of a Gaussian Process surrogate [22]. This is achieved by selecting an experimental design $\mathcal{D}_k = (x_{t_k}^{1:N})$ and then evaluating the right hand side of (2.1) at $x_{t_k}^{1:N}$ which yields values $v_{t_k}^{1:N}$, interpreted as pointwise estimates of the value function at respective $x_{t_k}^n$, $v_{t_k}^n \simeq V(t_k, x_{t_k}^n)$. To compute $v_{t_k}^n$ the expectation in (1.14) is approximated with a quantized sum, and the inner saddle-point is computed locally using a nonlinear optimization algorithm. More precisely, we use a numerical quadrature representation $\hat{E}$ of the integral over $\epsilon_{t_{k+1}}$, and a nested `optimize` call to get $\check{u}_{t_k}^n := \arg\sup_{u \in \mathcal{U}} \inf_\theta \{ g(t_k, y, u) + \hat{E}[\hat{V}(t_{k+1}, T(x_{t_k}^n, u, \theta, \epsilon))] \}$ which is interpreted as the estimated optimal control at $x_{t_k}^n$.

Because the training data $(x_{t_k}^{1:N}, v_{t_k}^{1:N})$ is based on multiple approximations, including approximating the true $\mathbb{E}[\cdot]$, approximating the true saddle-point, approximating the true $V(t_{k+1},\cdot)$, we treat them as *noisy* versions of $\hat{V}(t_k, x_{t_k}^n)$ and thus the surrogate fitting step includes smoothing, rather than solely interpolation. Similarly, we view $\check{u}_{t_k}^n$ as pointwise samples from the feedback control map $\hat{u}(t_k, x)$ that is also obtained via (an independent) surrogate fitting.

Our algorithm returns the fitted surrogates $\{\hat{V}(t_k,\cdot)\}$ and $\{\hat{u}(t_k,\cdot)\}$ for each time step. The interpretation of $\hat{V}(t_k, x)$ is as the value function of the underlying adaptive robust control problem. However, for practical purposes we typically wish to know the strategy and resulting utility based on a fixed measure $\mathbb{Q}^{test}$, while (1.3) assumes that the parameters are dynamically and adversarially drawn from Θ_k at each time-step. To this end, we concentrate on the outputted feedback control surrogate $\hat{u}(t_k,\cdot)$. To evaluate the resulting utility we rely on forward Monte Carlo, i.e. we generate *out-of-sample* forward paths by drawing $Y_{t_{k+1}}$-realizations based on $\mathbb{Q}^{test}$ that induces the controlled trajectories $X_{t_{0:K}}^{\hat{u}}$ and then evaluate the Monte Carlo estimate

$$\underline{V}^{\mathbb{Q}^{test}}(0, x_0) = \frac{1}{N'}\sum_{n=1}^{N'} C_K^n.$$

Observe that $V(0, x_0)$ can be written as an expectation, under the path-dependent, "pasted" measure Q^* which is defined in terms of the worst-case beliefs $\theta^*(t_k, X_{t_k})$ at each time step and does not admit a simple interpretation. Indeed, the expectation of $V(t_{k+1}, \mathbf{T}(t_k, X_{t_k},\cdot))$ is carried out with respect to $\mathbb{Q}^{\theta^*(t_k, X_{t_k})}$ which depends on X_{t_k}. For this reason, the representation $V(0, x_0) = \mathbb{E}^{Q^*}\left[\sum_k g(t_k, Y_{t_k}^{u^*}, u^*(t_k, X_{t_k})) + G(Y_{t_K}^{u^*})\right]$ is generally not practically relevant.

Remark 2.1. *During forward simulation one could revert to computing the actual minimizer $\check{u}(t_k, x)$ that is obtained by calling* `predict` *to evaluate $\hat{V}(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon_{t_{k+1}}))$ and then optimizing. However, optimization is expensive and so we rather utilize the surrogate-based $\hat{u}(t_k, x)$ that is obtained from a statistical prediction and is much faster to evaluate. Both $\check{u}(t_k, x)$ and $\hat{u}(t_k, x)$ are defined based on $\hat{V}(t_{k+1},\cdot)$ and hence will necessarily differ from the true optimal feedback $u^*(t_k, x)$ due to error back-propagation.*

Remark 2.2. *When the optimization over u is over a compact space such as $[0, 1]$, an alternative to generic gradient-free solvers, such as* `fminbnd` *or* `optim` *in* `Matlab`*, is to employ direct search over a discrete candidate set such as $\{0, \Delta u, 2\Delta u, \ldots, 1\}$. This is faster but of course introduces a fixed discretization error relative to the true optimum. Conversely, the surrogate can output the gradient $\partial\hat{V}/\partial x$ that could be combined with the chain rule to enable gradient-based optimization.*

Before proceeding to discuss the details of operationalizing the above method, we summarize the original proposal of Bielecki et al. [9] that relied on a Control Randomization (CR) + Regression approach. Bielecki et al. started with a Monte Carlo-based paradigm that works with trajectories of (X_{t_k}) and exploited the fact that the underlying state (Y_{t_k}) is an autonomous process. In their approach, one begins by generating N trajectories of Y_{t_k}. In addition, one fixes a measure $\mathbb{Q}^0$ and uses the above N trajectories to construct the corresponding filtered posterior estimates $\bar{\theta}_{t_k}$ as in (1.9)-(1.10). Note that $\mathbb{Q}^0$ is picked arbitrarily and serves

as a "baseline" probability measure like in the CR method [1, 17]. Next, the Bellman equation (1.3) is solved *pathwise* along the above $x_{t_k}^{1:N} = (y_{t_k}^{1:N}, \bar{\theta}_k^{1:N})$ as follows. Given $(t_k, Y_{t_k}, \bar{\mu}_k, \bar{\sigma}_k)$, one solves for $\hat{V}(t_k, \cdot)$ by replacing the integral with a weighted sum

$$\begin{aligned} \hat{E}[\hat{V}(t_{k+1}, x, u, \cdot)] &:= \int \hat{V}(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, z)) f_\epsilon(z) dz \\ &\approx \sum_{i=1}^{\mathfrak{I}} \hat{V}(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon^{(i),\mathfrak{I}})) w^{(i),\mathfrak{I}}, \end{aligned} \tag{2.2}$$

where the Gaussian quadrature recipe specifies the appropriate knots $\epsilon^{(i),\mathfrak{I}}$ and corresponding quadrature weights $w^{(i),\mathfrak{I}}$. The latter $(\epsilon^{(i)}, w^{(i)})$ are optimized to minimize a certain global criterion [4, 20]. Since the noise distribution $\epsilon_{t_{k+1}} \sim \mathcal{N}(0,1)$ is fixed throughout, the optimal quadrature recipe is pre-computed and stored offline. In the second sub-step, the one-step-ahead values $\hat{V}(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon^{(i),\mathfrak{I}}))$ are approximated—since they will not fall on the mesh $\hat{V}(t_{k+1}, x_{t_{k+1}}^{1:N})$—through linear interpolation of the training collection $\{\hat{V}(t_{k+1}, x_{t_{k+1}}^{1:N})\}$.

The above recipe has several limitations that our new proposal overcomes. First, the use of a pre-specified $\mathbb{Q}^0$ leads to a non-adaptive experimental design, which puts $\bar{\theta}_{t_k}^{1:N}$ potentially very far from the likely values of $\bar{\theta}_{t_k}$ (depending on whether $\mathbb{Q}^0$ is "good" or not). By breaking up the path-based construction, we are able to adaptively build better designs, which in particular leverage the structure of u^*. Second, linear interpolation of $\hat{V}(t_{k+1}, x_{t_{k+1}}^{1:N})$'s brings three restrictions: (i) it poorly scales in dimension d, because the interpolation requires sorting of all the N sites $x_{t_k}^{1:N}$ to identify nearest neighbors; (ii) it implies that $\hat{V}(t_{k+1}, \cdot)$ is piecewise linear which is a restrictive approximation architecture (in particular non-smooth which may cause problems when solving for $\check{u}$); (iii) it takes the estimated values of $\hat{V}(t_{k+1}, \cdot)$ as *exact*, providing no ability to *smooth* previous approximations. Third, the procedure only outputs the pointwise estimates $\{\check{u}(t_k, x_{t_k}^n)\}$; there is no simple way to obtain $\check{u}(t_k, x_*)$ for arbitrary x_*, essentially ruling out out-of-sample estimation under any other measure than $\mathbb{Q}^0$. Fourth, the approach has no way to avoid *extrapolation* when using (2.2); linear extrapolation is prone to significant errors that are likely to back-propagate. Fifth, pre-simulation under $\mathbb{Q}^0$ requires to use the same mesh size N across all time steps.

2.1. Gaussian Process Surrogates. Gaussian process models is a popular choice for flexible statistical models. They allow a consistent treatment of interpolation and regression with very few tunable hyperparameters. The idea of GPs is to view the function to be approximated as a realization of a Gaussian random field with covariance kernel $K(\cdot, \cdot)$. Training the model then reduces to applying the Gaussian conditional equations, i.e. evaluating the distribution of the random field given the data. The Gaussian structure implies that the conditional process is still Gaussian, so its distribution is summarized through the posterior mean $m_*(\cdot)$ and posterior (co-)variance $s_*(\cdot, \cdot)$, interpreted as the model-predicted approximation $\hat{V}(t_k, \cdot)$ and the corresponding posterior uncertainty or "standard error".

Intuitively, the GP offers a methodology to put a smooth surface through the data labelled generically as $(\mathbf{x}, \mathbf{v}) \equiv (x^{1:N}, v^{1:N})$. For numerical stability, we introduce the "nugget" η^2, so that rather than exactly interpolating the given v^n's, the surrogate also smoothes. Namely,

the nugget is equivalent to assuming

$$v^n_{t_k} = V(t_k, x^n_{t_k}) + \epsilon^n \qquad \text{with} \quad \epsilon^n \sim \mathcal{N}(0, \eta^2).$$

In our context, adding ϵ^n's is justified via the errors coming from quantization and numerical optimization and we use $\eta = 10^{-5}$. The predicted $\hat{V}(t_k, x_*) \equiv m_*(x_*)$ at any input x_* is given by

$$m_*(x_*) = \mathbf{k}(x_*)[\mathbf{K} + \eta^2\mathbf{I}]^{-1}\mathbf{v}, \tag{2.3}$$

with corresponding posterior covariance $s_*(x_*, x'_*)$

$$s_*(x_*, x'_*) = K(x_*, x'_*) - \mathbf{k}(x_*)[\mathbf{K} + \eta^2\mathbf{I}]^{-1}\mathbf{k}(x'_*), \tag{2.4}$$

with the $N \times 1$ vector $\mathbf{k}(x_*)$ and $N \times N$ matrix $\mathbf{K}$ defined by

$$\mathbf{k}(x_*) := K(x_*, \mathbf{x}) = [K(x_*, x^1), \ldots, K(x_*, x^N)], \quad \text{and} \quad \mathbf{K}_{i,j} := K(x^i, x^j). \tag{2.5}$$

Above $\mathbf{I}$ is a $N \times N$ identity matrix and $\eta^2\mathbf{I}$ is the noise matrix.

To construct a GP surrogate requires selecting a kernel *family* and the corresponding hyper-parameters. The latter step is typically done through Maximum Likelihood Estimation, i.e. a nonlinear optimization problem involving the respective likelihood of observations. A common kernel family we use is the Matern-5/2,

$$K_\vartheta(x, x') := \tau^2 \prod_{i=1}^{d+p} \left(1 + \frac{\sqrt{5}r}{\rho_i} + \frac{5r^2}{3\rho_i^2}\right) \exp\left(-\frac{\sqrt{5}r}{\rho_i}\right), \quad r := |x_i - x'_i|, \tag{2.6}$$

where the lengthscales ρ_i and the process variance τ^2 are estimated via maximum likelihood. Specifically, we use the GPML `Matlab` package to obtain the MLE for the hyperparameters $\vartheta := (\eta, \tau, \rho_{1:(d+p)})$. Another kernel is the squared-exponential

$$K_\vartheta(x, x') := \tau^2 \prod_i \exp\left(-\frac{(x_i - x'_i)^2}{2\rho_i^2}\right). \tag{2.7}$$

The GP predictive surface $x_* \mapsto m_*(x_*)$ is akin to kernel regression in the sense that the prediction (2.3) at x_* is always a weighted average of the v^n's, with the weights driven by the spatial covariance structure encoded in $K(\cdot, \cdot)$. In particular, if the spatial correlation decays quickly, the surface will tend to be more "bumpy", while for very strong spatial correlation the surface will be nearly flat/linear. The differentiability of the predictive surface $m_*(\cdot)$ is driven by the properties of $K(\cdot, \cdot)$. Under the Matern-5/2 choice (2.6), the resulting $m_* \in \mathcal{C}^2$ is twice-differentiable, while for (2.7) it is $\mathcal{C}^\infty$. In both families, the lengthscales ρ_i determine the spatial "wiggliness" of the fitted surface in the respective coordinate. Note that different lengthscales (anisotropy) in different coordinates allow for $\hat{V}$ to be, say, more flat in S, but more flexible in $\bar{\mu}$.

For the propagation operator $\mathscr{F}$ showing in the Bellman equation we proceed as in (2.2), approximating the integral over a standard normal distribution by a $\mathfrak{I}$-points optimal quantizer $\hat{E}$, i.e. a discrete sum over $\mathfrak{I}$ quadrature points $\epsilon^{(i),\mathfrak{I}}, i = 1, \ldots, \mathfrak{I}$.

Remark 2.3. *In CR-RMC [1, 17], solving the Bellman equation*

$$v(t_k, x) = \sup_{u \in \mathcal{U}} \left\{ \mathbb{E}\left[v(t_{k+1}, X^u_{t_{k+1}}) \right](x) \right\}$$

is handled by defining the q-value $q(t_k, x, u) := \mathbb{E}[v(t_{k+1}, X^u_{t_{k+1}})](x)$ *across* $u \in \mathcal{U}$*, building a surrogate* $(x, u) \mapsto \hat{q}(t_k, x, u)$ *and then finding* $\check{u}(t_k, x)$ *as the analytic maximizer of* $u \mapsto \hat{q}(t_k, x, u)$*. In contrast, in our approach, we first fix* x *and maximize* $\mathscr{F}$ *at* x *to obtain* $\check{u}(t_k, x)$*, then fit a surrogate* $\hat{u}(t_k, x)$*, effectively reversing the order of optimization and emulation.*

2.2. Experimental Design. The quality of the regression estimates is critically linked to the choice of the simulation design $\mathcal{D}_k = x^{1:N}_{t_k}$. Specifically, accuracy of the GP-based $\hat{V}(t_k, x_*)$ at some given input x_* is directly related to the *density* of the design $\mathcal{D}_k$ around x_*. This is the localization property of the GP and matches the intuition of constructing an interpolant for the training data $(x^{1:N}, v^{1:N})$: the interpolated prediction would be good in the neighborhoods of x^n's, and progressively worse far away. In particular, as x_* gets far from the training inputs $\mathbf{x}$, the GP prediction $m_*(x_*)$ is driven by the chosen asymptotes of $\hat{V}(t_k, \cdot)$ reflecting extreme extrapolation. Specifically, for GPs we have that $m_*(x_*) \to m(x_*)$ reverts to its prior mean.

Based on the above discussion, the design $\mathcal{D}_k$ should concentrate on the *region of interest*. In our context, the latter is driven by the pairs $(\bar{\mu}_{t_k}, \bar{\sigma}_{t_k})$ that are likely to be encountered by the controller at step k. In addition, since the key output we seek is the investment strategy $u^*(t_k, x) \in [0, 1]$, the hardest learning task is to identify the optimal control when the constraints are not binding. Therefore, we wish to set our time-dependent training region $\hat{R}_{t_k}$, i.e. the domain of approximation, to be the states x that are likely to be visited by the optimally controlled $X^{u^*}_{t_k}$ and where $0 < u^*(t_k, x) < 1$. Of course, the precise geometry of this ideal region of interest is a priori unknown.

Another problem that requires attention is extrapolation in estimating the surrogate $\hat{V}(t_k, \cdot)$. Extrapolation tends to lead to large estimation errors in approximating the true conditional expectation operator $\mathbb{E}$. With this in mind, we need to select the design $\mathcal{D}_k$ so that extrapolation is minimized. In order to compute $\hat{V}(t_k, y^n_{t_k}, \bar{\theta}^n_{t_k})$, we must consider $\hat{E}[\hat{V}(t_{k+1}, \mathbf{T}(t_k, x^n_{t_k}, u, \theta, \epsilon_{t_{k+1}}))]$ at any θ within $\Theta(t_k, \bar{\theta}^n_{t_k})$. This, in turn, means that we need good estimation of $\hat{V}(t_{k+1}, \mathbf{T}(t_k, x^n_{t_k}, u, \theta, \epsilon^{(i)}))$, $i = 1, \ldots, \mathfrak{I}$, where recall that $\epsilon^{(i)}$ comes from Gaussian quantization. Figure 2.1 displays the relationship between $\mathcal{D}_k$ and the set of all next-step locations $\{x_* \in \mathbf{T}(t_k, x^n_{t_k}, u, \check{\theta}(t_k, \bar{\theta}^n_k), \epsilon^{(i)})), n = 1, \ldots, N; \epsilon^{(i)} \in Quantizer_{\mathfrak{I}}\}$. These are the locations where we must evaluate $\hat{V}(t_{k+1}, x)$ and hence highlights to what extent the method requires extrapolation. On the right panel we compare all needed predictive sites θ_* against the $\bar{\theta}^{1:N}_{t_{k+1}}$ used for training $\hat{V}(t_{k+1}, \cdot)$.

There are two main concepts to go from a domain of approximation $\hat{R}_{t_k}$ to a discrete training set $\mathcal{D}_k$. The first idea is a "density-based" approach that aims to make $\mathcal{D}_k$ mimic the distribution of X_{t_k} under $\mathbb{Q}^{test}$. Recall that X_{t_k} includes the exogenous Y_{t_k} and the path-dependent $\bar{\theta}_{t_k}$. One method to handle this path-dependency is based on Control Randomization [1, 17] which was also the motivation for generating the path-simulations in Bielecki et al. [9]. We first pick some $\mathbb{Q}^0$ and generate paths under $\mathbb{Q}^0$. In other words, we pick some "true" model

parameters (e.g. using the prior mean estimates $\bar{\theta}_{t_0}$) that are used to generate $Y_{t_k}, \bar{\theta}_{t_k}$. A collection of N such independent "pilot" paths can be then used to set $\mathcal{D}_k := \{(y^n_{t_k}, \bar{\theta}^n_{t_k})\}$.

The second concept is to apply space-filling schemes which aim to yield a "uniform" sample from the region of interest $\hat{R}_{t_k}$. This approach is similar to classical maximin/max-entropy experimental designs in statistics. Space filling is achieved by specifying an input domain (usually some polyhedral or rectangular bounding box) and then a space-filling sequence. A popular choice are Quasi Monte Carlo (QMC) sequences that are used as a variance reduction tool to ensure that $\mathcal{D}_k$ does not contain any "holes" that might degrade estimation in their neighborhoods. A QMC sequence offers a deterministic way to sequentially fill the unit hypercube and can be straightforwardly scaled and clipped to fill any polyhedral domain. For example, Sobol QMC sequences offer a discrete $\mathcal{D}_k$ of any size N.

In Figure 2.1, we show an adaptive "mixture" design $\mathcal{D}_k$ that we utilize for the portfolio optimization problem with unknown (μ, σ). It is obtained in three steps. In the first step, we simulate $N'' = 250$ forward paths under a pre-specified $\mathbb{Q}^0$ that yield $(\bar{\mu}''_{t_k}, \bar{\sigma}''_{t_k})$. We then utilize these "pilot" paths to obtain a convex hull R'_{t_k}. In the second step, we utilize the Sobol QMC sequence to fill the respective R'_{t_k} with $N' = 200$ sites. Finally, in the third step, we augment with another $\tilde{N} = 56$ sites based on $R_{t_k} = \{x^n_{t_{k+1}} : \hat{u}(t_{k+1}, x^n_{t_{k+1}}) \in (0,1)\}$, i.e. add new sites where the control was non-trivial in the previous time-step.

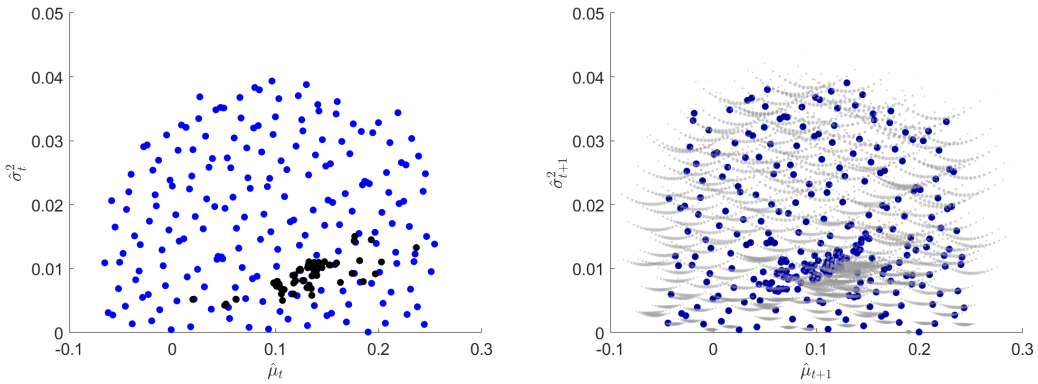


Figure 2.1: Constructed simulation design $\mathcal{D}_k = \{\bar{\mu}^{1:N}, \bar{\sigma}^{1:N}\}$ for the portfolio optimization case study. There are a total of 256 sites: Sobol QMC based (200 blue); adaptive based on $\hat{u}_{t_{k+1}}$ along pilot paths (56 black). The right panel shows the predictive locations $V(t_{k+1}, \cdot)$ used from $\mathcal{D}_k$, with the blue dots indicated the respective $\mathcal{D}_{k+1}$ to highlight inter- and extrapolation.

Remark: Given the complexity of (1.5) which requires solving a min-max problem at each design site $x^n_{t_k}$, it is not computationally feasible to have thousands of x^n's. Already for a rather small design like in Figure 2.1, we are looking at more than a million of `predict` calls to the GP surrogate *each time step* ($\mathfrak{I} = 100$ for each integral, multiplied by about 60 optimization steps to converge to the min-max optimal value, multiplied by $|\mathcal{D}| = N = 256$ design sites). This is a principal reason why we advocate Gaussian Process surrogates which

excel in learning from small-to-moderate designs. Note that GPs are still computationally intensive and their speed is very sensitive to N, which is the reason for all the above care in building a good experimental design. While a linear surrogate that utilizes ordinary least squares regression with a pre-specified set of basis functions is much faster, its rigid structure performs terribly for learning u^*; in our experience success of the algorithm hinges on having enough degrees of freedom. Other alternatives for value function approximation besides GPs are left for future research.

2.3. Algorithm. Algorithm 2.1 summarizes our procedure. It further makes a slight extension to allow for time-dependent design size N_k. The second Algorithm 2.2 evaluates the performance and strategy of the controller under some test measure $\mathbb{Q}^{test}$ utilizing a forward Monte Carlo average over N' fresh paths. The actual simulations are under $\mathbb{Q}^{\tilde{\theta}^n}$ where $\tilde{\theta}^n$ can vary across paths or even across time. For example, if we take $\tilde{\theta}^n \sim \Theta_0$ uniformly, then we mimic the static robust setup. The resulting out-of-sample estimator $\underline{V}$ is guaranteed to be a lower bound for $V(0, x_0)$ since it is directly based on the suboptimal strategy $\hat{u}(\cdot, \cdot)$ and its only error is the Monte Carlo averaging due to the finiteness of N'.

To summarize, the proposed framework organically integrates the construction of $\hat{V}$ and $\hat{u}$ with the paradigm of Design and Analysis of Computer Experiments that views the intermediate step as inference of an *expensive black-box* function. The adaptive $\mathcal{D}$'s target the learning of the optimal control, while the GP setup captures the spatial borrowing of information to predict $\hat{u}(t_k, x)$ without directly optimizing. Their combination improves efficiency, scalability, and interpretability, and should be contrasted to the conventional implementation where $\mathcal{D}$ is a grid and the approximation architecture $\mathcal{H}$ is a linear interpolant (piecewise linear $\hat{V}$). Our algorithm is also highly modular, allowing additional ways of approaching the aforementioned sub-problems of numerical integration and numerical optimization. Intuitively, the computation revolves around repeated optimization, and our framework achieves substantial gains by leveraging already obtained solutions of similar optimization problems, in analogy to *parametric optimization*. To our knowledge, we are the first article to propose this strategy for robust stochastic control.

Remark 2.4. *In the case studies below we have* $\mathcal{U} = [0, 1]$ *and the GP surrogate* $\hat{u}$ *tends to over-smooth in regions where* $\check{u}(t_k, x) \simeq 0$ *or* $\check{u}(t_k, x) \simeq 1$ *as it does not like flat responses. We regularize the constraint* $u \in [0, 1]$ *by carrying out optimization of* $\mathscr{F}_2(u, x)$ *over a larger domain* $u \in \tilde{\mathcal{U}} \supseteq \mathcal{U}$*. The relaxed maximizer* $\check{u}(x)$ *is then fed into the GP* $\hat{u}$ *representation of the optimal control, and the resulting prediction is* projected *as* $\mathfrak{P}_{\mathcal{U}}(\hat{u}) \in [0, 1]$*. This regularization makes sure that the data* $\check{u}^n_{t_k}$ *used to build the surrogate are smooth and do not have the non-smooth cut-offs at 0 and 1. In the first example below we take* $\tilde{\mathcal{U}} = [-0.2, 1.2]$.

2.4. Algorithm Stability. The overall algorithm complexity can be broken down into the overhead of fitting the surrogates and the cost of calling `predict` during the Bellman recursions. With a GP surrogate, the complexity of model fitting is $\mathcal{O}(N_k^3)$ at each time step, so $\mathcal{O}(\sum_k N_k^3)$ total. Each GP prediction requires $\mathcal{O}(N_k^2)$ effort and is employed $\mathfrak{I}$ times during each evaluation of the conditional expectation $\hat{E}$ and in turn is called N_{optim} times (which is state-dependent) by the nested optimizer that is solving for $\check{u}$. Overall, we obtain $\mathcal{O}(\mathfrak{I} \times N_{optim} \times \sum_k (N_k^3))$ complexity for Algorithm 2.1. Similarly we have $\mathcal{O}(\sum_k N_k^2 \times N')$

Algorithm 2.1 Backward Recursion to learn $\hat{V}$ and $\hat{u}$.

Require: Design sizes N_k

Initialize with terminal condition $\hat{V}(t_K, x) = G(y)\ \forall x$.

for $k = K-1, \ldots, 1$ **do**

Create a design $\mathcal{D}_k = (Y^n_{t_k}, \bar{\theta}^n_{t_k}) \equiv x^{1:N_k}_{t_k}$ that will be used to estimate $\hat{V}(t_{t_k}, \cdot)$.

for $n = 1, 2, \ldots, N_k$ **do**

Using an optimal Gaussian quantizer with $\mathfrak{I}$ terms set

$$\mathscr{F}_2(u, x^n_{t_k}) := \inf_{\theta \in \Theta(t_k, \bar{\theta}^n_{t_k})} \sum_{i=1}^{\mathfrak{I}} \hat{V}(t_{k+1}, \mathbf{T}(t_k, x^n_{t_k}, u, \theta, \epsilon^{(i),\mathfrak{I}}))w^{(i),\mathfrak{I}}. \tag{2.8}$$

Let

$$v^n_{t_k} := \sup_{u \in \mathcal{U}} \mathscr{F}_2(u; x^n_{t_k}). \tag{2.9}$$

Record the estimated optimal control $\check{u}^n_{t_k} \leftarrow \arg\sup \mathscr{F}_2(u; x^n_{t_k})$.

Record the driving worst-case parameters corresponding to $\check{u}^n_{t_k}$:

$$\check{\theta}^n_{t_k} \leftarrow \arg\inf_{\theta} \sum_{i=1}^{\mathfrak{I}} \hat{V}(t_{k+1}, \mathbf{T}(t_k, x^n_{t_k}, \check{u}^n_{t_k}, \theta, \epsilon^{(i),\mathfrak{I}}))w^{(i),\mathfrak{I}}.$$

end for

Build a GP model $\hat{V}(t_k, \cdot)$ for the link between $(x^{1:N_k}_{t_k})$ and $(v^{1:N_k}_{t_k})$; this is the functional representation of the value function at step k.

Build a GP model $\hat{u}(t_k, \cdot)$ for the link between $(x^{1:N_k}_{t_k})$ and $(\check{u}^{1:N_k}_{t_k})$; this is a (separate) functional representation of the optimal adaptive robust feedback control map

end for

complexity for generating N' forward paths to obtain the out-of-sample evaluation of $\underline{V}$ in Algorithm 2.2.

Note that while the main loop in Algorithm 2.1 is deterministic, in practice running the algorithm twice will generate slightly different solutions. This occurs because our simulation designs $\mathcal{D}_k$ are dependent on the random pilot paths, so that the training x^n's, and hence the GP surrogates, would vary from run to run. Moreover, many of the nonlinear optimizers that are necessary for surrogate fitting rely on internal randomization. Algorithm variance would be amplified if also the conditional expectation is approximated via Monte Carlo rather than Gaussian quadrature.

In this vein, the design size N_k plays a double role of (i) decreasing the above macro-replication variance through Law of Large Numbers with respect to the randomized aspects of $x^{1:N}_{t_k}$; (ii) increasing accuracy (i.e. reducing bias) by raising the fidelity of $\hat{u}$ and $\hat{V}$ (i.e. reducing functional approximation error between the surrogate and the true minimizers/value function).

Algorithm 2.2 Forward Monte Carlo to evaluate performance of u^* under $\mathbb{Q}^{test}$

Require: No. of simulations N', no. of time-steps K, initial $(X_{t_0}, \bar{\theta}_{t_0})$.

1: Initialize $\bar{\theta}^n_{t_0} \leftarrow \bar{\theta}_{t_0}, x^n_{t_0} \leftarrow x_{t_0},\ n = 1, \dots N'$
2: Set $\tilde{\theta}^n$ which is the parameter set (possibly time-dependent) for the n-th forward path (so simulated under $\mathbb{Q}^{\tilde{\theta}^n}$).
3: **for** $k = 0, \dots, K-1$ **do**
4: Draw i.i.d. $\epsilon^n_{t_{k+1}} \sim \mathcal{N}(0,1)$, $n = 1, \dots, N'$
5: Using the GP surrogate compute the control $u^n_{t_k} \leftarrow \hat{u}(t_{t_k}, x^n_{t_k})$.
6: (Optional) Evaluate (using prediction from the surrogate or a direct evaluation of the optimizer) the worst-case parameters $\hat{\theta}^n_{t_k}$ which are functions of $x^n_{t_k}$;
7: Compute the realized payoff $g(t_k, y^n_{t_k}, u^n_{t_k})$; update the cumulative $C^n_k \leftarrow \sum_{\ell=1}^{k} g(t_\ell, y^n_{t_\ell}, u^n_{t_\ell})$;
8: Update the states according to $x^n_{t_{k+1}} \leftarrow \mathbf{T}\left(t_k, x^n_{t_k}, u^n_{t_k}, \tilde{\theta}^n, \epsilon^n_{t_{k+1}}\right)$.
9: **end for**
10: Return $\underline{V}(0, x_0) := \frac{1}{N'} \sum_{n=1}^{N'} C^n_K$

Figure 2.2 displays a boxplot of $\hat{u}$ across algorithm macro-replications when using $N_k \equiv 100$ vs $N_k \equiv 250$. As expected, a larger design reduces overall variance, as well as the bias.

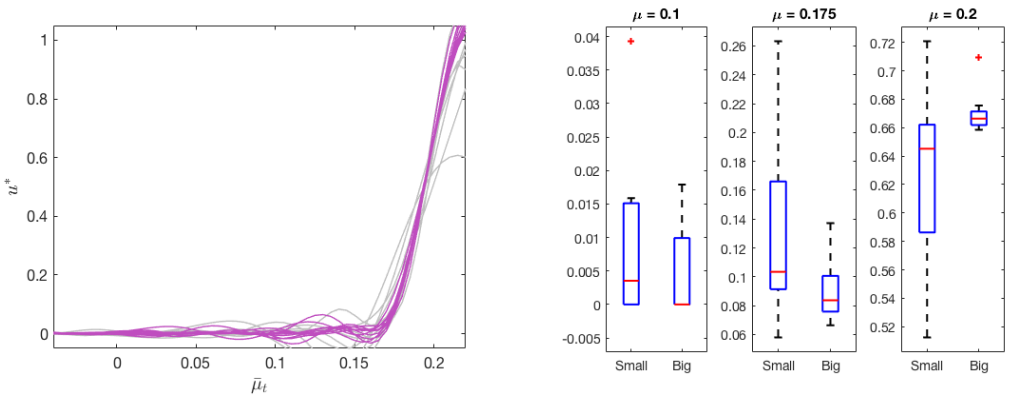


Figure 2.2: Comparison of Monte Carlo standard errors as a function of design size $N \equiv N_k \forall k$. Left: Ten empirical predictions $\hat{u}(t_k, \bar{\mu}_{t_k}, \bar{\sigma}_{t_k})$ at $t_k = 0.8$, $\bar{\sigma}^2_{t_k} = 0.01$ as a function of $\bar{\mu}_{t_k}$ to illustrate the intrinsic fluctuations of our algorithm. We show both low-budget (in purple) and high-budget (in grey) runs. Right: boxplot for $\hat{u}(t_k, \bar{\mu}_{t_k}, 0.01)$ at three different $\bar{\mu}_{t_k}$.

A detailed error analysis of Algorithm 2.1 is beyond the scope of this work. We note that the scheme makes several approximations that all interact with each other and moreover back-propagate over the Bellman recursion in t_k's. Indeed, the solution quality is impacted by the surrogate quality of $\hat{V}(t_k, \cdot)$ and of $\hat{u}(t_k, \cdot)$ (that can be viewed as a 2-layer approximation), with the latter also linked to the optimization and quantization errors in (2.9). Thus, there

$r = 0.02,\ T = 1,\ \Delta t = 0.05$
$\alpha = 0.1,\ \gamma = 4,\ \mathfrak{I} = 100, N = 250$

Table 3.1: Parameters for the portfolio optimization case study.

are 3 major sources of error: (i) how well do the GPs match the true underlying input-output relationships (which links to the experimental design $\mathcal{D}_k$ and N_k as well); (ii) how close are $\check{u}_{t_k}^{1:N}$ and $v_{t_k}^{1:N_k}$ to the true $V(t_k, x_{t_k}^{1:N_k})$ and $u^*(t_k, x_{t_k}^{1:N_k})$ and (iii) how well does the quantized sum on the right-hand-side of (2.8) approximate the conditional expectation of $\hat{V}(t_{k+1}, \cdot)$. Unraveling the interaction between the errors already embedded in $\hat{V}(t_{k+1}, \cdot)$ and the saddle-point computation that yields $\check{u}_{t_k}^n$ is one of the many thorny challenges to investigate.

3. Adaptive Robust Utility Maximization. In this section we illustrate our algorithm from Section 2.3 for the task of portfolio optimization under uncertain drift and volatility. The full list of parameters is given in Table 3.1. We present two different case studies.

In this setting, we can simplify the computation of $\mathscr{F}_2$ in (2.8) by dimension reduction. Namely, by monotonicity we take the 1-dimensional infimum in the unknown parameters μ, σ just over the boundary $\partial\Theta$ of the parameter space. Let $\check{\mu}(x_{t_k}^n), \check{\sigma}(x_{t_k}^n)$ denote the parameters achieving the infimum of $\mathscr{F}_2$. Switching to polar coordinates it suffices to find and record the corresponding angle $\varphi_{t_k}^n$ using the map

$$\check{\mu}_{t_k}^n := \bar{\mu}_{t_k}^n + \sqrt{\kappa k^{-1}}\bar{\sigma}_{t_k}^n \cos(\varphi_{t_k}^n); \tag{3.1}$$

$$(\check{\sigma}_{t_k}^n)^2 := (\bar{\sigma}_{t_k}^n)^2(1 + \sqrt{2 \cdot \kappa k^{-1}} \sin(\varphi_{t_k}^n)) \vee 0. \tag{3.2}$$

The minimization defining $\mathscr{F}_2$ (and hence $\varphi_{t_k}^n$) is then done over $\varphi \in [0, 2\pi]$ using the default `fminbnd` algorithm in Matlab with a tolerance threshold of 10^{-6}. Similarly, we use `fminbnd` to minimize $\mathscr{F}_2$ over the scalar control domain $\mathcal{U} = [0, 1] = [u_{\min}, u_{\max}]$.

3.1. Static Investment and Hedging. It is instructive to consider the one-period, aka *static* version of the optimal investment problem. In this simplified setting, the terminal controlled wealth Y_T under $\mathbb{Q}^{\mu,\sigma}$ is given by

$$Y_T^u = Y_{t_0}(1 + r(T - t_0) + u(e^{\mu(T-t_0)+\sigma\sqrt{T-t_0}\epsilon_T} - 1 - r(T - t_0))), \tag{3.3}$$

where t_0 is the date that trade happens. Above ϵ_T is standard Gaussian in $\mathbb{R}$ and due to the scaling of the power utility function, we may take without loss of generality $Y_{t_0} = 1$. The static optimal investment problem is

$$V(t_0, \bar{\mu}, \bar{\sigma}) = \sup_{u\in[0,1]} \inf_{(\mu,\sigma)\in\Theta(\bar{\mu},\bar{\sigma},t_0,\kappa)} \mathbb{E}^{\mu,\sigma}\left[\frac{Y_T^{1-\gamma}}{1-\gamma}\right] \tag{3.4}$$

with the uncertainty set

$$\Theta(\bar{\mu}, \bar{\sigma}, t_0, \kappa) = \left\{(\mu, \sigma) \in \mathbb{R}^2 : \frac{k_0 + 1}{\bar{\sigma}^2}(\bar{\mu} - \mu)^2 + \frac{k_0 + 1}{2\bar{\sigma}^4}(\bar{\sigma}^2 - \sigma^2)^2 \le \kappa\right\}. \tag{3.5}$$

Observe that due to the one-period feature, the role of $\bar{\mu}, \bar{\sigma}$ is purely to determine the set of $\mathbb{Q}^\theta$'s that are being considered. Moreover, without loss of generality we may take the horizon $T - t_0 = 1$ to be fixed, whereby the size of $\Theta(\bar{\mu}, \bar{\sigma}, t_0, \kappa)$ is controlled by the ratio $\alpha' := \kappa/(k_0 + 1)$ which is interpreted as the measure of robustness. As $\alpha' \to 0$, $\Theta(\bar{\theta}) \to \{\bar{\theta}\}$, and α' is the "radius" of adversity.

To solve (3.4) it suffices to evaluate the map $(u, \mu, \sigma) \mapsto \mathbb{E}^{\mu,\sigma}\left[\frac{Y_T^{1-\gamma}(u)}{1-\gamma}\right]$ and then find its saddle point (i.e. the $\sup - \inf$ location) over the constrained domain $u \in \mathcal{U}$ and $(\mu, \sigma) \in \Theta$. Due to the concavity of the utility preferences, this map is increasing in μ and decreasing in σ. Consequently, the infimum over the robustified beliefs will always be achieved in the NW quadrant, so that $\check{\mu}(\bar{\mu}, \bar{\sigma}) \leq \bar{\mu}$ and $\check{\sigma}(\bar{\mu}, \bar{\sigma}) \geq \bar{\sigma}$. Therefore we can again switch to polar coordinates reducing to a 2-dimensional function $F(u, \varphi)$ of the control u and the angle $\varphi \in [\pi/2, \pi]$. For example, $\varphi = \pi$ means taking the minimum possible drift.

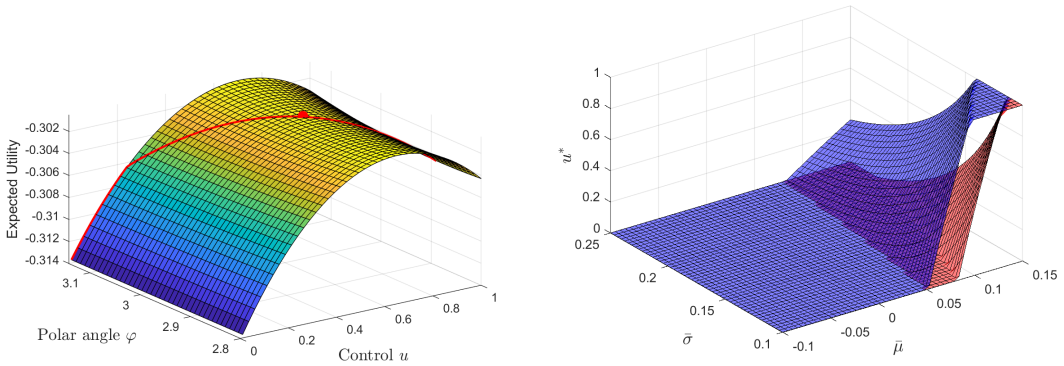


Figure 3.1: *Left:* $F(u, \varphi)$ for the static portfolio optimization case study for a fixed $\bar{\mu}, \bar{\sigma}$. The red line denotes the profile minimizers $u \mapsto \check{\varphi}(u)$. The red dot denotes the saddle point (u^*, φ^*) identified via $u^* = \arg\max_{u \in [0,1]} F(u, \check{\varphi}(u))$. *Right:* $u^* \in [0, 1]$ as a function of $\bar{\mu}, \bar{\sigma}$ for two different robustness levels $\kappa = 4.61$ (red), $\kappa = 1.39$ (blue) and $k_0 = 10$.

The left panel in Figure 3.1 shows $F(u, \varphi)$ for a representative pair of initial beliefs $\bar{\mu}, \bar{\sigma}$ in the range $u \in [0, 1], \varphi \in [2.8, \pi]$. For a fixed investment level $u > 0$, $F(u, \cdot)$ is a convex function with a minimizer $\check{\varphi}(u)$. However note that when $u = 0$, Y_T is independent of the S-dynamics, i.e. $F(0, \cdot)$ is constant in φ. The respective infimum is then undefined and there is no $\check{\varphi}(0)$. The saddle-point is the maximum $\arg\max_u F(u, \check{\varphi}(u))$ and is indicated by the red dot on the left panel, where $u \mapsto F(u, \check{\varphi}(u))$ is visualized by the red line. In this particular case, $u^*(\bar{\mu}, \bar{\sigma}) \simeq 0.669$ and $\varphi^*(\bar{\mu}, \bar{\sigma}) \simeq 2.998$.

The right panel of Figure 3.1 displays u^* as a function of the initial beliefs $\bar{\mu}, \bar{\sigma}$ for two different robustness levels. As α decreases, Θ gets larger and therefore the worst-case (inner infimum) becomes worse. Consequently, the controller acts more conservatively, i.e. u^* increases in α. Otherwise, u^* has the familiar shape of increasing in $\bar{\mu}$ (asset returns being more favorable) and decreasing in $\bar{\sigma}$ (asset risk rising). Note that when returns are not sufficiently high and/or volatility is too large, the optimal action is $u^* = 0$, i.e. to invest only in the

risk-free bond. This happens in particular when $\Theta(\bar{\mu}, \bar{\sigma}, t_0, \kappa) \bigcap \mathbb{R}^2_- \neq \emptyset$ intersects with the negative half-plane, i.e. the investor believes that negative returns $\mu < 0$ are possible.

The left panel of Figure 3.2 visualizes the structure of the robustified beliefs (μ^*, σ^*) as a function of $(\bar{\mu}, \bar{\sigma})$. Recall that $\varphi^* = \pi$ (quivers pointing to the West) corresponds to the worst-case making μ^* as negative as feasible. This is the robustified belief in the middle and top of the state space. In the bottom-right (very high $\bar{\mu}$ and very low $\bar{\sigma}$), the worst-case trades off reduced returns against increased risk. Finally, in the left half of the plot we have $u^* = 0$, so that there is no φ^* (no quivers displayed) since $F(0, \cdot)$ is constant. In that case the consistent interpretation is to keep $\mu^* = \bar{\mu}, \sigma^* = \bar{\sigma}$ unchanged.

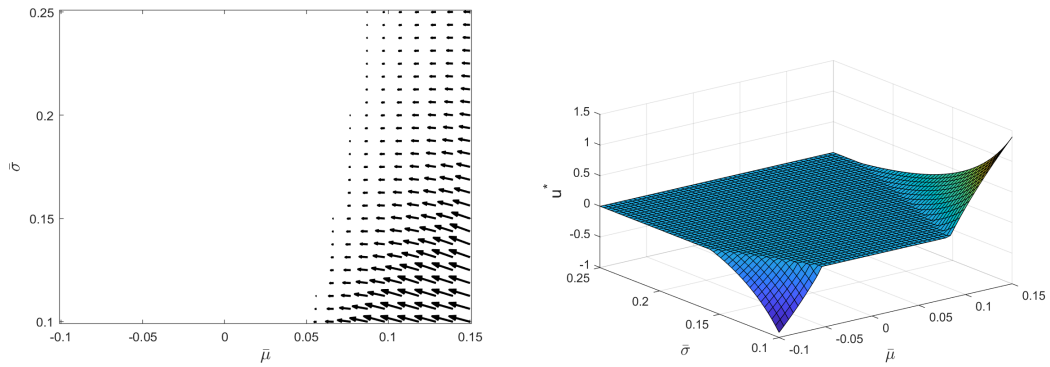


Figure 3.2: *Left panel:* quiver plot showing the relative position of $(\check{\mu}, \check{\sigma})$ relative to $(\bar{\mu}, \bar{\sigma})$. The angle of the quivers is the robustified belief angle φ^* and the length is proportional to u^*; no quiver is shown when $u^* = 0$. *Right:* optimal investment fraction u^* as a function of $\bar{\mu}, \bar{\sigma}$ using unconstrained optimization over $\mathbb{R}$.

Finally, the right panel of Figure 3.2 addresses the question of trading constraints. Typically we find that $u \mapsto F(u, \check{\varphi}(u))$ is convex, i.e. has a unique global maximum. In that case we may find the unconstrained global maximum u^*_{unc} and then constraints of the form $u \in [u_{\min}, u_{\max}]$ translate into $u^*_{con} = u_{\min} \vee u^*_{unc} \wedge u_{\max}$. The plot reveals several useful insights. First, we observe the "shrinkage to zero" effect: robust control implies that a risky position will be assumed only if there is a clear understanding regarding μ. Therefore, if $|\bar{\mu} - r|$ is small, the robust solution is to take $u^* = 0$. In that sense, u^* shrinks the non-robust control u^*_{adpt} to zero. Because the size of $\Theta(t_k,)$ is influenced by $\hat{\sigma}$ the amount of this shrinkage grows in $\hat{\sigma}$ as observed in the plot. Second, we observe that for a fixed $\bar{\sigma}$ u^* is essentially piecewise linear in $\bar{\mu}$, capturing the original Merton intuition that u^* is linear in asset returns. Third, we note that leverage and short-selling remain feasible with adaptive robust control provided that the adversarial set is far enough to one side of the half-plane. Thus, we observe $u^* < 0$ in the bottom-left corner (strongly negative Sharpe ratio) and $u^* \gg 1$ in the bottom-right corner (very high Sharpe ratio). Fourth, we stress that the plot is (roughly) symmetric about $\bar{\mu} = r$ which is the risk-free alternative.

3.2. Multi-period Optimal Investment. We now return to the original multi-period setting with $K = 20$ time-steps, i.e. $t_k = t_0 + k\Delta t$ and $T = t_K$. This setup is very similar to

the previous section, except that $\hat{u}(t_k, x)$ is now time-dependent. Figure 3.3 illustrates this dependence by showing a contour-plot of $(\bar{\mu}., \bar{\sigma}.) \mapsto \hat{u}(\cdot, \bar{\mu}, \bar{\sigma})$ for two different time steps t_k. Recall that at early epochs Θ_k is larger so ceteris paribus the investor is more conservative and $\hat{u}$ tends to increase in t_k. This is indeed observed in the figure, matching the intuition of "learning-as-you-go". We note that there are competing effects, in particular due to the time-dependence in the dynamics of $\bar{\theta}_{t_k}$ (learning slows over time) and the time-dependent effective risk-aversion (the value function becomes less concave as $T - t_k$ increases).

Structure of Optimal Control: The left panel of Figure 3.3 shows the estimated optimal feedback control surface $(\bar{\mu}, \bar{\sigma}) \mapsto \hat{u}(t_k, x)$. We observe that $\hat{u}(t_k, x) = 0$ when $\bar{\mu}$ is low and monotonically increases as the posterior mean asset return $\bar{\mu}_{t_k}$ rises. Eventually for high enough $\bar{\mu}_{t_k}$, $\hat{u}(t_k, x) = 1$. This structure suggests that we can concentrate the statistical modeling efforts in the intermediate region R where $\hat{u}(t_k, x) \in (0, 1)$ since otherwise the corresponding feedback control surface is completely flat and therefore easy to interpolate. The latter fact also implies that even extrapolation is feasible since we just need to ensure that the surrogate is set such that the respective asymptotes in $\bar{\mu}$ are 0 to the left and +1 to the right.

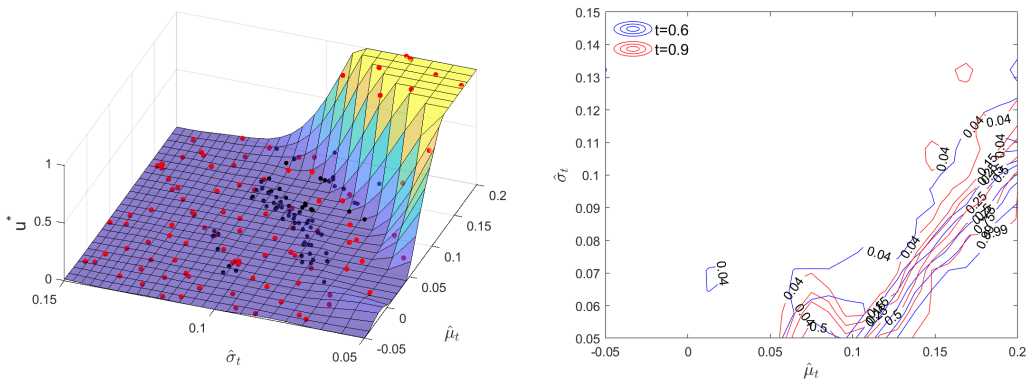


Figure 3.3: Left: Optimal investment strategy from the GP surrogate, $\hat{u}(t_k, x)$ at $t_k = t_0 + 15\Delta t$. Right: time-dependency of $\hat{u}(t_k, \cdot)$: comparing $k = 15$ (cyan) and $k = 10$ (blue).

Structure of Robustified Beliefs: We recall that the angle $\hat{\varphi}(t_k, x)$ is only well-defined when $\hat{u}(t_k, x) \neq 0$, since otherwise the investor is not exposed to returns uncertainty. Therefore, $\hat{\varphi}(t_k, x)$ only exists for $\bar{\mu}_{t_k}$ high enough and $\bar{\sigma}_{t_k}$ low enough (the SE corner of the state space). In the latter region, we find that the principal effect is to decrease the drift rather than to increase volatility, so that $\hat{\varphi} \simeq \pi$. In the multi-period context, low $\bar{\sigma}_{t_k}$ would also hurt the investor, so $\hat{\varphi} > \pi$ will also occur at the early time steps.

3.3. Distribution of Terminal Wealth. To obtain an out-of-sample performance metric of the computed investment strategy $\hat{u}$ we apply Algorithm 2.2 in Section 2.3. For the test measure $\mathbb{Q}^{test}$ we use $\mathbb{Q}^{\Theta_0}$, i.e. static Knightian uncertainty where $\tilde{\theta}^n$ is drawn from a known prior Θ_0 at $t = t_0$ and then kept constant through time. To be consistent, we utilize the initial adversarial set $\Theta_0 = \Theta_\alpha(t_0, \bar{\theta}_{t_0})$. Specifically, for the forward simulations we sample

independently and uniformly $\tilde{\theta}^n$ for $n = 1, \ldots, N'$. For the discussion below, we took $\tilde{\mu}^n \sim \mathcal{N}(0.15, 0.02^2), \tilde{\sigma}^n = 0.1$ and $\bar{\theta}_{t_0} = (0.1, 0.08)$ so the investor starts out under-estimating both the mean returns and the return volatility.

We recall that the classical Merton solution with fixed (μ, σ) is time stationary and given by

$$u^*(t,x) = u^*(\mu,\sigma) = \frac{\mu - r}{\gamma\sigma^2}.$$

Using the above we can compare our adaptive robust solution to the following alternative strategies:

- Merton strategy based on $u^*(\hat{\theta}(t_0, \bar{\theta}_{t_0}))$ which is the static robust formulation;
- Merton strategy based on $u^*(\bar{\theta}_{t_k})$ which is the adaptive formulation;
- Strategy equivalent to our adaptive robust algorithm with $\alpha = 1$ ($\Theta_\alpha(t_k, \bar{\theta}_{t_k}) = \{\bar{\theta}_{t_k}\}$) which dispenses with robustness.

For the static robust approach, the worst case of asset dynamics corresponds to a low Sharpe ratio. In particular, for our parameter values, since negative return rate cannot be ruled out $\Theta_0 \bigcup (-\infty, 0) \times \mathbb{R}_+ \neq \emptyset$, the robust $u^{(SR,*)}(t,x) = 0$ for all $t \in \mathcal{T}\backslash\{T\}$, and no investment would be ever undertaken. This point clearly illustrates why learning is desirable, as it allows the investor to overcome initial pessimism (or worry about model risk) through gradual collection of information. If enough favorable information is learned then she will eventually become optimistic enough to invest. At the other extreme, the adaptive case leads to over-confidence and hence over-investment. In particular, the adaptive method generates much larger variance, i.e. risk, in Y_T compared to adaptive robust, as the investor trusts her myopic beliefs too much.

Figure 3.4 shows the distribution of terminal wealth W_T as we vary the risk aversion parameter γ and the robustness parameter α. As expected, less risk averse investors will have higher u^* and therefore larger $\mathbb{E}[W_T]$ accompanied by much larger $\mathbb{V}ar(W_T)$. Similarly, agents that are less conservative (and therefore have smaller $\Theta_k(\bar{\theta}_{t_k})$) will tend to again invest more. However, being too "trusting" and not guarding against model mis-specification will eventually hurt wealth accumulation (compare $\alpha = 0.5$ to $\alpha = 0.2$, so that $\mathbb{E}^{\mathbb{Q}^{\Theta_0}}[W_T]$ has a hump shape as a function of α).

4. Adaptive Robust Optimal Hedging. It is of financial and theoretical importance to study optimal hedging problem in an incomplete market setup under model uncertainty. The uncertainty issue was addressed, for example, in the paper [19] where the author used Bayesian estimators in place of the corresponding parameters. We will compare our approach to a similar method that uses maximum likelihood estimators instead. Another standard method widely used among banks when dealing with model risk is the so-called conservative or worst-case approach, and it is in principle the same as the static robust approach.

We denote by $S = \{S_t, t \in \mathcal{T}\}$ the market price process of the stock and we postulate that the increments of S are log-normal with parameters μ, σ:

$$S_{t_{k+1}} = S_{t_k} \exp\left(\mu\Delta t + \sigma\sqrt{\Delta t}\epsilon_{t_{k+1}}\right), \qquad \epsilon_{t_{k+1}} \sim \mathcal{N}(0,1). \tag{4.1}$$

Dynamic trading strategies are based on a discount bond and the stock S, and are represented by a predictable discrete-time process $\{u_{t_k},\ t_k \in \mathcal{T}'\}$ identified with the number of shares invested in the stock from t_k until t_{k+1}. The bond investment is then determined from the

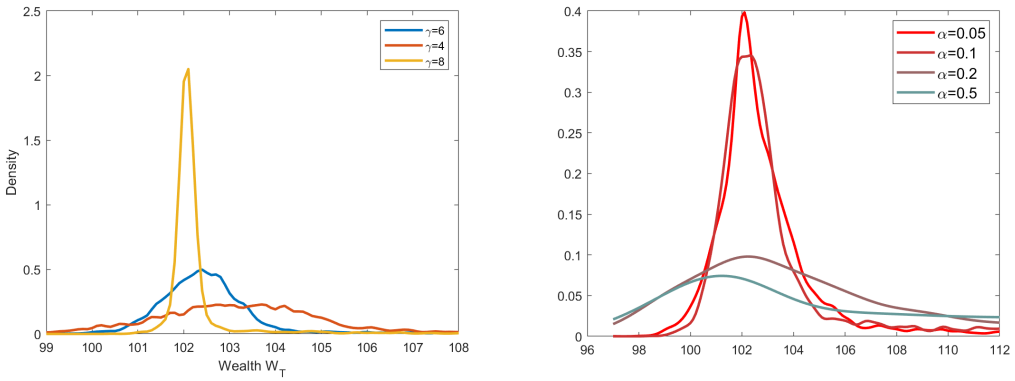


Figure 3.4: Left: Distribution of W_T for different risk aversion parameters γ. Right: Distribution of W_T for different robustness parameters α.

self-financing constraint. For an admissible $\vec{u}$ we define the corresponding wealth process W as

$$W_{t_{k+1}} = W_{t_k} + u_{t_k}(S_{t_{k+1}} - S_{t_k}) = W_{t_k} + u_{t_k} S_{t_k}(e^{\mu\Delta t + \sigma\sqrt{\Delta t}\epsilon_{t_{k+1}}} - 1), \quad W_{t_0} = w, \tag{4.2}$$

where w is the initial endowment, assumed to be fixed and deterministic.

We are interested in dynamic hedging, using an admissible trading strategy, of a European style option written on the stock S, maturing at time T and with the payoff $\Phi(S_T)$, where Φ is a measurable function. To address nonlinear hedging, we define the objective as optimization of the hedging *error* $H = \Phi(S_T) - W_T$ through [24]

$$\mathbb{E}\left[\ell(\Phi(S_T) - W_T)\right] \mapsto \min! \tag{4.3}$$

where $\ell(\cdot)$ is a positive measurable *loss function* such that $\ell(0) = 0$. This setup offers a loss-based criterion that is applicable both in complete and incomplete markets. The main loss function we will consider is of the form

$$\ell(h) = h^+ + \lambda h^-, \tag{4.4}$$

where the constant $\lambda \geq 0$ reflects the relative importance of super- and sub-hedging. When $\lambda = 0$ the investor is penalized only for not having enough to cover her liability, while $\lambda = 1$ leads to $\ell(h) = |h|$ and is the L^1-equivalent of the classical quadratic hedging criterion.

We consider adaptive robust hedging in the context of uncertain asset drift and volatility, leading to a four-dimensional state: $x = (W, S, \bar{\mu}, \bar{\sigma})$ which under $\mathbb{Q}^{\mu,\sigma}$ has the dynamics prescribed by (4.1)-(4.2)-(1.9)-(1.10). The Bellman equation becomes

$$\begin{aligned} V(T, x) &= \ell(\Phi(S) - W), \\ V(t_k, x) &= \inf_{u\in\mathcal{U}} \sup_{\theta=(\mu,\sigma)\in\Theta(t_k,\bar{\mu},\bar{\sigma})} \mathbb{E}\left[V(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon_{t_{k+1}}))\right]. \end{aligned} \tag{4.5}$$

$r = 0,\ T = 1,\ \Delta t = 0.1,\ k_0 = 150$
$\alpha = 0.1,\ \kappa = 4.61,\ \mathfrak{I} = 40,\ N = 250$
$\bar{\mu}_{t_0} = 0.12,\ \bar{\sigma}_{t_0} = 0.4,\ S_0 = 100,\ \mathcal{K} = 100$

Table 4.1: Parameters for the optimal hedging case study.

We may interpret V as the robust expected loss, so that $V(t_k, x) \geq 0$ and lower V implies better ability to ℓ-hedge. For example, in the one-sided example with $\ell(h) = h^+$, $V(t_k, \cdot)$ will be monotone decreasing in both S and W.

This setup is conceptually similar to the previous section but presents a harder computational challenge due to its higher dimensionality and the non-smooth relationships between $V(t_k, \cdot)$ and the state variables. Moreover, it features the fully endogenous wealth process W which cannot be reasonably "forward-simulated" without assuming u_{t_k}. To overcome this issue we adopt the Control Randomization approach of selecting a $\mathbb{Q}^0$ measure to build the designs $\mathcal{D}_k$.

For illustration, we henceforth consider a European style Call option written on the risky asset S with strike $\mathcal{K}$, $\Phi(S) = (S - \mathcal{K})^+$. The investor is short (sold) the Call and aims to minimize her hedging error starting with initial endowment w.

4.1. Implementation Details. For the hedging problem, x is four-dimensional, and there is a high correlation between S_{t_k} and $\bar{\mu}_{t_k}$, so that the design $\mathcal{D}_{t_k}$ has a certain 3-d structure. Moreover, necessarily there will be a dependence between S_{t_k} and wealth W_{t_k}. To construct $\mathcal{D}_k$ we use a three-step procedure. First, we generate "pilot" forward paths of X under $\mathbb{Q}^0$. The pilot paths are based on randomized $\bar{\mu}_{t_0}^{1:N_0}$, $\bar{\sigma}_{t_0}^{1:N_0}$, as well as initial stock prices $S_0^{1:N_0}$, where $N_0 = 250$. Specifically, given μ_0, σ_0, K we sample initial beliefs $\bar{\mu}_{t_0}^{1:N_0}$, $\bar{\sigma}_{t_0}^{1:N_0}$ uniformly from $[0.5\mu_0, 1.5\mu_0]$ and $[0.6\sigma_0, 1.3\sigma_0]$, respectively, as well as $S_{t_0}^{1:N_0} \in [0.5\mathcal{K}, 2\mathcal{K}]$ around the strike price $\mathcal{K}$. Second, we construct a set of augmented pilot sites by adding a few more $(\bar{\mu}, \bar{\sigma})$'s based on the edges of adversarial beliefs $(\mu, \sigma) \in \partial\Theta(t_k, \bar{\mu}_{t_k}^n, \bar{\sigma}_{t_k}^n)$. This step helps us reduce extrapolation when evaluating $\hat{V}(t_{k+1}, \mathbf{T}(t_k, x, u, \theta, \epsilon^{(i)}))$ during Gaussian quadrature. Third, we use a space-filling strategy to fill in "holes", employing a QMC sequence-based design over the convex hull of the augmented pilot sites at each t_k. This is done by "pulling forward" the design $\mathcal{D}_k$ to the next time-step $t_{t_{k+1}}$ using the map $\mathbf{T}(t_k, x_{t_k}^n, u, (\mu, \sigma), \epsilon_{t_{k+1}}^{(i)}))$ with the empirically sampled ϵ's and then randomly replacing 100 pilot sites by sites from the respective QMC (Sobol) sequence in $\mathbb{R}^3$. This yields the experimental design $(S_{t_k}^{1:N}, \bar{\mu}_{t_k}^{1:N}, \bar{\sigma}_{t_k}^{1:N})$, $N = 250$.

Finally, to specify $W_{t_k}^{1:N}$ we compute the Black Scholes prices $P^{BS}(t_k, S_{t_k}^n; \bar{\theta}_{t_k}^n)$ and independently sample $W_{t_k}^n \sim Unif(0.5P(t_k, S_{t_k}^n), 1.5P^{BS}(t_k, S_{t_k}^n))$. This leads to a "tube" in the (S, W) plane that contains the Black Scholes prices, with the idea that these are the likely wealth levels (assuming that $W_{t_0} \approx P^{BS}(t_0, S_{t_0}; \bar{\theta}_{t_0})$) to be visited.

The algorithm described in Section 2.3 is tailored for applying the adaptive robust framework to the optimal hedging problem. For the hedging problem, monotonicity of

$$\hat{E}[\hat{V}(t_{k+1}, \mathbf{T}(t_k, x_{t_k}^n, u, (\mu, \sigma), \epsilon_{t_{k+1}}))]$$

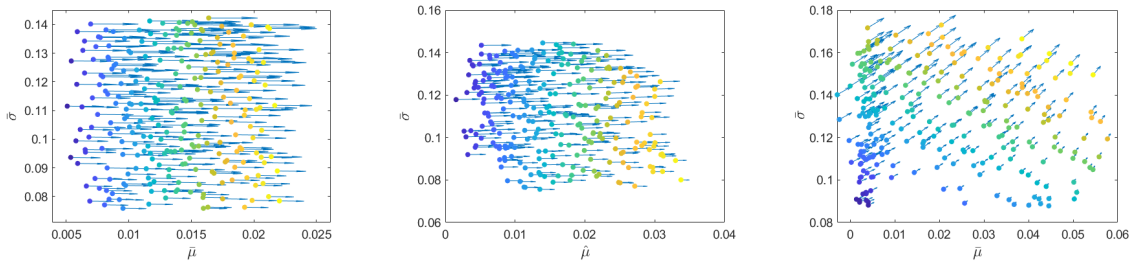
in terms of μ or σ no longer holds true. Hence, the supremum is not necessarily attained on the boundary $\partial\Theta(t_k, \bar{\mu}^n_{t_k}, \bar{\sigma}^n_{t_k})$. Nevertheless, we continue to take advantage of the elliptical nature of $\Theta(t_k, \cdot)$ working in the respective polar coordinates (φ, ρ) representing the angle and distance from $(\bar{\mu}, \bar{\sigma})$. We then discretize $\Theta(t_k, \bar{\mu}^n_{t_k}, \bar{\sigma}^n_{t_k})$ uniformly in terms of $\varphi \in [0, 2\pi]$ and $\rho \in (0, \kappa)$ and fixing $u \in \mathcal{U}$ carry out a direct maximization over the discrete set Θ' of adversarial beliefs. This finally yields the robustified parameters $(\check{\mu}, \check{\sigma})$:

$$\check{\mu}^n_{t_k} = \bar{\mu}^n_{t_k} + \sqrt{\check{\rho}^n_{t_k} \cdot (\bar{\sigma}^n_{t_k})^2/(k+k_0)}\cos(\check{\varphi}^n_{t_k})$$
$$\check{\sigma}^n_{t_k} = \bar{\sigma}^n_{t_k}\sqrt{1 + \sqrt{2 \cdot \check{\rho}^n_{t_k}/(k+k_0)}\sin(\check{\varphi})} \vee 0.$$

Minimization of the resulting function

$$\mathscr{F}_2^{hedge}(u; t_k, S^n_{t_k}, W^n_{t_k}, \bar{\theta}^n_{t_k}) := \sup_{(\mu,\sigma)\in\Theta(t_k,\bar{\mu}^n_{t_k},\bar{\sigma}^n_{t_k})} \hat{E}[\hat{V}(t_{k+1}, \mathbf{T}(t_k, x^n_{t_k}, u, (\mu,\sigma), \epsilon_{t_{k+1}}))] \tag{4.6}$$

with respect to $u \in \mathcal{U}$ is done by using `fminbnd` in Matlab with a tolerance threshold of 10^{-6}, where we double-check whether the minimum is achieved at the boundaries $u \in \{0, 1\}$. For the super-hedging loss function $\ell(h) = h^+$, we note that when $W \gg P^{BS}(t_k, S; \bar{\theta})$, the expected loss is numerically zero, so that $V(t_k, S, W)$ is numerically constant and the maximizer $\check{u}(t_k, x)$ of (4.6) is ill-defined. This is the scenario where the option can be super-hedged with very high probability so it does not matter what the current hedge is. We found that it helps to detect such cases a priori, setting $\check{u}(t_k, x) = 0$ for them.



$t = \Delta t$ $t = 4\Delta t$ $t = 9\Delta t$

Figure 4.1: Estimated angle $\check{\varphi}$ (quiver direction) and distance $\check{\rho}$ (quiver length) for the robustified model parameters $\check{\theta}$ as a function of current beliefs $(\bar{\mu}, \bar{\sigma})$ for the hedging problem with $S_t = 100$, $W_t = P^{BS}(t_k, 100)$. The quivers are color coded in terms of $\hat{u}(t, \bar{\theta}_t)$ (purple is $\hat{u}(t, \bar{\theta}) = 0$, yellow is $\tilde{u}(t, \bar{\theta}) = 1$) that is based on a GP surrogate.

Figure 4.1 shows the estimated angle $\check{\varphi}(t_k, \bar{\mu}, \bar{\sigma})$ and distance $\check{\rho}(t_k, \bar{\mu}, \bar{\sigma})$ via a quiver plot at three different time instances $t_k = k\Delta t$. A first observation is that $\check{\rho}$ is not trivial as the adversarial beliefs are not necessarily on the boundary. Secondly, in this problem $\check{\varphi}$ depends both on the current stock price S_{t_k} and the current wealth W_{t_k}. In the plot we consider at-the-money case $S_{t_k} = \mathcal{K} = 100$ and at-the-wealth case $W_{t_k} = P^{BS}(t_k, S_{t_k})$. We observe

that for early t_k, the worst-case adversarial $\check{\theta}$ corresponds to maximizing the stock return μ, $\check{\varphi}(t_1, \cdot) \approx 0$, so as to maximize the Call value. Asset volatility begins to play a more important role as expiration approaches and the Vega/Gamma of the Call increase. As a result, $\check{\varphi}$ rotates counterclockwise and the worst-case $\check{\theta}$ now involves making *both* μ and σ larger. Third, the Figure indicates the impact of t_k on $\hat{u}(t_k, \cdot)$: the distribution of $\hat{u}$ follows the rotation of $\check{\varphi}$. Time-dependency of $\hat{u}$ is also illustrated in Figure 4.2. The optimal strategy is increasing in W when W is relatively small, and is decreasing when W is large for $t = t_0$. On the other hand, the hedging strategy in general increases with respect to S at $t = t_9$. These show the competition between the two factors $\Phi(S)$ and W at different time steps. At early stages W plays a more important part: small W causes under-hedge and large W leads to over-hedge. When we approach terminal time, $\Phi(S)$ is the main driver of the hedging error: higher stock price implies higher projected option payoff, hence one would invest a larger proportion of the wealth into the stock to reduce the hedging error. In addition, change in the region of the contour reflects different experimental designs $\mathcal{D}_k$ for different time steps.

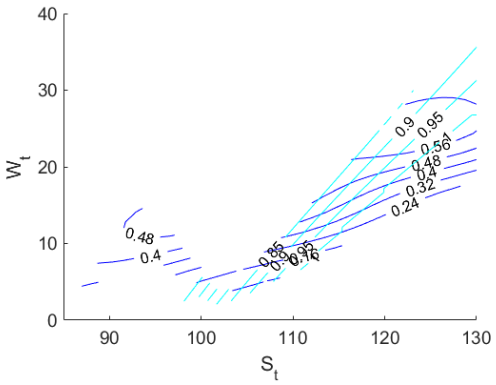


Figure 4.2: Contour plot of the adaptive robust hedging strategy $\hat{u}(t_k, S, W, \bar{\mu}, \bar{\sigma})$ based on the GP surrogate within the region $\{(S, W) : 0.5P^{BS}(t_k, S) \leq W \leq 1.5P^{BS}(t_k, S)\}$ with $\bar{\mu} = 0.12$, $\bar{\sigma} = 0.4$, $t = t_0$ (blue), $t = t_9$ (cyan).

4.2. Comparison to Other Approaches. As comparators to adaptive robust control, we consider again the myopic adaptive (MA) and static robust (SR) formulations, as well as a naive adaptive Delta (AD) hedging. The latter buys $\Delta(t_k, S_{t_k}; \bar{\theta}_{t_k})$ shares at each step t_k, plugging-in the latest parameter estimates into the classical Delta-hedging strategy of the Black-Scholes model. MA control corresponds to first solving the optimization problem treating θ as a parameter, which yields $\tilde{u}^{MA}(t_k; \theta)$. The adopted strategy is then $\tilde{u}^{MA}(t_k; \bar{\theta}_{t_k})$ which, like the adaptive Delta, leads to a purely learning-based approach. The SR formulation takes $\Theta_{t_k} \equiv \Theta_0$ and then solves the Bellman equations (4.5).

To make the comparison, we compute the distribution of the respective terminal hedging errors $H = \Phi(S_T) - W_T$ on $N' = 50000$ out-of-sample forward paths, using $\ell(h) = h^+ + 0.75h^-$. Figure 4.3 shows the empirical histogram of H from the forward simulations. Better hedging corresponds to lower average loss $\mathbb{E}[\ell(H)]$ and in particular should concentrate errors closer

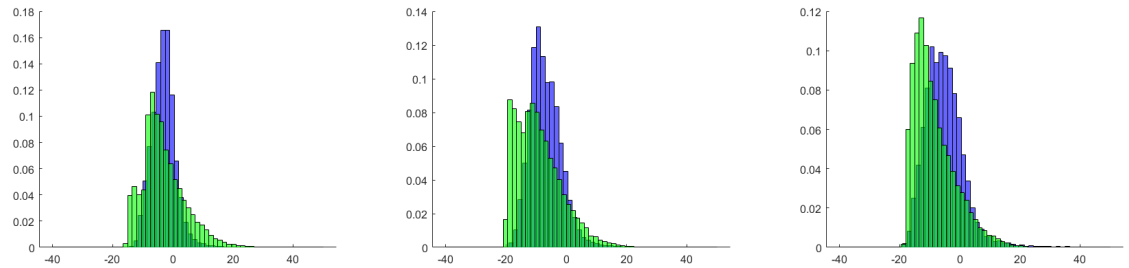


Figure 4.3: Histograms of hedging error $H = \Phi(S_T) - W_T$ for loss function $\ell(h) = h^+ + 0.75h^-$. Adaptive robust control (blue); adaptive Black-Scholes Δ (green). Left: $S_{t_0} = 90$, $W_0 = 10$; Center: $S_{t_0} = 100$, $W_{t_0} = 20$; Right: $S_{t_0} = 110$, $W_{t_0} = 25$.

to zero, so that mean$(H) \simeq 0$ and Var(H) small are preferred. We see in Figure 4.3 that the hedging error of Adaptive Robust strategy is more concentrated around $H = 0$. In the out-of-the-money and in-the-money cases, Adaptive Delta leads to significantly larger tail on the positive side. Hence, AR produces a comparatively better strategy which "super-hedges" (consistently more negative H) AD. This is not very surprising as the AD recipe is ad hoc and is not targeting the loss function.

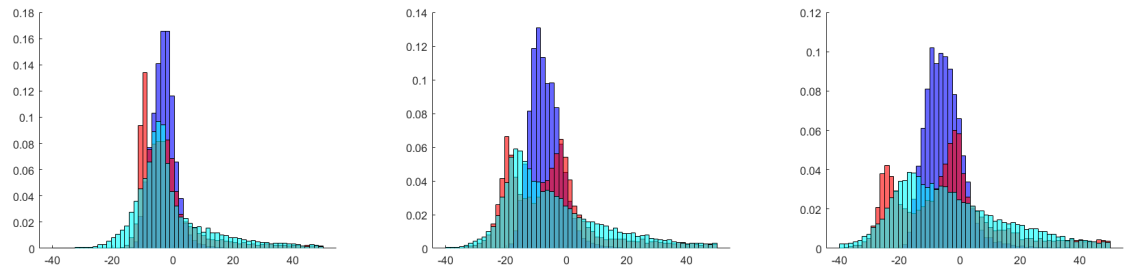


Figure 4.4: Distribution of $H = \Phi(S_T) - W_T$: adaptive robust (blue); static robust (red); myopic adaptive (cyan) for loss function $\ell(h) = h^+ + 0.75h^-$. Left: out-of-the-money $S_{t_0} = 90$, $W_{t_0} = 10$; Center: at-the-money $S_{t_0} = 100$, $W_{t_0} = 20$; Right: in-the-money $(S_{t_0}, W_{t_0}) = (110, 25)$.

Figure 4.4 shows that the AR approach frequently performs better than SR and MA strategies. When taking a two-sided loss function ($\lambda = 0.75$), the hedging error of AR concentrates more around zero and has apparent thinner tails on both sides compared to the other two. Due to lack of learning, SR strategies are more "conservative" and over-invest in the underlying asset (see Figure 4.5 below) which increases the variance of $\Phi(S_T) - W_T$. Myopic adaptive control is over-optimistic and tends to lead to very high variance, already noted in [9]. Table 4.2 shows the impact of λ on the performances of these three approaches. SR does better than the other two when $\lambda = 0$. Note that this corresponds to the super-hedging objetive, hence it is not surprising that the conservative approach wins as it over-invests in

	$\lambda = 0$			$\lambda = 0.5$			$\lambda = 0.75$		
	AR	SR	MA	AR	SR	MA	AR	SR	MA
mean(H)	-6.09	-8.82	-2.79	-7.11	-3.34	-0.60	-6.81	-1.01	-1.26
std(H)	16.06	5.07	21.62	5.24	19.62	25.05	5.26	24.09	22.95
$q_{0.95}(H)$	21.76	-1.04	37.57	1.72	33.19	49.81	1.86	52.07	43.06
V_0	3.52	0.08	6.50	4.07	8.44	13.83	5.70	13.59	14.31

Table 4.2: Mean, standard deviation, and 95%-quantile of the hedging error $H = \Phi(S_T) - W_T$, as well as the mean loss $V(t_0, S_{t_0}, W_{t_0})$ with $S_{t_0} = 100$, $W_{t_0} = 20$. We compare the Adaptive Robust (AR), Static Robust (SR) and Myopic Adaptive (MA) methods and three different loss functions $\ell(h)$ in (4.4).

the risky asset. For $\lambda > 0$ where positive hedging error is also penalised, adaptive robust formulation outperforms the other two. Choosing the optimal strategy in these cases are more delicate, and a method which combines learning and robustness handles the situation better.

4.3. Comparison of Optimal Hedges. Compared to hedging strategies generated by other approaches, the adaptive robust strategy follows the movement of the underlying S_t more closely. Such behavior can be understood as a better learning of the model than other methods. Figure 4.5 displays a sample trajectory of S_t and corresponding $\tilde{u}(t_k, S_{t_k})$ across the above three approaches. When stock price is volatile during the time period, robust strategies $\tilde{u}^{AR}(t_k, S_{t_k})$ and $\tilde{u}^{SR}(t_k, S_{t_k})$ are much more stable than the pure learning-based strategies $\tilde{u}^{MA}(t_k, S_{t_k})$ and $\tilde{u}^{AD}(t_k, S_{t_k}, \bar{\theta}_{t_k})$. It is also worth mentioning that for the static robust approach the assumed drift $\check{\mu}$ corresponding to the worst case model is very high. Namely the SR worst case is that the Call ends in-the-money and a large positive hedging error $H \gg 0$ results. To compensate against this scenario, the static robust strategy $\tilde{u}^{SR}$ over-invests in the risky asset to ensure that $W_T \simeq \Phi(S_T)$ conditional on $S_T \gg \mathcal{K}$, which conversely generates larger risk in other, less adversarial scenarios. This explains why $u^{SR}(t_k, S_{t_k}) > u^{AR}(t_k, S_{t_k})$ in Figure 4.5.

5. Discussion.

5.1. Enhancements. A central feature of our machine learning approach is the rich set of opportunities to enhance the computations. These concern the ultimate aim of (i) faster running time; (ii) more stable and accurate estimates of $\hat{V}$ and $\hat{u}$. Practically, we would like to be able to specify a small simulation budget N_k and then obtain good results quickly. We do note that there is a third dimension of actual running time—if the statistical surrogate model is too complicated then its overhead can be high even when N_k is low. For example, we can propose heuristics for sequential, adaptive designs $\mathcal{D}_k$ that would maximize the learning ability of $\hat{u}$ as a function of N_k, but actually slow down the algorithm (hence we deem them low-priority extensions).

Faster optimization: Generating a single training pair (x_{t_k}, v_{t_k}) in (2.9) requires solving

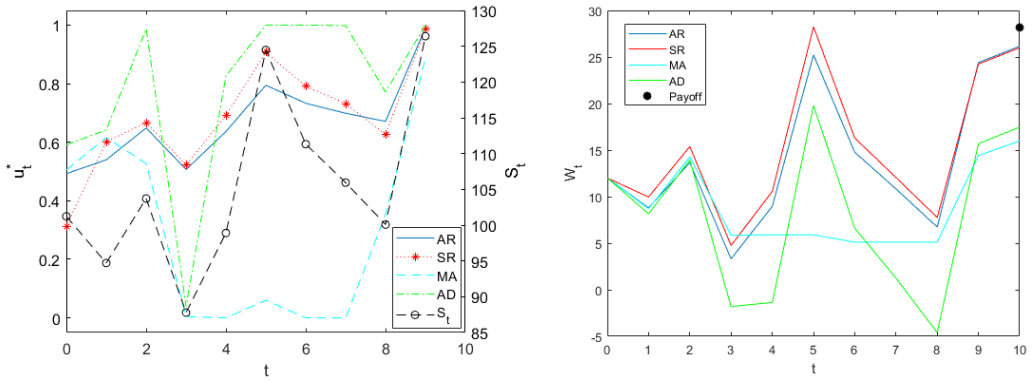


Figure 4.5: *Left panel:* path of adaptive robust hedge $t_k \mapsto \tilde{u}^{AR}(t_k, x_{t_k})$ (blue) for a European Call option in comparison to static robust $\tilde{u}^{SR}(t_k, x_{t_k})$ (red), myopic adaptive $\tilde{u}^{MA}(t_k, x_{t_k})$ (cyan), and adaptive Delta $\tilde{u}^{AD}(t_k, S_{t_k}; \bar{\theta}_{t_k})$ (green). The respective path of S_{t_k} is shown in black (recall strike of $\mathcal{K} = 100$). *Right:* paths of the respective wealth processes W_{t_k} color-coded accordingly. The difference between plotted terminal wealth W_T and $\Phi(S_T) \simeq 29$ (black dot, option expires in-the-money) is the respective hedging error H.

a nested optimization problem (maximizing over u, minimizing over (μ, σ)), which is the most expensive part of the algorithm. Since this is done inside all the loops (over k and over n), making this step efficient is central to fast performance. In our existing implementation we utilize off-the-shelf gradient-free optimizers (`fminbnd` and `fminsearch` in `Matlab`) which are agnostic to the structure of $\hat{E}$ and $\hat{V}(t_{k+1}, \cdot)$ and operate completely generically. One way to speed up is through warm-starting the optimizers at the location $\check{u}(t_{k+1}, x^n_{t_k})$, i.e. assuming that the new optimizer is the same as the one from the previous (later) time-step. For $\mathscr{F}_2$ in (2.8)-(4.6) we may warm-start the optimizer at $\varphi = \pi$. A further enhancement is to rely on gradient-descent optimizers which would require to estimate $\nabla \mathscr{F}_2(u)$. Within the GP setup we may analytically differentiate the GP kernel to learn $\nabla_x \hat{V}$ which in turn could be converted into $\nabla_u \mathscr{F}_2$.

GP Hyperparameters: A significant portion of the algorithm running time is spent on fitting the GP surrogates, i.e. inferring (typically by maximum likelihood maximization) the respective hyperparameters ϑ_{t_k}. This is a hard nonlinear optimization problem with total time complexity $\mathcal{O}(K \cdot N^3)$, with the dependence on K due to the need to fit a surrogate at each time-step t_k. Generally, the hyperparameters $\hat{\vartheta}_{t_k}$ are stable over time, so one could use this for another warm-start in the MLE optimizer, or even directly set $\hat{\vartheta}_{t_k} = \hat{\vartheta}_{t_{k+1}}$, freezing hyperparameters across (some) time-steps. Because the GP surrogate is data-driven, the final prediction depends mostly on the training data and less on the precise value of ϑ_{t_k}.

In terms of accuracy, an important piece of building the surrogate are its extrapolation properties. Recall that for extrapolation the GP prediction reverts to its prior $m_*(x) \to m_0(x)$ outside of the input domain. For fitting the control surrogate $\hat{u}$ in both case studies we take $m_0(x) \equiv 0$, but other choices would be appropriate depending on the problem/surrogate and in our experience can play a nontrivial role in ultimate solution quality. (However, note that

$m_0(\cdot)$ is only relevant for extrapolation—within $\mathcal{D}$ the prediction is driven by the training data.)

5.2. True Monte Carlo. The use of numerical quadrature to integrate out $\hat{V}(t_{k+1},\cdot)$ is probabilistically biased in the sense that $\hat{E}^{Quad}[\hat{V}(t_{k+1},\cdot)](x) \neq \mathbb{E}[\hat{V}(t_{k+1},\cdot)](x)$ so that a consistent, deterministic error is introduced during this sub-step. A well-known alternative is to employ Monte Carlo integration, namely replacing the integral with a *random* sum

$$\int \hat{V}(t_{k+1},\mathbf{T}(t_k,x,u,\theta,z))f_\epsilon(z)dz \approx \frac{1}{\mathfrak{I}}\sum_{i=1}^{\mathfrak{I}}\hat{V}(t_{k+1},\mathbf{T}(t_k,x,u,\theta,Z_i)) =: \hat{V}^{MC}(t_k), \tag{5.1}$$

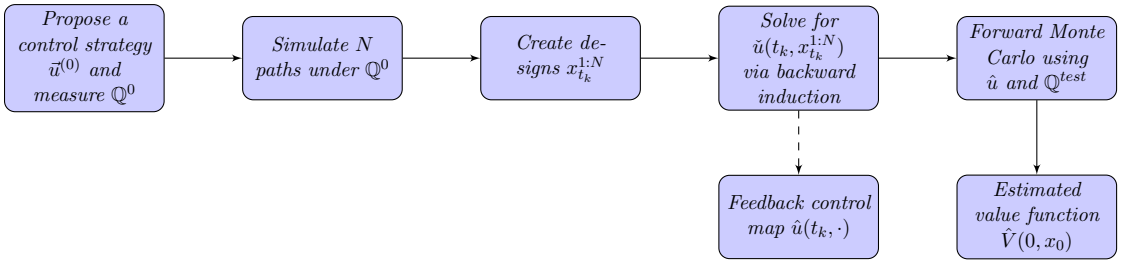
where $Z_i \sim \mathcal{N}(0,1), i = 1,\ldots,\mathfrak{I}$ are now i.i.d. Gaussian samples, furthermore independent across different x^n's. Relative to quadrature, the MC approximation is unbiased, but noisy. We find that the latter noise makes a Monte Carlo-based scheme much less accurate.

Specifically, for the optimal hedging problem, we computed $\tilde{u}^{\mathrm{MC}}(t_k,\cdot)$ and $\tilde{u}^{\mathrm{Quad}}(t_k,\cdot)$ using several different quantization/inner-simulation levels $\mathfrak{I}$ and relying on identical design $(S^{1:N},\bar{\mu}^{1:N},\bar{\sigma}^{1:N},W^{1:N})$. We find that to achieve comparable accuracy one needs 3-5 times more MC points compared to quadrature points. Given that the computational complexity of both approaches is the same, it seems much preferable to employ quantization.

5.3. Related Control Formulations. As already demonstrated in the one-period example of Section 3.1, our numerical tools can be straightforwardly applied to *parametric* control problems. In the latter formulation, rather than seeking to optimize/robustify against a collection of potential θ's, we simply wish to understand the dependence of the value function or feedback control on θ. The proposed Gaussian process surrogates is a natural tool for that purpose, allowing the modeler to solve the underlying control problem at several instances $\bar{\theta}^n$, $n = 1,\ldots,N$ and then interpolate to obtain $V(\cdot;\theta)$ and $u^*(\cdot;\theta)$ at arbitrary $\theta \in \Theta$. In particular, such parametric control is relevant for the myopic adaptive problem where one pre-computes $V(\cdot;\theta)$ and then plugs-in the latest belief $\bar{\theta}_t$.

Secondly, our scheme also immediately nests the dynamic adaptive problem where the adversarial set $\Theta(\bar{\theta}) = \{\bar{\theta}\}$ is a singleton, so the inner optimization disappears. Indeed, one can simply take $\kappa = 0$ to collapse the radius of Θ to zero, and then run exactly the same code as for the adaptive robust setting. Third, the scheme can be adapted to handle Bayesian formulations that replace $\inf_\theta$ with an integral $\int_\theta(\cdot)\nu_{t_k}(d\theta)$ against a distribution $\nu_{t_k}(d\theta)$. Learning becomes encoded as updating the posterior $\nu_{t_k}(\cdot)$ of θ over time. In the practically tractable case, there are finite-dimensional sufficient statistics (such as the posterior mean) $\bar{\theta}_{t_k}$ for $\mathcal{L}(\theta^*|\mathcal{F}_{t_k})$ so the integral over $\nu_{t_k}(d\theta)$ can be reduced to a numerical quadrature over the parametrized posterior distribution. For example, under drift uncertainty and assuming the setting of trying to learn an unknown fixed $\tilde{\mu}$, $\mathcal{L}(\tilde{\mu}|\mathcal{F}_T)$ is Gaussian with a certain formula for mean $\bar{\mu}_{t_k}$ and posterior variance V_t and integrating against $\nu_{t_k}(d\theta)$ becomes a one-dimensional Gaussian integral.

Remark 5.1. *Our algorithm can also be interpreted as a type of Control Randomization. In CR, the solution pipeline is of the form depicted below, where $\vec{u}^{(0)}$ is possibly randomized.*



With this interpretation, any experimental design can be viewed as an implicit recipe for $\mathbb{Q}^0$*, namely specifying the respective marginal distribution of* X_{t_k}*. This perspective emphasizes the fact that the quality of the design is linked to the quality of the ansatz for* $u^{(0)}$*. It also implies that the training procedure can be embedded into top-level iterations, whereby better and better estimates of* u^* *are fed back as new guesses* $u^{(r)}, r = 1, \ldots$ *to generate more and more targeted simulation designs* $x_{t_k}^{1:N,(r)}$*.*

5.4. Conclusion. We have developed a machine learning-based algorithm for adaptive robust control. Our motivation comes from the multiple desirable features of adaptive control and aims to mitigate the associated computational challenges, making the adaptive robust framework numerically feasible. The resulting case studies provide new insights on the interplay between learning, controlling for model uncertainty and risk aversion in the context of the classical Merton problem and the loss-based hedging problem.

The key innovation in our methodology is to build multiple surrogates for different pieces of the Bellman recursion, in particular not just for the value function but also for the feedback control map and worst-case parameters. To do so we propose utilization of Gaussian Process surrogates which check off multiple important computational considerations. The developed algorithm is certainly of independent interest. Further investigations of non-financial contexts and of other related control formulations are deferred to future research.

Acknowledgement. We thank the anonymous reviewers for helpful comments to improve our presentation. We are also grateful to the participants of the Vietnam Institute for Advanced Study in Mathematics 2019 Tuan Chau Workshop "Recent developments in mean-field game, machine learning and quantitative finance" for feedback on an earlier draft. Ludkovski is partially supported by NSF DMS-1821240.

REFERENCES

[1] R. Aid, L. Campi, N. Langrené, and H. Pham, *A probabilistic numerical method for optimal multiple switching problem and application to investments in electricity generation*, SIAM J. Financial Math., 5 (2014), pp. 191–231.

[2] A. Bachouch, C. Huré, N. Langrené, and H. Pham, *Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications*, Methodol. Comput. Appl. Probab., (2021).

[3] A. Balata, M. Ludkovski, A. Maheshwari, and J. Palczewski, *Statistical learning for probability-constrained stochastic optimal control*, European J. Oper. Res., 290 (2021), pp. 640–656.

[4] V. Bally, G. Pagès, and J. Printems, *A quantization tree method for pricing and hedging multidimensional American options*, Math. Finance, 15 (2005), pp. 119–168.

[5] C. Bao, Z. Zhu, N. Langrené, and G. Lee, *Multi-period dynamic portfolio optimization through least squares learning*, in IAENG Transactions On Engineering Sciences: Special Issue for the International Association of Engineers Conferences 2014, 2015, pp. 29–42.

[6] D. Belomestny, A. Kolodko, and J. Schoenmakers, *Regression methods for stochastic control problems and their convergence analysis*, SIAM J. Control Optim., 48 (2010), pp. 3562–3588.

[7] D. P. Bertsekas, *Approximate policy iteration: A survey and some new methods*, J. Control Theory App., 9 (2011), pp. 310–335.

[8] T. Bielecki, T. Chen, and I. Cialenco, *Recursive construction of confidence regions*, Electron. J. Stat., 11 (2017), pp. 4674–4700.

[9] T. R. Bielecki, T. Chen, I. Cialenco, A. Cousin, and M. Jeanblanc, *Adaptive robust control under model uncertainty*, SIAM J. Control Optim., 57 (2019), pp. 925–946.

[10] M. W. Brandt, A. Goyal, P. Santa-Clara, and J. R. Stroud, *A simulation approach to dynamic portfolio choice with an application to learning about return predictability*, Rev. Financ. Stud., 18 (2005), pp. 831–873.

[11] V. Chen, D. Ruppert, and C. Shoemaker, *Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming*, Oper. Res., 47 (1999), pp. 38–53.

[12] F. Cong and C. Oosterlee, *Multi-period mean-variance portfolio optimization based on Monte Carlo simulation*, J. Econom. Dynam. Control, 64 (2016), pp. 23 – 38.

[13] I. Guo, N. Langrené, G. Loeper, and W. Ning, *Robust utility maximization under model uncertainty via a penalization approach*, arXiv preprint arXiv:1907.13345, (2019).

[14] J. Han and W. E, *Deep Learning Approximation for Stochastic Control Problems*, tech. rep., arXiv 1611.07422, 2016.

[15] L. P. Hansen, T. J. Sargent, G. Turmuhambetova, and N. Williams, *Robust control and model misspecification*, J. Econom. Theory, 128 (2006), pp. 45–90.

[16] C. Huré, H. Pham, A. Bachouch, and N. Langrené, *Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis*, SIAM J. Numer. Anal., 59 (2021), pp. 525–557.

[17] I. Kharroubi, N. Langrené, and H. Pham, *A numerical algorithm for fully nonlinear HJB equations: an approach by control randomization*, Monte Carlo Methods Appl., 20 (2014), pp. 145–165.

[18] M. Ludkovski and A. Maheshwari, *Simulation methods for stochastic storage problems: A statistical learning perspective*, Energy Syst., (2019), pp. 1–39.

[19] M. Monoyios, *Optimal hedging and parameter uncertainty*, IMA J. Manag. Math., 18 (2007), pp. 331–351.

[20] G. Pagès, H. Pham, and J. Printems, *An optimal Markovian quantization algorithm for multi-dimensional stochastic control problems*, Stoch. Dyn., 4 (2004), pp. 501–545.

[21] W. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, Wiley-Blackwell, 2007.

[22] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[23] J. Risk and M. Ludkovski, *Statistical emulators for pricing and hedging longevity risk products*, Insurance Math. Econom., 68 (2016), pp. 45–60.

[24] W. Runggaldier, B. Trivellato, and T. Vargiolu, *A Bayesian adaptive control approach to risk management in a binomial model*, Dalang R.C., Dozzi M., Russo F. (eds) Seminar on Stochastic Analysis, Random Fields and Applications, III Progress in Probability, 52 (2002).

[25] T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*, Springer Science & Business Media, 2013.

[26] R. Zhang, N. Langrené, Y. Tian, Z. Zhu, F. Klebaner, and K. Hamza, *Efficient simulation method for dynamic portfolio selection with transaction cost, liquidity cost and market impact*, arXiv preprint arXiv:1610.07694, (2016).