OPTIMAL CONTROL BY POLICY IMPROVEMENTS AND CONSTRAINED GAUSSIAN PROCESS REGRESSIONS

Donatien Hainaut, Jean-Loup Dupret

LIDAM Discussion Paper ISBA
2025 / 12

UCLouvain

ISBA | LIDAM
Louvain Institute of Data Analysis and Modeling in economics and statistics

ISBA

Voie du Roman Pays 20 - L1.04.01

B-1348 Louvain-la-Neuve

Email : lidam-library@uclouvain.be

https://uclouvain.be/en/research-institutes/lidam/isba/publication

Optimal control by policy improvements and constrained Gaussian process regressions

Donatien Hainaut*
UCLouvain - LIDAM
Jean-Loup Dupret†
ETH Zürich - RiskLab

This article proposes a new iterative algorithm combining policy improvements and Gaussian process regressions for solving stochastic control problems. At each iteration, we update an approximated value function and then improve the controls. We sample state variables in the inner domain and on the terminal boundary of the Hamilton-Jacobi-Bellman (HJB) equation. The approximated value function, the solution of this equation, is obtained by fitting a constrained regression function. The regression function matches the terminal utility on the boundary sample and satisfies the HJB equation on the inner sample. Assuming the regression function is a Gaussian process, we find a closed-form approximation of the value function and of the controls. In a numerical illustration, we test the efficiency of the method for solving the consumption-investment and linear-quadratic regulator problems.

KEYWORDS: Optimal control, Gaussian process regression, Hamilton-Jacobi-Bellman equation, reinforcement learning, policy improvement.

1 Introduction

This work introduces a novel iterative algorithm based on constrained Gaussian process regressions (CGPR) for solving optimal control problems in continuous time. We show that this method is an accurate and numerically efficient alternative to existing approaches. Its methodological foundations are Gaussian process regressions and policy improvements.

Gaussian processes are used for regressing a response y on a vector of covariates $\boldsymbol{x}$. We refer the reader to Rasmussen and Williams (2006, chapter 6) or to Murphy (2012, chapter 15) for a detailed presentation. The Gaussian process regression is a Bayesian method using as a-priori distribution for responses y, a Gaussian process (GP), with zero mean and covariance $k(\boldsymbol{x}, \boldsymbol{x\prime})$, called kernel. This choice is made prior to taking into account observations. The regression function is the a-posteriori expectation of this GP, conditioned by observations. Seminal works discussing the use of GPs as surrogate models for computational science and engineering applications include the papers of Sacks et al. (1989) and Santner et al. (2003), and the book by Rasmussen and Williams (2006). Without being exhaustive, GPs are used in various fields such

*Corresponding author. Postal address: Voie du Roman Pays 20, 1348 Louvain-la-Neuve Belgium. E-mail to: donatien.hainaut(at)uclouvain.be
†Postal address: Rämistrasse 101HG F 428092 Zurich Switzerland E-mail to: jeanloup.dupret@math.ethz.ch

as time-series, Roberts et al. (2013), data mining of large datasets, Banerjee (2013), materials sciences, Deringer et al. (2021), operational research, Price et al. (2019), quantitative finance, De Spiegeleer et al. (2018) or missing data, Choi et al. (2024).

In a similar manner to Physics-Inspired Neural Networks (PINNs), we can constrain a Gaussian process regression with a partial differential equation (PDE). This PDE represents the physical law governing the experiment from which responses are measured. The literature on constrained Gaussian process regression (CGPR) is vast, and we refer the reader to Swiler et al. (2020) for a detailed survey. In this article, the PDE has the form $\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}}g = z(\boldsymbol{x})$ where g, $\mathcal{L}_{x,\boldsymbol{c}}$ and $\boldsymbol{x}$ are respectively, the value, a linear differential operator controlled by $\boldsymbol{c}$, and a vector of risk processes. The boundary constraint is $g(\boldsymbol{x}) = h(\boldsymbol{x})$, where $h(\boldsymbol{x})$ is the terminal utility at expiry. To solve this, we adopt the methodology developed in Hainaut and Vrins (2024) and Hainaut (2024) for pricing financial derivatives. We fit a constrained regression on two samples of points, $\mathring{\boldsymbol{X}}$ and $\bar{\boldsymbol{X}}$, respectively in the inner and boundary domains of the PDE. In this setting, the solution is a regression function $g(\boldsymbol{x})$ with $g(\boldsymbol{x}) = h(\boldsymbol{x})$ for all $\boldsymbol{x} \in \bar{\boldsymbol{X}}$, which satisfies a PDE constraint, $\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}}g = z(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathring{\boldsymbol{X}}$. We can draw a parallel with methods based on PINNs, in which the CGPR is replaced by a neural network. We refer the reader to Raissi et al. (2019) for a general presentation of PINNs. For other examples of PINNs applied to financial valuation or stochastic control, we refer to Sirignano and Spiliopoulos (2018), Al-Aradi et al. (2022), Glau and Wunderlich (2022, 2024), Hainaut (2024), and Hainaut and Casas (2024). The main drawback of PINNs compared to CGPR is that they rely on numerical differentiation for solving the PDE. In comparison, the CGPR approximation with a Gaussian kernel is fully analytical.

On the other hand, the value function, $V(\boldsymbol{x})$, of a stochastic optimal control problem in continuous time, is the solution of the Hamilton-Jacobi-Bellman (HJB) equation. In many circumstances, the optimal control admits an explicit representation in terms of partial derivatives of $V(\boldsymbol{x})$, and the HJB equation, after inserting the optimal controls, becomes a non-linear PDE. For more details, we e.g. refer to Touzi (2012). The CGPR, being designed for linear PDEs, cannot directly be applied, A solution coming from the reinforcement learning literature consists of using a policy improvement (PI) algorithm, which iteratively approximates the value function and then updates the controls. We refer the reader to Howard (1960), Werbos (1992) or Sutton and Barto (2018) for a detailed presentation, and to Jacka and Mijatovic (2017) for an analysis of the convergence. The PI algorithm is used for various purposes from process quality improvement, Fine et Porteus (1989) or Bachouch et al. (2022) to deep learning for stochastic control, Dupret et Hainaut (2024). For a survey, we refer to Bertsekas (2011). In this article, we combine the PI algorithm to CGPR. At each iteration, we solve the HJB equation for a given control with a CGPR. Next, we modify the controls in order to improve the value function during the next iteration. The numerical illustrations confirm the efficiency of this approach which offers a new alternative to existing methods for solving optimal control problems.

The article is organized as follows. We first present the optimal control problem and review the algorithm of policy improvements (PI). We prove that under mild assumptions, that this procedure increases the value function at each iteration. Section 3 starts with a brief review of general principles of the CGPR. Next we present the full algorithm for solving the HJB equation by policy improvements and constrained Gaussian process regressions (PI-CGPR algorithm). In section 4, we develop the algorithm in the case of Gaussian kernels. These kernels allow us to find analytical expressions of the approximate value function and controls. In Section 5, we provide a thorough numerical analysis of the PI-CGPR algorithm, on the investment-consumption and on the linear-regulator problems.

2 Optimal control by policy improvements

We consider a system ruled by a $p-1$ vector of risk processes[1],

$$(\boldsymbol{Y}_t)_{t\geq 0} = \left(\left(Y_t^{(1)}, ..., Y_t^{(p-1)}\right)\right)_{t\geq 0} .$$

They are defined on a complete probability space, Ω, endowed with a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ and a probability measure, $\mathbb{P}$. We assume that risk processes are ruled by a system of stochastic differential equations (SDEs):

$$d\boldsymbol{Y}_t \;=\; \boldsymbol{\mu}_{\boldsymbol{Y}}(t, \boldsymbol{Y}_t, \boldsymbol{c}_t)dt + \Sigma_{\boldsymbol{Y}}(t, \boldsymbol{Y}_t, \boldsymbol{c}_t)d\boldsymbol{B}_t\,, \tag{1}$$

where $\boldsymbol{B} = (\boldsymbol{B}_t)_{t\geq 0}$ is a v-vector of independent Brownian motions, $\boldsymbol{\mu}_{\boldsymbol{Y}}(.)$ is a vector of dimension $(p-1)$ and $\Sigma_{\boldsymbol{Y}}(.)$ is a $(p-1)\times v$ matrix. The $\mathbb{R}^l$-valued control $(\boldsymbol{c}_t)_{t\geq 0}$ is $\mathbb{F}$-adapted and takes values in a subset A of $\mathbb{R}^l$.

The set of admissible strategies is denoted by $\mathcal{A}$. In later developments, we consider $\mathbb{F}$-adapted Markov controls that can be written as $\boldsymbol{c}_t = \boldsymbol{c}(t, \boldsymbol{Y}_t)$, $\forall t \geq 0$ for some measurable map $\boldsymbol{c} : \mathbb{R}^+ \times \mathbb{R}^{p-1} \to \mathbb{R}^l$, with upper and lower bounds:

$$\mathcal{A} = \{(c_t)_{t\geq 0} \mid \mathbb{F}-\text{adapted and } \boldsymbol{c}_{min}(t, \boldsymbol{Y}_t) \leq \boldsymbol{c}_t \leq \boldsymbol{c}_{max}(t, \boldsymbol{Y}_t)\}\,,$$

where $\boldsymbol{c}_{min}(t, \boldsymbol{Y}_t)$ and $\boldsymbol{c}_{max}(t, \boldsymbol{Y}_t)$ are functions from $\mathbb{R}^+ \times \mathbb{R}^{p-1} \to A \subset \mathbb{R}^l$. The functions $\boldsymbol{\mu}_{\boldsymbol{Y}} : \mathbb{R}^+ \times \mathbb{R}^{p-1} \times A \to \mathbb{R}^{p-1}$ and $\Sigma_{\boldsymbol{Y}} : \mathbb{R}^+ \times \mathbb{R}^{p-1} \times A \to \mathbb{R}^{(p-1)\times v}$ satisfy a condition of linear growth : $\exists C \geq 0$, $\forall \boldsymbol{y} = (y_1, ..., y_{p-1})^\top \in \mathbb{R}^{p-1}$ $\forall \boldsymbol{c} \in A$

$$\begin{aligned} \|\boldsymbol{\mu}_{\boldsymbol{Y}}(t, \boldsymbol{y}, \boldsymbol{c})\|_2 &\leq C\left(1 + \|\boldsymbol{y}\|_2 + \|\boldsymbol{c}\|_2\right)\,, \\ \|\Sigma_{\boldsymbol{Y}}(t, \boldsymbol{y}, \boldsymbol{c})\|_2 &\leq C\left(1 + \|\boldsymbol{y}\|_2 + \|\boldsymbol{c}\|_2\right)\,, \end{aligned} \tag{2}$$

and are Lipschitz: $\exists C \geq 0$, $\forall \boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^{p-1}$ $\forall \boldsymbol{c} \in A$

$$\begin{aligned} \|\boldsymbol{\mu}_{\boldsymbol{Y}}(s, \boldsymbol{y}_2, \boldsymbol{c}) - \boldsymbol{\mu}_{\boldsymbol{Y}}(t, \boldsymbol{y}_1, \boldsymbol{c})\|_2 &\leq C\left(\|\boldsymbol{y}_2 - \boldsymbol{y}_1\|_2 + \|s - t\|_2\right) \\ \|\Sigma_{\boldsymbol{Y}}(s, \boldsymbol{y}_2, \boldsymbol{c}) - \Sigma_{\boldsymbol{Y}}(t, \boldsymbol{y}_1, \boldsymbol{c})\|_2 &\leq C\left(\|\boldsymbol{y}_2 - \boldsymbol{y}_1\|_2 + \|s - t\|_2\right) \end{aligned} \tag{3}$$

Furthermore, $\boldsymbol{\mu}_{\boldsymbol{Y}}(., ., \boldsymbol{c})$ and $\Sigma_{\boldsymbol{Y}}(., ., \boldsymbol{c})$ are of class C^1 in t and $\boldsymbol{y}$. Those conditions guarantee the existence of a unique solution to (1). Next, we define the criterion to be maximized, $J(t, \boldsymbol{y}, \boldsymbol{c})$, that we name payoff. Let U_1, U_2 be continuous positive functions which satisfy the following polynomial growth conditions for some $k \in [0, \infty)$:

$$\begin{aligned} |U_1(t, \boldsymbol{y}, \boldsymbol{c})| &\leq C\left(1 + \|\boldsymbol{y}\|_2^k + \|\boldsymbol{c}\|_2^k\right) \\ |U_2(t, \boldsymbol{y})| &\leq C\left(1 + \|\boldsymbol{y}\|_2^k\right) \end{aligned}$$

Given the pair $\boldsymbol{x} = (t, \boldsymbol{y}) \in \mathbb{R}^+ \times \mathbb{R}^{p-1}$ and an admissible control[2] $\boldsymbol{c}$, we define the payoff as follows

$$J(\boldsymbol{x}, \boldsymbol{c}) = \mathbb{E}\left(\int_t^T e^{-\alpha(s-t)} U_1(s, \boldsymbol{Y}_s, \boldsymbol{c}_s)ds + e^{-\alpha(T-t)} U_2(T, \boldsymbol{Y}_T) \mid \boldsymbol{Y}_t = \boldsymbol{y}\right)\,, \tag{4}$$

where T is the time horizon of the problem and α is a discount rate. Let $\mathring{\mathcal{X}} = [0, T) \times \mathbb{R}^{(p-1)}$, $\bar{\mathcal{X}} = T \times \mathbb{R}^{(p-1)}$ and $\mathcal{X} = \mathring{\mathcal{X}} \cup \bar{\mathcal{X}}$. We aim to find the control $(\boldsymbol{c}_t)_{t\geq 0}$ maximizing $J(\boldsymbol{x}, \boldsymbol{c})$. The value function $V(\boldsymbol{x})$ for $\boldsymbol{x} = (t, \boldsymbol{y})$ is the maximum of Eq. (4) among all admissible strategies:

$$V(\boldsymbol{x}) = \max_{c\in\mathcal{A}} J(\boldsymbol{x}, \boldsymbol{c})\,. \tag{5}$$

[1] We use bold notations for vectors.

[2] Notice that $\boldsymbol{c}$ either refers to the map in $\mathcal{A}$ or to the control value in A depending upon the context.

If we adopt the notation $\boldsymbol{X}_t = (t, \boldsymbol{Y}_t)$ and use nested expectations, we can prove that for any stopping time θ such that $t \leq \theta \leq T$,

$$V(\boldsymbol{x}) = \max_{\boldsymbol{c}\in\mathcal{A}} \mathbb{E}\left(\int_t^\theta e^{-\alpha(s-t)} U_1(\boldsymbol{X}_s, \boldsymbol{c}_s)\, ds + e^{-\alpha(\theta-t)} V(\boldsymbol{X}_\theta) \,|\, \boldsymbol{Y}_t = \boldsymbol{y} \right). \tag{6}$$

Assuming that $V \in C^{1,2}(\mathring{\mathcal{X}})$, the gradient and the Hessian of $V(.)$ with respect to $\boldsymbol{y}$ are denoted by $\nabla_{\boldsymbol{y}} V$ and $\mathcal{H}_{\boldsymbol{y}}(V)$. Let us define the linear differential operator:

$$\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}} \cdot = \partial_t \cdot - \alpha \cdot + \boldsymbol{\mu}_{\boldsymbol{Y}}(t, \boldsymbol{y}, \boldsymbol{c})^\top \nabla_{\boldsymbol{y}} \cdot + \frac{1}{2} \mathrm{tr}\left(\Sigma_{\boldsymbol{Y}}(t, \boldsymbol{y}, \boldsymbol{c}) \Sigma_{\boldsymbol{Y}}(t, \boldsymbol{y}, \boldsymbol{c})^\top \mathcal{H}_{\boldsymbol{y}} \cdot \right). \tag{7}$$

Using standard arguments, the solution to the problem (6) satisfies the following Hamilton-Jacobi-Bellman (HJB) equation:

$$\sup_{\boldsymbol{c}\in A} \left(U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}) + \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}} V(\mathring{\boldsymbol{x}}) \right) = 0 \quad \mathring{\boldsymbol{x}} \in \mathring{\mathcal{X}}\,, \tag{8}$$

$$V(\bar{\boldsymbol{x}}) = U_2(\bar{\boldsymbol{x}}) \quad \bar{\boldsymbol{x}} \in \bar{\mathcal{X}}\,. \tag{9}$$

We solve this problem with the policy improvement (PI) algorithm. The PI algorithm yields an intuitive approach to optimal control by generating a sequence of controls $\boldsymbol{c}^{(n)} = \left(\boldsymbol{c}_t^{(n)}\right)_{t\geq 0}$, which improve the payoffs $J(\boldsymbol{x}, \boldsymbol{c}^{(n)})$, at each iteration. Let us assume that after $(n-1)^{th}$ steps, we have obtained a policy $\boldsymbol{c}^{(n-1)}$ and let us denote by

$$V^{(n-1)}(\boldsymbol{x}) = J(\boldsymbol{x}, \boldsymbol{c}^{(n-1)}) \quad , \quad \forall x \in \mathcal{X}$$

the corresponding payoff. We first update the control as follows

$$\boldsymbol{c}^{(n)} \;=\; \arg\sup_{\boldsymbol{c}\in A} \left(U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}) + \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}} V^{(n-1)}(\mathring{\boldsymbol{x}}) \right), \forall \mathring{\boldsymbol{x}} \in \mathring{\mathcal{X}}. \tag{10}$$

$\boldsymbol{c}^{(n)}$ is called an improvement of $\boldsymbol{c}^{(n-1)}$. Once the $n^{th}-$optimal control is determined, we calculate $V^{(n)}(\boldsymbol{x})$ solving

$$\begin{cases} \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}^{(n)}} V^{(n)}(\mathring{\boldsymbol{x}}) = -U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}) & \forall \mathring{\boldsymbol{x}} \in \mathring{\mathcal{X}}\,, \\ V^{(n)}(\bar{\boldsymbol{x}}) = U_2(\bar{\boldsymbol{x}}) & \forall \bar{\boldsymbol{x}} \in \bar{\mathcal{X}}\,. \end{cases} \tag{11}$$

We explain in the next section how to find an approximate solution with a Gaussian process regression. In later developments of this section, we assume that the exact solution exists and show that each iteration improves the value function.

It is also interesting to mention that from Eq. (11), $\mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}^{(n)}} V^{(n)}(\boldsymbol{x}) \leq 0$ as $U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}) \geq 0$. The next proposition is an intermediate result that is used later to demonstrate that $V^{(n)}(\boldsymbol{x})$ is improved at each iteration.

Proposition 1. *$\forall \boldsymbol{x} \in \mathcal{X}$, the following inequality holds:*

$$\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}^{(n)}} \left(V^{(n)}(\boldsymbol{x}) \right) \;\leq\; \mathcal{L}_{\boldsymbol{x},\boldsymbol{c}^{(n)}} \left(V^{(n-1)}(\boldsymbol{x}) \right). \tag{12}$$

Proof. From Eq. (11), $\forall \mathring{\boldsymbol{x}} \in \mathring{\mathcal{X}}$, we know that

$$U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(k)}) + \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}^{(k)}} V^{(k)}(\mathring{\boldsymbol{x}}) = 0 \qquad \text{for} \quad k = n-1\,,\, n\,.$$

By subtraction, we infer the equality

$$U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}) - U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n-1)}) + \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}^{(n)}} V^{(n)}(\mathring{\boldsymbol{x}}) - \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}^{(n-1)}} V^{(n-1)}(\mathring{\boldsymbol{x}}) = 0\,. \tag{13}$$

But from Eq. (10), $\boldsymbol{c}^{(n)}$ is optimal compared to $\boldsymbol{c}^{(n-1)}$:

$$U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n-1)}) + \mathcal{L}_{\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n-1)}} V^{(n-1)}(\mathring{\boldsymbol{x}}) \leq U_1(\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}) + \mathcal{L}_{\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}} V^{(n-1)}(\mathring{\boldsymbol{x}})\,. \tag{14}$$

Summing up Eq. (13) and (14), we obtain the inequality:

$$\mathcal{L}_{\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}} V^{(n)}(\mathring{\boldsymbol{x}}) - \mathcal{L}_{\mathring{\boldsymbol{x}}, \boldsymbol{c}^{(n)}} V^{(n-1)}(\mathring{\boldsymbol{x}}) \leq 0\,.$$

For $\bar{\boldsymbol{x}} \in \bar{\mathcal{X}}$, we have $V^{(n)}(\bar{\boldsymbol{x}}) = V^{(n-1)}(\bar{\boldsymbol{x}}) = U_2(\bar{\boldsymbol{x}})$ and Eq. (12) holds. □

To avoid any confusion in later developments, we temporarily denote by $\left(\boldsymbol{Y}_t^{(n)}\right)_{t\geq 0}$, the process controlled by $\left(\boldsymbol{c}_t^{(n)}\right)_{t\geq 0}$:

$$d\boldsymbol{Y}_t^{(n)} \;=\; \boldsymbol{\mu}_{\boldsymbol{Y}}(t, \boldsymbol{Y}_t^{(n)}, \boldsymbol{c}_t^{(n)})dt + \Sigma_{\boldsymbol{Y}}(t, \boldsymbol{Y}_t^{(n)}, \boldsymbol{c}_t^{(n)})d\boldsymbol{B}_t\,. \tag{15}$$

and $\boldsymbol{X}_t^{(n)} = (t, \boldsymbol{Y}_t^{(n)})$. The next proposition states that each iteration of the PI algorithm increases the value function, under mild assumptions.

Proposition 2. *Under the assumption that the difference of two consecutive value functions converge in L^1 to a non-negative random variable:*

$$\lim_{t\to T}\left(V^{(n)}(\boldsymbol{X}_t^{(n)}) - V^{(n-1)}(\boldsymbol{X}_t^{(n)})\right) \overset{L_1}{=} Z \geq 0 \quad , \tag{16}$$

and that $\mathcal{L}_{\boldsymbol{x}, \boldsymbol{c}^{(n)}} V^{(n)}(\boldsymbol{x})$ is well defined, the value function increases at each iteration:

$$V^{(n)}(\boldsymbol{x}) \;\geq\; V^{(n-1)}(\boldsymbol{x}) \quad , \quad \forall x \in \mathcal{X}.$$

Proof. For $s \geq t$ and $k = n-1\,,\, n$, we have that $\mathbb{E}\left(dV^{(k)}(\boldsymbol{x})|\mathcal{F}_{t-}\right) = \mathcal{L}_{\boldsymbol{x}, \boldsymbol{c}^{(k)}} V^{(k)}(\boldsymbol{x})\, dt$. Therefore,

$$V^{(k)}\left(\boldsymbol{X}_s^{(n)}\right) - \int_0^s \mathcal{L}_{\boldsymbol{X}^{(n)}, \boldsymbol{c}^{(n)}} V^{(k)}(\boldsymbol{X}_u^{(n)})\, du\,,$$

is a martingale, denoted $\left(M_s^{(k)}\right)_{s\geq 0}$. Using the martingale representation theorem and $X_t^{(n)} = \boldsymbol{x}$, we can rewrite $V^{(k)}\left(\boldsymbol{X}_s^{(n)}\right)$ as follows

$$V^{(k)}\left(\boldsymbol{X}_s^{(n)}\right) \;=\; V^{(k)}\left(\boldsymbol{x}\right) + \int_t^s \mathcal{L}_{\boldsymbol{X}^{(n)}, \boldsymbol{c}^{(n)}} V^{(k)}\left(\boldsymbol{X}_u^{(n)}\right) du + M_s^{(k)} - M_t^{(k)}\,. \tag{17}$$

Let us define the following process:

$$S_s^{(n)} = V^{(n)}\left(\boldsymbol{X}_s^{(n)}\right) - V^{(n-1)}\left(\boldsymbol{X}_s^{(n)}\right)\,.$$

Using the martingale representation of $V^{(k)}$, Eq. (17), we infer that

$$\begin{aligned} S_s^{(n)} \;=\; & V^{(n)}\left(\boldsymbol{x}\right) - V^{(n-1)}\left(\boldsymbol{x}\right) + \int_t^s \left(\mathcal{L}_{\boldsymbol{X}^{(n)}, \boldsymbol{c}^{(n)}} V^{(n)} - \mathcal{L}_{\boldsymbol{X}^{(n)}, \boldsymbol{c}^{(n)}} V^{(n-1)}\right)(\boldsymbol{X}_u^{(n)})\, du \\ & + M_s^{(n)} + M_s^{(n-1)} - M_t^{(n)} - M_t^{(n-1)}\,. \end{aligned}$$

From Proposition 1, $\mathcal{L}_{\boldsymbol{X}^{(n)}, \boldsymbol{c}^{(n)}} V^{(n)} \leq \mathcal{L}_{\boldsymbol{X}^{(n)}, \boldsymbol{c}^{(n)}} V^{(n-1)}$ and the integral is non-positive. Taking expectations and letting $s \to T$, as $\lim_{s\to T} \mathbb{E}\left(S_s^{(n)}\right) \geq 0$ under Assumption (16), we conclude that

$$V^{(n)}\left(\boldsymbol{x}\right) - V^{(n-1)}\left(\boldsymbol{x}\right) \geq 0\,.$$

□

In most cases the assumption 16 is satisfied: for a problem with a finite time horizon, $V^{(n)}(\boldsymbol{X}_T^{(n)}) = V^{(n-1)}(\boldsymbol{X}_T^{(n)}) = U_2(T, \boldsymbol{Y}_T)$ and $Z = 0$. Under additional regularity assumptions, Jacka and Mijatovic (2017) prove the convergence of the controls and the value function. We refer the reader to their article for details. The next section develops the method based on constrained Gaussian process regression for approximating the solution of Eq. (11).

3 The constrained Gaussian process regression (CGPR)

We start by briefly reviewing the general principles of the CGPR. We consider a series of Gaussian processes $\{g^{(n)}(\boldsymbol{x})\,,\,\boldsymbol{x}\in\mathcal{X}\}$, where $n\in\mathbb{N}$. By definition, a Gaussian process is a collection such that for any $d\in\mathbb{N}$ and $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_d\in\mathcal{X}$, the vector $\left(g^{(n)}(\boldsymbol{x}_1),\ldots,g^{(n)}(\boldsymbol{x}_d)\right)$ is a multivariate Gaussian random variable. Without loss of generality, the mean is set to zero and the covariance function is defined by a kernel $k(\boldsymbol{x},\boldsymbol{x}')$:

$$\mathbb{C}\left(g^{(n)}(\boldsymbol{x}),g^{(n)}(\boldsymbol{x}')\right)=k(\boldsymbol{x},\boldsymbol{x}')\quad\forall(\boldsymbol{x},\boldsymbol{x}')\in\mathcal{X}\times\mathcal{X}\,.$$

A necessary and sufficient condition for the function $k:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ to be a valid covariance kernel is that the $d\times d$ matrix of $(k(\boldsymbol{x}_i,\boldsymbol{x}_j))_{i,j=1,\ldots,d}$ is positive semi-definite for all possible samples.

The CGPR provides an approximation $\widehat{V^{(n)}}(\boldsymbol{x})$, of the objective function $V^{(n)}(\boldsymbol{x}):\mathcal{X}\to\mathbb{R}$ after n iterations. We will see that $\widehat{V^{(n)}}(\boldsymbol{x})$ is defined as the a posteriori conditional expectation of $g^{(n)}(\boldsymbol{x})$. Let us detail this. In the CGPR framework, the functional form of $V^{(n)}(\boldsymbol{x})$ is unknown, but its values are observed at certain points of $\bar{\mathcal{X}}$ and are ruled by the PDE of Eq. (11) at other points of $\mathring{\mathcal{X}}$. For this reason, we consider two random samples. We define a first set with d samples of state variables at expiry T, denoted as $\bar{\boldsymbol{X}}=\{\bar{\boldsymbol{x}}_i\}_{i=1,\ldots,d}$ and corresponding noised terminal utility, $\boldsymbol{h}=(h_i)_{i=1,\ldots,d}$. The $\bar{\boldsymbol{x}}_i$'s are randomly sampled from $\bar{\mathcal{X}}$ and $h_i=U_2(\bar{\boldsymbol{x}}_i)+\bar{\epsilon}$ are noised by $\bar{\epsilon}\sim\mathcal{N}(0,\sigma^{*2})$. The second training set contains m samples of state variables before expiry, denoted as $\mathring{\boldsymbol{X}}=\{\mathring{\boldsymbol{x}}_i\}_{i=1,\ldots,m}$ with $\mathring{\boldsymbol{x}}_i\in\mathring{\mathcal{X}}$. The noised values of the PDE of Eq. (11) are grouped in a m-vector, $\boldsymbol{z}^{(n)}=\left(z_i^{(n)}\right)_{i=1,\ldots,m}$ with $z_i^{(n)}=-U_1(\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)})+\mathring{\epsilon}$ and $\mathring{\epsilon}\sim N(0,\sigma^{*2})$.

Assuming noisy observations of $U_2(\bar{\boldsymbol{x}}_i)$ for $\bar{\boldsymbol{x}}_i\in\bar{\boldsymbol{X}}$ and $\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}V^{(n)}(\mathring{\boldsymbol{x}}_i)$ for $\mathring{\boldsymbol{x}}_i\in\mathring{\boldsymbol{X}}$, is a way to introduce a regularization term via a linear shrinkage of the covariance matrix $k(\boldsymbol{X},\boldsymbol{X})$, analogous to Ridge estimators. We will come back to this point later. At each iteration $n\in\mathbb{N}$, we fit a Gaussian process $g^{(n)}$ approximating the value function $V^{(n)}$ over $\bar{\boldsymbol{X}}$ and $\mathring{\boldsymbol{X}}$. The Gaussian process satisfies the following relations on the two training sets:

$$\begin{cases}\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}g^{(n)}(\mathring{\boldsymbol{x}}_i)=z_i^{(n)} & \forall\mathring{\boldsymbol{x}}_i\in\mathring{\boldsymbol{X}},\\ g^{(n)}(\bar{\boldsymbol{x}}_i)=h_i & \forall\bar{\boldsymbol{x}}_i\in\bar{\boldsymbol{X}}.\end{cases}\tag{18}$$

Eq. (18) is a discrete noised version of the system (11). We recognize the structure of a constrained regression problem: we fit $g^{(n)}$ such that $\mathbb{E}\left(g^{(n)}(\bar{\boldsymbol{x}})\right)=U_2(\bar{\boldsymbol{x}})$ for $\bar{\boldsymbol{x}}\in\bar{\boldsymbol{X}}$, under the constraint that $\mathbb{E}\left(\mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}^{(n)}}g^{(n)}(\mathring{\boldsymbol{x}})\right)=-U_1(\mathring{\boldsymbol{x}},\boldsymbol{c}^{(n)})$ for $\mathring{\boldsymbol{x}}\in\mathring{\boldsymbol{X}}$. We denote by $\boldsymbol{H}$ and $\boldsymbol{Z}^{(n)}$, the random d- and m-vectors whose realizations are $\boldsymbol{h}$ and $\boldsymbol{z}^{(n)}$, the right-hand term in Eq. (18).

To summarize, we a priori encode our belief that instances of $g^{(n)}(\boldsymbol{x})$, satisfying Eq. (18), are drawn from a Gaussian process with null mean and covariance $k(\boldsymbol{x},\boldsymbol{x}')$, prior to taking into account observations $\boldsymbol{h}$ and $\boldsymbol{z}^{(n)}$ of terminal utility and HJB PDE. In this Bayesian approach, the estimator of $V^{(n)}(\boldsymbol{x})$ is the a posteriori expectation of $g^{(n)}\,(\boldsymbol{x})$ conditioned by realizations of $\boldsymbol{Z}^{(n)}$ and $\boldsymbol{H}$:

$$\widehat{V^{(n)}}(\boldsymbol{x})=\mathbb{E}\left(g^{(n)}\,(\boldsymbol{x})\mid\boldsymbol{Z}^{(n)}=\boldsymbol{z}^{(n)}\,,\,\boldsymbol{H}=\boldsymbol{h}\right).$$

To estimate $g^{(n)}$, we develop it as a sum a basis functions weighted by random normal weights. This is feasible because the function $g^{(n)}(\boldsymbol{x})$ belongs to a reproducing kernel Hilbert space spanned by eigenfunctions of the kernel (see e.g. Chapter 6 of Rasmussen and Williams, 2006). Without loss of generality, we consider kernels defined by a finite number of eigenfunctions. This assumption is not restrictive as any kernel can be approximated by a finite combination

of eigenfunctions for a given accuracy. This implies that the following results asymptotically hold for any kernel with an infinite number of eigenfunctions. In this case, there exist q basis functions $\boldsymbol{\varphi}(\boldsymbol{x}) = (\varphi_j(\boldsymbol{x}))_{j=1,\dots,q}$ such that

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\varphi}(\boldsymbol{x})^\top \boldsymbol{\varphi}(\boldsymbol{x}') \; , \; \forall \boldsymbol{x}, \, \boldsymbol{x}' \in \mathcal{X} \; .$$

As $g^{(n)}(\boldsymbol{x})$ is a-priori normal with null mean and $\mathbb{C}\left(g^{(n)}(\boldsymbol{x}), g^{(n)}(\boldsymbol{x}')\right) = k(\boldsymbol{x}, \boldsymbol{x}^{'})$, $g^{(n)}(\boldsymbol{x})$ can be written a sum of q functions $\varphi_j(.)$ weighted by a vector $\boldsymbol{w}^{(n)} \sim N(0, I_q)$:

$$g^{(n)}(\boldsymbol{x}) = \sum_{j=1}^{q} w_j^{(n)} \varphi_j(\boldsymbol{x}) = \boldsymbol{w}^{(n)\top} \boldsymbol{\varphi}(\boldsymbol{x}) \; . \tag{19}$$

We define the function $\boldsymbol{\varphi}_{\mathcal{L}^{(n)}}(\boldsymbol{x}) = \left(\mathcal{L}_{\boldsymbol{x}, \boldsymbol{c}^{(n)}} \varphi_j(\boldsymbol{x})\right)_{j=1,\dots,q}$ from $\mathcal{X} \to \mathbb{R}^q$. Using the linearity of the operator (7),c is the scalar product:

$$\mathcal{L}_{\boldsymbol{x}, \boldsymbol{c}^{(n)}} g^{(n)}(\boldsymbol{x}) = \sum_{j=1}^{q} w_j^{(n)} \left(\mathcal{L}_{\boldsymbol{x}, \boldsymbol{c}^{(n)}} \varphi_j(\boldsymbol{x})\right) = \boldsymbol{w}^{(n)\top} \boldsymbol{\varphi}_{\mathcal{L}^{(n)}}(\boldsymbol{x}) \, .$$

Next, we denote by $\mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}}$ the $m \times q$ matrix of $(\mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}})_{i,j} = \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} \varphi_j(\mathring{\boldsymbol{x}}_i)$ where $\mathring{\boldsymbol{x}}_i \in \mathring{\boldsymbol{X}}$ and by $\bar{\boldsymbol{\varphi}}$, the $d \times q$ matrix of $(\bar{\boldsymbol{\varphi}})_{i,j} = \varphi_j(\bar{\boldsymbol{x}}_i)$ with $\bar{\boldsymbol{x}}_i \in \bar{\boldsymbol{X}}$. Conditionally to $\boldsymbol{w}^{(n)}$, the joint distribution of $(\boldsymbol{Z}^{(n)}, \boldsymbol{H})$, the random vectors of right-hand terms in Eq. (18) is a multivariate normal:

$$\left(\begin{array}{c} \boldsymbol{Z}^{(n)} \\ \boldsymbol{H} \end{array} \right) \Bigg| \, \boldsymbol{w}^{(n)} \sim \mathcal{N}\left(\left(\begin{array}{c} \mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}} \boldsymbol{w}^{(n)} \\ \bar{\boldsymbol{\varphi}} \boldsymbol{w}^{(n)} \end{array} \right) , \left(\begin{array}{cc} \sigma^{*2} I_m & \boldsymbol{0} \\ \boldsymbol{0} & \sigma^{*2} I_d \end{array} \right) \right) \; . \tag{20}$$

From the properties of the multivariate normal distribution, the regression weights $\boldsymbol{w}^{(n)}$, conditionally to $\boldsymbol{z}^{(n)}$ and $\boldsymbol{h}$, are realizations of a normal distribution $\boldsymbol{W}^{(n)}$:

$$\boldsymbol{W}^{(n)} | \boldsymbol{z}^{(n)}, \boldsymbol{h} \;\; \sim \;\; \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}^{(n)}, \boldsymbol{h}} \, ; \, \Sigma_{\boldsymbol{z}^{(n)}, \boldsymbol{h}}\right) \; ,$$

where $\boldsymbol{\mu}_{\boldsymbol{z}^{(n)}, \boldsymbol{h}}$, and $\Sigma_{\boldsymbol{z}^{(n)}, \boldsymbol{h}}$ are respectively a q-vector and a $(q \times q)$-matrix equal to

$$\begin{cases} \Sigma_{\boldsymbol{z}^{(n)}, \boldsymbol{h}} & = \left(I_q + \sigma^{*-2} \mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}}^\top \mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}} + \sigma^{*-2} \bar{\boldsymbol{\varphi}}^\top \bar{\boldsymbol{\varphi}}\right)^{-1} \; , \\ \boldsymbol{\mu}_{\boldsymbol{z}^{(n)}, \boldsymbol{h}} & = \sigma^{*-2} \Sigma_{\boldsymbol{z}^{(n)}, \boldsymbol{h}} \left(\mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}}^\top \boldsymbol{z}^{(n)} + \bar{\boldsymbol{\varphi}}^\top \boldsymbol{h}\right) \; . \end{cases} \tag{21}$$

In later developments, we adopt the following notations:

$$\begin{cases} k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}}, \boldsymbol{x}\right) & = \mathring{\boldsymbol{\varphi}}_{\mathcal{L}^{(n)}} \boldsymbol{\varphi}(\boldsymbol{x}) = \left(\mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i)\right)_{i=1,\dots,m} \; , \\ k\left(\bar{\boldsymbol{X}}, \boldsymbol{x}\right) & = \left(k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right)\right)_{j=1,\dots,d} \, , \\ k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right) & = \mathring{\boldsymbol{\varphi}}_{\mathcal{L}} \bar{\boldsymbol{\varphi}}^\top = \left(\mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k(\bar{\boldsymbol{x}}_j, \mathring{\boldsymbol{x}}_i)\right)_{i=1,\dots,m, j=1,\dots,d} \; , \\ k_{\mathcal{L}^{2\,(n)}}\left(\mathring{\boldsymbol{X}}, \mathring{\boldsymbol{X}}\right) & = \mathring{\boldsymbol{\varphi}}_{\mathcal{L}} \mathring{\boldsymbol{\varphi}}_{\mathcal{L}}^\top = \left(\mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} \mathcal{L}_{\mathring{\boldsymbol{x}}_j, \boldsymbol{c}^{(n)}} k(\mathring{\boldsymbol{x}}_i, \mathring{\boldsymbol{x}}_j)\right)_{i,j=1,\dots,m} \; , \\ k\left(\bar{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right) & = \bar{\boldsymbol{\varphi}} \bar{\boldsymbol{\varphi}}^\top = \left(k(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j)\right)_{i,j=1,\dots,d} \; . \end{cases}$$

The next propositions are directly adapted from Hainaut and Vrins (2024) (Propositions 4 and 5), and we refer the reader to this article for the proofs.

Proposition 3. *Let us define $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$, the $(m+d)$-vector:*

$$\boldsymbol{\beta}^{(n)}(\boldsymbol{x}) = \left(\begin{array}{c} k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}}, \boldsymbol{x}\right) \\ k\left(\bar{\boldsymbol{X}}, \boldsymbol{x}\right) \end{array} \right) \; , \tag{22}$$

and the Gram matrix

$$C^{(n)}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right) = \sigma^{*2}I_{m+d} + \begin{pmatrix} k_{\mathcal{L}^{2\,(n)}}\left(\mathring{\boldsymbol{X}},\mathring{\boldsymbol{X}}\right) & k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right) \\ k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right)^{\top} & k\left(\bar{\boldsymbol{X}},\bar{\boldsymbol{X}}\right) \end{pmatrix}. \tag{23}$$

The approximate solution $\widehat{V^{(n)}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{W}^{(n)}|\boldsymbol{z}^{(n)},\boldsymbol{h}}\left(g^{(n)}(\boldsymbol{x})\right)$ *of the HJB Eq. (11) is given by*

$$\widehat{V^{(n)}}(\boldsymbol{x}) = \boldsymbol{\beta}^{(n)}(\boldsymbol{x})^{\top}C^{(n)}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix}\boldsymbol{z}^{(n)}\\ \boldsymbol{h}\end{pmatrix}. \tag{24}$$

Algorithm 1 PI-CGPR : policy improvements algorithm with CGPR.

1. Choose an initial admissible control $\boldsymbol{c}^{(0)}$ and set $\widehat{V^{(0)}}(\boldsymbol{x}) = J(\boldsymbol{x},\boldsymbol{c}^{(0)})$.
2. Sample m points $\mathring{\boldsymbol{x}}_i = (\mathring{t}_i, \mathring{\boldsymbol{y}}_i) \in \mathring{\mathcal{X}}$.
3. Sample d points $\bar{\boldsymbol{x}}_i = (T, \bar{\boldsymbol{y}}_i) \in \bar{\mathcal{X}}$ and set $(h_i = U_2(\bar{\boldsymbol{x}}_i) + \epsilon)_{i=1\ldots d}$ where $\epsilon \sim N(0,\sigma^*)$.
4. Loop on $n \in \mathbb{N}$

   a) Update of $\boldsymbol{z}^{(n)} = \left(z_1^{(n)},...,z_d^{(n)}\right)^{\top}$:

   $$z_i^{(n)} = -U_1(\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}) + \epsilon\,,\ \epsilon \sim N(0,\sigma^*)\,.$$

   b) Compute the Gram matrix, and its inverse:

   $$C^{(n)}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right) = \sigma^{*2}I_{m+d} + \begin{pmatrix} k_{\mathcal{L}^{2\,(n)}}\left(\mathring{\boldsymbol{X}},\mathring{\boldsymbol{X}}\right) & k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right) \\ k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right)^{\top} & k\left(\bar{\boldsymbol{X}},\bar{\boldsymbol{X}}\right) \end{pmatrix} \tag{25}$$

   Calculate $\boldsymbol{\beta}^{(n)}\left(\mathring{\boldsymbol{x}}_i\right) = \begin{pmatrix} k_{\mathcal{L}^{(n)}}\left(\mathring{\boldsymbol{X}},\mathring{\boldsymbol{x}}_i\right) \\ k\left(\bar{\boldsymbol{X}},\mathring{\boldsymbol{x}}_i\right)\end{pmatrix}$ and intermediate value functions

   $$\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}_i) = \boldsymbol{\beta}^{(n)}(\mathring{\boldsymbol{x}}_i)^{\top}C^{(n)}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix}\boldsymbol{z}^{(n)}\\ \boldsymbol{h}\end{pmatrix}. \tag{26}$$

   c) find the improvement of $\boldsymbol{c}^{(n)}$

   $$\boldsymbol{c}^{(n+1)} = \sup_{\boldsymbol{c}\in A}\left(U_1(\mathring{\boldsymbol{x}},\boldsymbol{c}) + \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}}\widehat{V^{(n)}}(\mathring{\boldsymbol{x}})\right)\ \forall \mathring{\boldsymbol{x}} \in \mathring{\boldsymbol{X}}. \tag{27}$$

   d) Stop if $\frac{1}{m}\sum_{i=1}^{m}\left(\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}_i) - \widehat{V^{(n-1)}}(\mathring{\boldsymbol{x}}_i)\right)^2 \leq \delta$, for a small $\delta \in \mathbb{R}^+$.

As previously mentioned, the Gaussian noises added to h_i's and z_i's introduce a regularization term, that is the diagonal matrix $\sigma^{*2}I_{m+d}$, in the Gram matrix (23). This term stabilizes a possibly ill-conditioned matrix. The standard deviation σ^* may then be viewed as a parameter tuning the numerical robustness of the CGPR. Of course, this affects the accuracy of the solution but we will see in the numerical illustrations that its impact is limited. The next proposition provides the conditional variance of the estimator. For a proof, we again refer the reader to Proposition 5 of Hainaut and Vrins (2024).

Proposition 4. *Let $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ be the $(m+d)$ vector defined in Eq. (22). The conditional variance of $g^{(n)}(\boldsymbol{x})$ is equal to*

$$\mathbb{V}_{\boldsymbol{W}^{(n)}|,\boldsymbol{z}^{(n)},\boldsymbol{h}}\left(g^{(n)}(\boldsymbol{x})\right) = k(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{\beta}^{(n)}(\boldsymbol{x})^{\top} C^{(n)}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right)^{-1}\boldsymbol{\beta}^{(n)}(\boldsymbol{x})\,. \tag{28}$$

The frame 1 presents the full algorithm for solving the HJB equation by policy improvements and constrained Gaussian process regressions (PI-CGPR algorithm).

After the completion of the algorithm, we can measure the errors on the inner and boundary domains by computing the average residuals of Equations (11):

$$\mathring{Err} = \frac{1}{m}\sum_{\mathring{\boldsymbol{x}}_i\in\mathring{\boldsymbol{X}}}\left|\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}_i) + U_1(\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)})\right|, \tag{29}$$

$$\bar{Err} = \frac{1}{d}\sum_{\bar{\boldsymbol{x}}_i\in\bar{\boldsymbol{X}}}\left|\widehat{V^{(n)}}(\bar{\boldsymbol{x}}_i) - U_2(\bar{\boldsymbol{x}}_i)\right|. \tag{30}$$

4 PI-CGPR algorithm with Gaussian kernels

Solving the intermediate PDE's requires applying the differential operator $\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}^{(n)}}$ twice to the kernel function. Although many variants exist (see e.g. Beckers, 2021), we choose a Gaussian kernel for its analytical tractability. Let us denote $\boldsymbol{x} = (t,\boldsymbol{y})$, $\boldsymbol{y} = (y_1,...,y_{p-1})$, $\tilde{\boldsymbol{x}} = (\tilde{t},\tilde{\boldsymbol{y}})$ and $\tilde{\boldsymbol{y}} = (\tilde{y}_1,...,\tilde{y}_{p-1})$. The Gaussian kernel has the general form:

$$k(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \exp\left(\eta_0 - \frac{\left(t-\tilde{t}\right)^2}{2\,\eta_t^2} - \sum_{k=1}^{p-1}\frac{\left(y_k-\tilde{y}_k\right)^2}{2\,\eta_k^2}\right) \quad \forall \boldsymbol{x},\tilde{\boldsymbol{x}}\in\mathcal{X}\,, \tag{31}$$

where $\boldsymbol{\eta} = (\eta_0,\eta_t,\eta_1,...,\eta_{p-1})^{\top}$is a $(p+1)$-vector in $\mathbb{R}_0^{p+1}$ of hyper-parameters. The coefficients $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ of the CGPR have, in this case, a closed-form expression. The last d elements of $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ are equal to $\left(k(\bar{\boldsymbol{x}}_i,\boldsymbol{x})\right)_{i=1,...,d}$. The first m items of $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ require the calculation of $\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})$. For this purpose, we introduce the following p-vector:

$$\boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \begin{pmatrix} \left(y_1-\tilde{y}_1\right)/\eta_1^2 \\ \vdots \\ \left(y_{p-1}-\tilde{y}_{p-1}\right)/\eta_{p-1}^2 \end{pmatrix}, \tag{32}$$

and the $(p-1)\times(p-1)$ matrix:

$$\boldsymbol{H}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \begin{pmatrix} \frac{(y_1-\tilde{y}_1)^2-\eta_1^2}{\eta_1^4} & \frac{(y_1-\tilde{y}_1)(y_2-\tilde{y}_2)}{\eta_1^2\eta_2^2} & \cdots & \frac{(y_1-\tilde{y}_1)(y_{p-1}-\tilde{y}_{p-1})}{\eta_1^2\eta_{p-1}^2} \\ \frac{(y_1-\tilde{y}_1)(y_2-\tilde{y}_2)}{\eta_1^2\eta_2^2} & \ddots & \ddots & \frac{(y_2-\tilde{y}_2)(y_{p-1}-\tilde{y}_{p-1})}{\eta_2^2\eta_{p-1}^2} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{(y_1-\tilde{y}_1)(y_{p-1}-\tilde{y}_{p-1})}{\eta_1^2\eta_{p-1}^2} & \frac{(y_2-\tilde{y}_2)(y_{p-1}-\tilde{y}_{p-1})}{\eta_2^2\eta_{p-1}^2} & \cdots & \frac{(y_{p-1}-\tilde{y}_{p-1})^2-\eta_{p-1}^2}{\eta_{p-1}^4} \end{pmatrix}. \tag{33}$$

By direct differentiation, we obtain the gradient and Hessian of the kernel:

$$\begin{cases} \nabla_{\tilde{\boldsymbol{y}}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) &= \boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}})\,k(\boldsymbol{x},\tilde{\boldsymbol{x}})\,, \\ \mathcal{H}_{\tilde{\boldsymbol{y}}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) &= \boldsymbol{H}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}})\,k(\boldsymbol{x},\tilde{\boldsymbol{x}})\,. \end{cases}$$

We infer from these expressions, that $\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})$ can be rewritten in terms of $\boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}})$ and $\boldsymbol{H}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}})$:

$$\begin{aligned}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \Big(\left(t-\tilde{t}\right)/\eta_t^2 - \alpha + \boldsymbol{\mu}_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})^\top \boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) \\ + \frac{1}{2}\mathrm{tr}\left(\Sigma_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})\Sigma_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})^\top \boldsymbol{H}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)\Big)\, k(\boldsymbol{x},\tilde{\boldsymbol{x}})\,, \end{aligned} \tag{34}$$

which allows us to evaluate analytically the first m elements of $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$: $\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\mathring{\boldsymbol{x}}_i)$ for $i=1,...,m$. The construction of the Gram matrix, $C^{(n)}$ requires to compute the matrix $k_{\mathcal{L}^{2\,(n)}}\left(\mathring{\boldsymbol{X}},\mathring{\boldsymbol{X}}\right)$ of $\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}\mathcal{L}_{\mathring{\boldsymbol{x}}_j,\boldsymbol{c}^{(n)}}k(\mathring{\boldsymbol{x}}_i,\mathring{\boldsymbol{x}}_j)$ for $i,j=1,...,m$. This step can be done analytically using the relation

$$\begin{aligned}\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}^{(n)}}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \partial_t \mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) \\ -\alpha\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) + \boldsymbol{\mu}_{\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{c}^{(n)})^\top \nabla_{\boldsymbol{y}}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) \\ +\frac{1}{2}\mathrm{tr}\left(\Sigma_{\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{c}^{(n)})\Sigma_{\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{c}^{(n)})^\top \mathcal{H}_{\boldsymbol{y}}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)\,, \end{aligned} \tag{35}$$

where the derivative w.r.t. time of $\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})$ is equal to

$$\partial_t \mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) \;=\; \frac{1}{\eta_t^2}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) - \left(\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)\frac{\left(t-\tilde{t}\right)}{\eta_t^2}\,. \tag{36}$$

On the other hand, the gradient $\nabla_{\boldsymbol{y}}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \left(\partial_{y_k}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)_{k=1,...,p-1}$ in Eq. (35) is a $(p-1)$ vector filled with:

$$\begin{aligned}\partial_{y_k}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right) = \Big[\boldsymbol{e}_k^\top \Sigma_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})\Sigma_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})^\top \boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) \\ +\boldsymbol{e}_k^\top \boldsymbol{\mu}_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})\Big]\frac{k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right)}{\eta_k^2} - \mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right)\,\boldsymbol{e}_k^\top \boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}})\,, \end{aligned} \tag{37}$$

where $\boldsymbol{e}_k$'s are canonical vectors. Let us define the following matrix:

$$\boldsymbol{H}_{\boldsymbol{y}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \boldsymbol{H}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) + \begin{pmatrix} 2/\eta_1^2 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & 2/\eta_{p-1}^2 \end{pmatrix}.$$

The Hessian $\mathcal{H}_{\boldsymbol{y}}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \left(\partial_{y_l}\partial_{y_k}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)_{k,l=1,...,p-1}$ is a $(p-1)\times(p-1)$ matrix with elements:

$$\begin{aligned}\partial_{y_l}\partial_{y_k}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right) = \left(\boldsymbol{e}_k^\top \Sigma_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})\Sigma_{\boldsymbol{Y}}(\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)})^\top \boldsymbol{e}_l\right)\frac{k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right)}{\eta_k^2\eta_l^2} \\ -\left(\partial_{y_k}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right)\right)\boldsymbol{e}_l^\top \boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) - \left(\partial_{y_l}\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right)\right)\boldsymbol{e}_k^\top \boldsymbol{d}_{\tilde{\boldsymbol{y}}}(\boldsymbol{x},\tilde{\boldsymbol{x}}) \\ -\mathcal{L}_{\tilde{\boldsymbol{x}},\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\tilde{\boldsymbol{x}}\right)\left(\boldsymbol{e}_l^\top\left(\boldsymbol{H}_{\boldsymbol{y}}(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)\boldsymbol{e}_k\right)\,. \end{aligned} \tag{38}$$

In the numerical illustrations, we consider optimization problems for which the optimal intermediate controls, solving Eq. (10), admit an explicit formulation in terms of the intermediate approximated payoff and its first and second order derivatives, i.e.

$$\boldsymbol{c}^{(n)} = \boldsymbol{c}^{(n)}\left(\boldsymbol{x},\partial_t\widehat{V^{(n-1)}}(\boldsymbol{x}),\nabla_{\boldsymbol{y}}\widehat{V^{(n-1)}}(\boldsymbol{x}),\mathcal{H}_{\boldsymbol{y}}\widehat{V^{(n-1)}}(\boldsymbol{x})\right)\,.$$

This avoid us having to numerically solve any optimization problem. The gradient and Hessian of $\widehat{V^{(n-1)}}(\boldsymbol{x})$ also admits analytical expressions inferred from Eq. (37) and (38). We provide

them in Appendix A.

To conclude this section, we remark that Eq. (24) and (28) correspond to the conditional expectation and variance of $g^{(n)}(\boldsymbol{x}) \mid \boldsymbol{z}^{(n)}, \boldsymbol{h}$. All variables being Gaussian, the triplet $\left(g^{(n)}(\boldsymbol{x}), \boldsymbol{Z}^{(n)}, \boldsymbol{H}\right)$ where n is the last iteration of Algorithm 1, has a multivariate normal distribution:

$$\begin{bmatrix} g^{(n)}(\boldsymbol{x}) \\ \boldsymbol{Z}^{(n)} \\ \boldsymbol{H} \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x},\boldsymbol{x}) & \boldsymbol{\beta}(\boldsymbol{x})^\top \\ \boldsymbol{\beta}(\boldsymbol{x}) & C\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right) \end{bmatrix}\right).$$

In theory, this observation would allows us to optimize hyper-parameters $\boldsymbol{\eta}$ by maximizing the marginal log-likelihood of $(\boldsymbol{Z}^{(n)}, \boldsymbol{H})$:

$$\begin{aligned} \log L(\boldsymbol{z}^{(n)}, \boldsymbol{h}) = & \frac{(m+d)}{2}\log 2\pi - \frac{1}{2}\log\left|C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)\right| \\ & - \frac{1}{2}\begin{pmatrix} \boldsymbol{z}^{(n)} \\ \boldsymbol{h} \end{pmatrix}^\top C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix} \boldsymbol{z}^{(n)} \\ \boldsymbol{h} \end{pmatrix}. \end{aligned} \tag{39}$$

In practice, this maximization is time-consuming and subject to numerical instabilities because the log-likelihood is a non-convex function of hyper-parameters. We consider an alternative efficient method in which the hyper-parameters minimize the errors of approximation on the inner and boundary sample sets. We denote by $V^{(n^*,\boldsymbol{\eta})}(.)$ the optimal value function obtained after n^* iterations of Algorithm 1. The inner and terminal total errors are defined as follows:

$$\begin{aligned} \mathring{TErr}(\boldsymbol{\eta}) &= \sum_{\mathring{\boldsymbol{x}}_i \in \mathring{\boldsymbol{X}}} \left|\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}}\widehat{V^{(n^*,\boldsymbol{\eta})}}\left(\mathring{\boldsymbol{x}}_i\right) - z_i^{(n^*)}\right|, \\ \bar{TErr}(\boldsymbol{\eta}) &= \sum_{\bar{\boldsymbol{x}}_i \in \bar{\boldsymbol{X}}} \left|\widehat{V^{(n^*)}}\left(\bar{\boldsymbol{x}}_i\right) - h_i\right|. \end{aligned}$$

The total error is the sum of inner and terminal errors:

$$TErr(\boldsymbol{\eta}) = \mathring{TErr}(\boldsymbol{\eta}) + \bar{TErr}(\boldsymbol{\eta}). \tag{40}$$

We run a standard gradient descent procedure and update hyper-parameters $\boldsymbol{\eta}^{(k)}$ at each iteration:

$$\boldsymbol{\eta}^{(k)} = \boldsymbol{\eta}^{(k-1)} - \rho\,\nabla_{\boldsymbol{\eta}^{(k-1)}} TErr\left(\boldsymbol{\eta}^{(k-1)}\right), \tag{41}$$

where ρ is a learning rate. The gradient in this last expression does not admit any simple analytical expression and is, for this reason, numerically computed. We initialize the gradient descent with $\eta_0^{(0)} = 0$, and $\eta_t^{(0)}$, $\left(\eta_j^{(0)}\right)_{j=1,\dots,p-1}$ are respectively set to the standard deviations of $\left\{\mathring{t}_i\right\}_{i=1,\dots,m}$ and $\{\mathring{\boldsymbol{y}}_i\}_{i=1,\dots,m}$. We will see that this method yields good results for the two optimization problems analyzed in the following section.

5 Numerical illustration

We test the capacity of the PI-CGPR algorithm to solve the linear-quadratic regulator and the consumption-investment problems. As they admit closed-form solutions for the value function and the controls, we can accurately measure the errors of approximation induced by the PI-CGPR algorithm.

5.1 Optimal investment-consumption problem

In this section, we focus on the optimal consumption-allocation problem. An investor has an initial wealth Y_0, invested in a portfolio of cash, earning the risk-free rate $r \in \mathbb{R}^+$, and v stocks, $\boldsymbol{S}_t = \left(S_t^{(1)}, ..., S_t^{(v)}\right)$. The stock prices are ruled by a geometric Brownian motion:

$$\frac{d\boldsymbol{S}_t}{\boldsymbol{S}_t} = \boldsymbol{\mu}\, dt + \Sigma\, d\boldsymbol{B}_t\,,$$

where $\boldsymbol{\mu} = (\mu_1, ..., \mu_v) \in \mathbb{R}^v$ and $\Sigma \in \mathbb{R}^{v\times v}$ and $\frac{d\boldsymbol{S}_t}{\boldsymbol{S}_t}$ is an element-wise division. At time t, the fractions of the fund invested in stocks are stored in a vector denoted $\boldsymbol{\pi}_t = \left(\pi_t^{(1)}, \ldots, \pi_t^{(v)}\right)^\top$. The investor also withdraws a percentage b_t of Y_t for its own consumption. The wealth process is therefore ruled by the following SDE:

$$\begin{aligned} dY_t &= Y_t \boldsymbol{\pi}_t^\top \frac{d\boldsymbol{S}_t}{\boldsymbol{S}_t} + Y_t\Big(1 - \boldsymbol{\pi}_t^\top \mathbf{1}_v\Big) r\, dt - b_t Y_t\, dt \\ &= \Big(r + (\boldsymbol{\mu} - r)^\top \boldsymbol{\pi}_t - b_t\Big) Y_t dt + Y_t \boldsymbol{\pi}_t^\top \Sigma\, d\boldsymbol{B}_t\,, \end{aligned} \tag{42}$$

where $\mathbf{1}_v$ is the unit vector of dimension v. In this problem, we have a single risk process[3] $(Y_t)_{t\geq 0}$ whereas the control is a $(v+1)$ vector $\boldsymbol{c}_t = (b_t, \boldsymbol{\pi}_t)^\top$ ($p = 2$ and $l = v+1$). In this setting, the drift and volatility matrix of risk processes are:

$$\boldsymbol{\mu_Y}(t, Y_t, \boldsymbol{c}_t) = \Big(r + (\boldsymbol{\mu} - r)^\top \boldsymbol{\pi}_t - b_t\Big) Y_t\ ,\ \Sigma_{\boldsymbol{Y}}(t, Y_t, \boldsymbol{c}_t) = Y_t \boldsymbol{\pi}_t^\top \Sigma\,.$$

The running utility of consumption and the utility of the terminal wealth, are concave and invertible functions denoted by $U_1(.)$ and $U_2(.)$. We choose utilities with a constant relative risk aversion (CRRA) coefficient γ:

$$U_1\left(b_t y\right) = \frac{(b_t y)^\gamma}{\gamma}\ ,\ U_2\left(y\right) = \frac{(y)^\gamma}{\gamma}\,.$$

The payoff is the sum of the discounted expected utility of consumption and terminal wealth over $[0, T]$:

$$J(t, y, \boldsymbol{c}) = \mathbb{E}\left(\int_t^T e^{-\alpha(s-t)} \frac{(b_s Y_s)^\gamma}{\gamma} ds + e^{-\alpha(T-t)} \frac{(Y_T)^\gamma}{\gamma} \,|\, Y_t = y\right)\,.$$

The linear differential operator is in this case defined by

$$\begin{aligned} \mathcal{L}_{\boldsymbol{x},\boldsymbol{c}}\cdot = \partial_t \cdot - \alpha \cdot + \Big(r + (\boldsymbol{\mu} - r)^\top \boldsymbol{\pi}_t - b_t\Big)\, y\, \nabla_{\boldsymbol{y}} \cdot \\ + \frac{1}{2} y^2 \mathrm{tr}\left(\boldsymbol{\pi}_t^\top \Sigma\, \Sigma^\top \boldsymbol{\pi}_t\, \mathcal{H}_{\boldsymbol{y}} \cdot\right)\,. \end{aligned} \tag{43}$$

The inner and terminal domains are $\mathring{\mathcal{X}} = [0, T) \times \mathbb{R}^+$ and $\bar{\mathcal{X}} = T \times \mathbb{R}^+$. Noting $\boldsymbol{x} = (t, y)$, the value function, $V(\boldsymbol{x}) = \max_{c\in\mathcal{A}} J(\boldsymbol{x}, \boldsymbol{c})$, is the solution of the following HJB equation:

$$\sup_{\boldsymbol{c}\in A}\left(\frac{(b\mathring{y})^\gamma}{\gamma} + \mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}} V(\mathring{\boldsymbol{x}})\right) = 0 \quad \forall \mathring{\boldsymbol{x}} \in \mathring{\mathcal{X}}\,, \tag{44}$$

$$V(\bar{\boldsymbol{x}}) = \frac{(\bar{y})^\gamma}{\gamma} \quad \forall \bar{\boldsymbol{x}} \in \bar{\mathcal{X}}\,. \tag{45}$$

[3] As $(Y_t)_{t\geq 0}$ is a scalar process, we do not use bold notation.

The optimal controls, $\boldsymbol{c}^*(t,y) = (b^*(t,y), \boldsymbol{\pi}^*(t,y))$, are obtained by canceling the derivative of Eq. (44) w.r.t. to consumption rate and asset proportions:

$$\begin{cases} b^*(t,y) & = \frac{1}{y}\left(\partial_y V(\boldsymbol{x})\right)^{\frac{1}{\gamma-1}} , \\ \boldsymbol{\pi}^*(t,y) & = -\left(\Sigma\Sigma^\top\right)^{-1}(\boldsymbol{\mu}-r)\,\frac{1}{y}\frac{\partial_y V(\boldsymbol{x})}{\partial_{yy} V(\boldsymbol{x})} . \end{cases}$$

The next proposition provides the analytical formula of the value function and controls.

Proposition 5. *$V(t,y) = \max_{c\in\mathcal{A}} J(t,y,\boldsymbol{c})$ is a power function of the wealth:*

$$V(t,y) \;=\; A(t)\frac{y^\gamma}{\gamma}, \tag{46}$$

where $A(t) : [0,T] \to \mathbb{R}^+$ is given by

$$A(t) \;=\; \left(\frac{1-\gamma}{\theta}\left(e^{\frac{\theta}{1-\gamma}(T-t)} - 1\right) + e^{\frac{\theta}{1-\gamma}(T-t)}\right)^{1-\gamma} . \tag{47}$$

The optimal consumption and investment policy are respectively equal to:

$$\begin{cases} b_t^* = & A(t)^{\frac{1}{\gamma-1}} , \\ \boldsymbol{\pi}_t^* = & \left(\Sigma\Sigma^\top\right)^{-1}\frac{(\boldsymbol{\mu}-r)}{(1-\gamma)} . \end{cases} \tag{48}$$

We can check that the value function (46) satisfies the HJB equation, with the optimal control (48).

The analytical solution of Proposition 5 will serve as a benchmark for the PI-CGPR algorithm 1. For the consumption-investment problem, the step 4.b) of the algorithm becomes:

$$\begin{cases} b^{(n)}(t,y) & = \frac{1}{y}\left(\partial_y \widehat{V^{(n-1)}}(t,y)\right)^{\frac{1}{\gamma-1}} , \\ \boldsymbol{\pi}^{(n)}(t,y) & = -\left(\Sigma\Sigma^\top\right)^{-1}(\boldsymbol{\mu}-r)\,\frac{1}{y}\frac{\partial_y \widehat{V^{(n-1)}}(t,y)}{\partial_{yy}\widehat{V^{(n-1)}}(t,y)} , \end{cases} \tag{49}$$

where $\widehat{V^{(n-1)}}(\boldsymbol{x})$ is the CGPR approximation.

In numerical tests, we consider a market with two stocks. Table 1 provides the parameters defining the asset dynamics, the risk-free rate and the utility functions. Table 2 reports the parameters of the kernel. σ^* is set to 10^{-4}. This value offers a good trade-off between accuracy and robustness. In practice, σ^* should be set to the lowest possible value for which the Gram matrix $C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)$ is not ill-conditioned. Considering very large sample sets is often the origin of numerical errors when inverting it. In this case, increasing σ^* gives more weight to the regularization term in the Gram matrix and prevents numerical instabilities during the inversion.

α	4.00%	γ	0.30
r	3.00%	μ_1	5.00%
μ_2	7.00%	T	2 to 8 years
$\Sigma = \begin{pmatrix} 0.20 & -0.05 \\ 0 & 0.20 \end{pmatrix}$			

Table 1: Parameters of the investment-consumption problem.

$\mathring{t}$	uniform distribution $[0, T)$
$\mathring{y}$	uniform distribution $[0, 500)$
σ^*	10^{-4}
$\eta_0^{(0)}$	0
$\eta_t^{(0)}$	1.5×standard deviation of $\mathring{t}$, $\sqrt{T^2/12}$
$\eta_1^{(0)}$	0.75×standard deviation of $\mathring{y}$, $\sqrt{500^2/12}$
$c^{(0)}(t, y)$	10%
$\pi^{(1)(0)}(t, y)$	10%
$\pi^{(2)(0)}(t, y)$	10%

Table 2: Parameters of the PI-CGPR for the investment-consumption problem.

Table 3 reports the mean relative errors (MREs) between approximated and exact value functions for sample sets $\mathring{\boldsymbol{X}}$ of various sizes m. The MRE is computed as:

$$\text{MRE} \quad = \quad \frac{1}{m} \sum_{\mathring{\boldsymbol{x}}_i \in \mathring{\boldsymbol{X}}} \left| \frac{\widehat{V^{(n)}(\mathring{\boldsymbol{x}}_i)} - V(\mathring{\boldsymbol{x}}_i)}{V(\mathring{\boldsymbol{x}}_i)} \right| . \tag{50}$$

The PI-CGPR algorithm is stopped when the average quadratic variation of $\widehat{V^{(n)}} - \widehat{V^{(n-1)}}$, defined in step d) of Algorithm 1, falls below δ=1%. Whatever the configuration of the problem, the MRE is smaller than 0.80%, which is an acceptable level for a numerical approximation. For a fixed sample size m, the MRE increases with the time horizon, T. Increasing the sample size does not have a significant impact on the MREs. Table 3 reveals that convergence is attained after at most three iterations.

	m and $d = m/2$					
	300	400	500	300	400	500
T	MRE (%)			# of iterations		
2.0	0.23	0.20	0.31	2	2	2
4.0	0.40	0.31	0.44	3	3	3
6.0	0.60	0.42	0.62	3	3	3
8.0	0.82	0.69	0.80	3	3	3

Table 3: Mean relative errors (MREs) in % and the number of iterations to attain convergence.

The computational times[4], reported in Table 4, vary from 47s to 127s seconds. The time-consuming step is the optimization of hyper-parameters, which requires several runs of the policy improvement algorithm 1. We perform 10 iterations of the gradient descent, Equation (41), with a learning rate $\rho = 0.01$. As seen in Table 3, this is sufficient to achieve good overall accuracy. As $\boldsymbol{\eta}$ can be assimilated to standard deviations, the starting values in Table 2 are proportional to the standard deviations of $\mathring{t}$ and $\mathring{y}$. The gradient with respect to $\boldsymbol{\eta}$ in Equation (41) is computed by finite differences, with increments of $\boldsymbol{\eta}$ set to 0.20 .

[4]Configuration: laptop with an Intel Core I7 and 32Mb of RAM.

	m and $d = m/2$		
	300	400	500
T	Time (seconds)		
2.0	47.04	68.04	90.30
4.0	63.02	95.33	123.11
6.0	65.86	93.15	122.29
8.0	66.36	100.78	126.69

Table 4: Computational time, optimal consumption-investment problem.

Table 5 reports the average errors $\mathring{Err}$ and $\bar{Err}$ on the inner and boundary domains, as defined in Equations (29) and (30). For problems without an analytical solution, these are the only available indicators for measuring the accuracy of the PI-CGPR. Contrary to MREs, we do not observe any clear trend between T and m but the average errors are very small, regardless of the setting of the PI-CGPR.

	m and $d = m/2$					
	300	400	500	300	400	500
T	$\mathring{Err}$			$\bar{Err}$		
2.0	0.0022	0.1520	0.0119	0.0021	0.0267	0.0065
4.0	0.0020	0.0371	0.0059	0.0022	0.0234	0.0063
6.0	0.0026	0.0347	0.0054	0.0024	0.0196	0.0070
8.0	0.0074	0.0766	0.0049	0.0026	0.0219	0.0069

Table 5: Errors of approximation on the inner and boundary domains.

Table 6 provides the hyper-parameters of the Gaussian kernel, obtained with the gradient descent procedure. We see that η_t increases with T and that η_0 is significantly different from zero. η_1 remains close to the starting value of 108.25.

	η_1				η_t		
T/m	300	400	500	T/m	300	400	500
2.0	110.2534	110.2638	109.8532	2.0	2.8663	2.9189	2.4658
4.0	110.0529	109.4682	109.8511	4.0	3.5320	2.9072	3.3330
6.0	109.0532	110.3639	109.6528	6.0	3.3981	4.6490	3.9983
8.0	109.0542	108.6537	109.6552	8.0	4.2652	3.8647	4.8657

	η_0		
T/m	300	400	500
2.0	2.0007	2.0156	1.5991
4.0	1.8001	1.1897	1.5210
6.0	0.8002	2.1116	1.4001
8.0	0.8017	0.3976	1.4007

Table 6: Hyper-parameters obtained after 10 iterations of gradient descent.

To gain better insight into the PI-CGPR's capacity to provide an accurate approximation, we focus on the consumption-investment problem with $T = 5$ and m=500. Table 6 presents the evolution of inner and boundary errors, (29) and (30), for each iteration of the policy improvement algorithm. It confirms the convergence of the procedure and the reduction of errors after the update of controls.

	$\mathring{Err}$	$\bar{Err}$	$\frac{1}{m}\sum_{i=1}^{m}\left(\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}_i) - \widehat{V^{(n-1)}}(\mathring{\boldsymbol{x}}_i)\right)^2$
1	1.8902	0.0082	1275.56
2	0.0696	0.0075	25.55
3	0.0078	0.0064	0.03
4	0.0017	0.0062	0.00

Table 7: Evolution of the average inner and boundary errors by PI iterations.

Figure 1 displays the absolute spreads between the exact and approximated value functions, $\left|\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}) - V(\mathring{\boldsymbol{x}})\right|$, on the domain $[0,5]\times[50,200]$. On this subset, errors are of the same order and small nearly everywhere. We observe a few small local maxima located in areas not containing any sampled points in $\mathring{\boldsymbol{X}}$. We also observe an increase in errors in the upper right part of the plot. This is due to the propagation of errors near the boundaries.

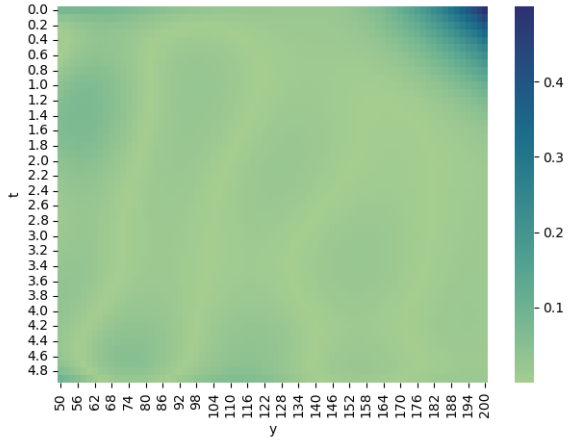

Figure 1: Inner errors for $T = 5$ years and m =500.

Figure 2 compares approximated and exact value functions and controls in a single scenario where the wealth remains equal to 100, still for $T = 5$ and $m = 500$. We observe a clear similarity between results obtained with the PI-CGPR algorithm and their analytical counterparts.

To detect where the PI-CGPR is less efficient, we plot heatmaps of relative absolute errors for $V(.)$ and optimal controls, b_t^*, $\pi_t^{(1)*}$, $\pi_t^{(2)*}$ in Figure 3 and 4. We display heatmaps on the domain $[0,5]\times[50,200]$. These graphs confirm the accuracy of the PI-CGPR, on this subset of the training domain. The relative spread between exact and approximated value functions does not exceed 0.08%. Approximated and exact controls are also close in most configurations. The largest gaps are observed near the starting date and for the highest values of y. Other tests reveal that the accuracy of approximation declines for pairs (t, y) close to the boundaries of the training domain.

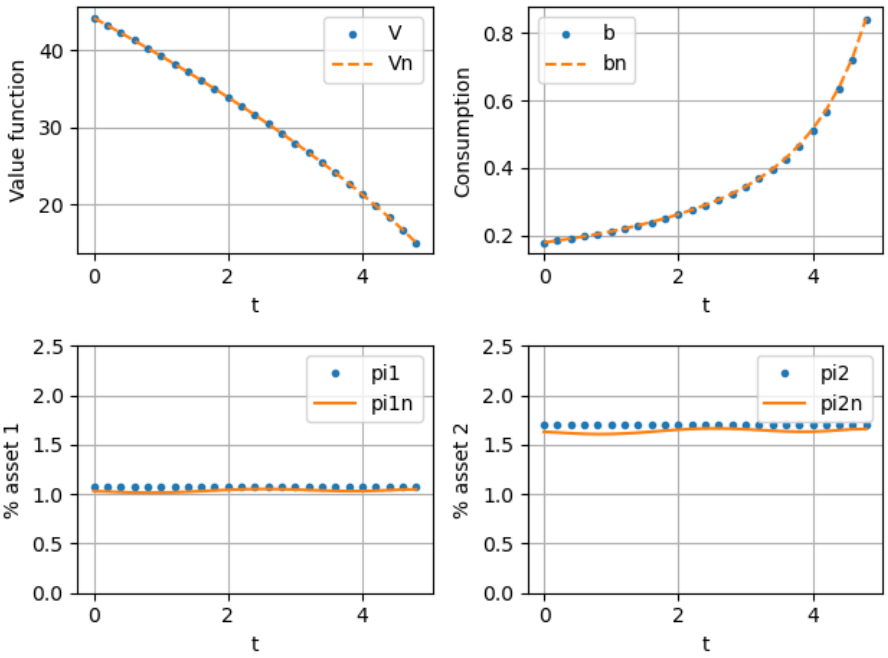

Figure 2: Comparison of exact and approximated value functions, consumption and optimal investments in one sample path: $y_t = 100$ and $T = 5$ years.

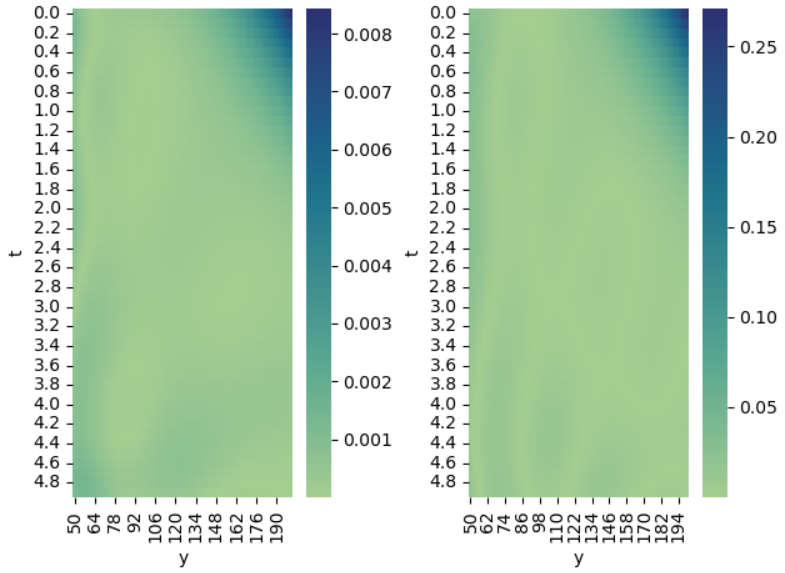

Figure 3: Left plot: heatmap of $\left|\frac{\widehat{V^{(n)}}(t,y)-V(t,y)}{V(t,y)}\right|$. Right plot: heatmap of $\left|\frac{b^{(n)}(t,y)-b^*(t,y)}{b^*(t,y)}\right|$.

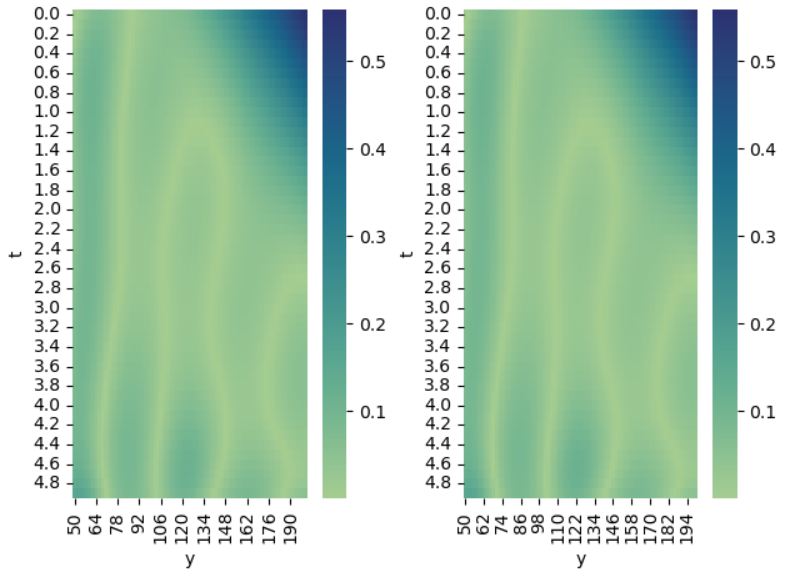

Figure 4: Left plot: heatmap of $\left|\frac{\pi_t^{(1)(n)}-\pi_t^{(1)*}}{\pi_t^{(1)*}}\right|$. Right plot: heatmap of $\left|\frac{\pi_t^{(2)(n)}-\pi_t^{(2)*}}{\pi_t^{(2)*}}\right|$

5.2 The linear-quadratic regulator

In this section, we test the capacity of the PI-CGPR algorithm to optimize the linear regulation of a system with quadratic reward and penalty utilities. We consider a $(p-1)$ vector of controlled risk processes, $\boldsymbol{Y}_t=\left(Y_t^{(1)},\ldots,Y_t^{(p-1)}\right)$ ruled by the SDEs:

$$d\boldsymbol{Y}_t \quad = \quad (M\boldsymbol{Y}_t+\boldsymbol{c}_t)\,dt+\Sigma\,d\boldsymbol{B}_t\,,$$

where $M:\mathbb{R}^{(p-1)\times(p-1)}$ and $\Sigma\in\mathbb{R}^{(p-1)\times(p-1)}$. The control, $\boldsymbol{c}_t$, and the vector of Brownian motions, $\boldsymbol{B}_t$, are of dimension $(p-1)$. In this setting, the drift and volatility matrix of risk processes are:

$$\boldsymbol{\mu}_{\boldsymbol{Y}}(t,\boldsymbol{Y}_t,\boldsymbol{c}_t)=M\boldsymbol{Y}_t+\boldsymbol{c}_t\;,\;\Sigma_{\boldsymbol{Y}}(t,\boldsymbol{Y}_t,\boldsymbol{c}_t)=\Sigma\,.$$

Over a time horizon T, we aim to maximize the utility of $\boldsymbol{Y}_T$ and minimize the running costs of controls. We consider running and terminal utilities which have quadratic forms

$$\begin{aligned}U_1(t,\boldsymbol{y},\boldsymbol{c})&=-\boldsymbol{c}^\top Q_1\,\boldsymbol{c}\,,\\ U_2(t,\boldsymbol{y})&=\boldsymbol{y}^\top Q_2\,\boldsymbol{y}\,,\end{aligned}$$

for some positive invertible, positive definite matrix $Q_1,Q_2\in\mathbb{R}^{(p-1)\times(p-1)}$. The payoff is the sum of the discounted expected utilities of controls and terminal values of risk processes:

$$J(t,\boldsymbol{y},\boldsymbol{c}) \quad = \quad \mathbb{E}\left(-\int_t^T e^{-\alpha(s-t)}\boldsymbol{c}_s^\top\,Q_1\,\boldsymbol{c}_s\,ds+e^{-\alpha(T-t)}\boldsymbol{Y}_T^\top\,Q_2\,\boldsymbol{Y}_T\,|\,\boldsymbol{Y}_t=\boldsymbol{y}\right)\,. \tag{51}$$

The linear differential operator is in this case defined by

$$\mathcal{L}_{\boldsymbol{x},\boldsymbol{c}}\cdot=\partial_t\cdot-\alpha\cdot+(M\boldsymbol{y}+\boldsymbol{c})^\top\nabla_{\boldsymbol{y}}\cdot+\frac{1}{2}\mathrm{tr}\left(\Sigma\,\Sigma^\top\mathcal{H}_{\boldsymbol{y}}\cdot\right)\,. \tag{52}$$

The inner and terminal domains are respectively $\mathring{\mathcal{X}}=[0,T)\times\mathbb{R}^{(p-1)}$ and $\bar{\mathcal{X}}=T\times\mathbb{R}^{(p-1)}$. Noting $\boldsymbol{x}=(t,\boldsymbol{y})$, the value function, $V(\boldsymbol{x})=\max_{c\in\mathcal{A}}J(\boldsymbol{x},\boldsymbol{c})$, is the solution of the following

HJB equation:

$$\sup_{\boldsymbol{c}\in A}\Big(-\boldsymbol{c}^\top Q_1\,\boldsymbol{c}+\mathcal{L}_{\mathring{\boldsymbol{x}},\boldsymbol{c}}V(\mathring{\boldsymbol{x}})\Big)=0\quad\forall\mathring{\boldsymbol{x}}\in\mathring{\mathcal{X}}\,, \tag{53}$$

$$V(\bar{\boldsymbol{x}})=\bar{\boldsymbol{y}}^\top Q_2\,\bar{\boldsymbol{y}}\quad\forall\bar{\boldsymbol{x}}\in\bar{\mathcal{X}}\,. \tag{54}$$

By deriving Eq. (53) w.r.t. $\boldsymbol{c}$, we infer that the optimal control, $\boldsymbol{c}^*(t,\boldsymbol{y})$, is proportional to the gradient of the value function:

$$\boldsymbol{c}^*(t,\boldsymbol{y}) \;=\; \frac{1}{2}Q_1^{-1}\nabla_{\boldsymbol{y}}V(\boldsymbol{x})\,. \tag{55}$$

Furthermore, the value function admits a closed-form solution developed in the next proposition.

Proposition 6. *$V(t,\boldsymbol{y})=\max_{c\in\mathcal{A}}J(t,\boldsymbol{y},\boldsymbol{c})$ is a quadratic function of risk processes*

$$V(t,\boldsymbol{y}) \;=\; \boldsymbol{y}^\top A(t)\boldsymbol{y}+\boldsymbol{y}^\top B(t)+C(t)\,, \tag{56}$$

where $A(t):[0,T]\to\mathbb{R}^{(p-1)\times(p-1)}$, $B(t):[0,T]\to\mathbb{R}^{(p-1)}$, $C(t):[0,T]\to\mathbb{R}$ are the solutions of a system of differential equations:

$$\begin{aligned}
A(t)' &= -\Big(2M^\top-\alpha I_{p-1}\Big)\,A(t)-A(t)^\top Q_1^{-1\,\top}A(t)\,,\\
B(t)' &= -\Big(M^\top-\alpha I_{p-1}\Big)\,B(t)\,,\\
C(t)' &= \alpha C(t)-tr\Big(\Sigma\,\Sigma^\top A(t)\Big)-\frac{1}{4}B(t)^\top Q_1^{-1\,\top}B(t)\,,
\end{aligned} \tag{57}$$

with the terminal conditions $A(T)=Q_2$, $B(T)=0$ and $C(T)=0$.

To prove this result, it is sufficient to check that the value function (56) satisfies the HJB equation with the optimal control (55). The functions $A(t)$, $B(t)$ and $C(t)$ are computed by solving numerically the ODEs (57).

We use Proposition 6 to measure the accuracy of the approximation yielded by the PI-CGPR algorithm 1. For the linear-quadratic regulator problem, the step 4.b) of the algorithm becomes:

$$\boldsymbol{c}^{(n)}(t,\boldsymbol{y}) \;=\; \frac{1}{2}Q_1^{-1}\nabla_{\boldsymbol{y}}\widehat{V^{(n-1)}}(\boldsymbol{x})\,, \tag{58}$$

where $\widehat{V^{(n-1)}}(\boldsymbol{x})$ is the CGPR approximation of the value function.

Table 8 presents the parameters driving the risk processes of a two-dimensional linear-quadratic regulator. We consider maturities ranging from 2 up to 8 years. As for the consumption-investment problem, we set σ^* to 10^{-4}. Considering a lower value prevents the PI-CGPR algorithm from converging due to an ill-conditioned Gram matrix, $C^{(n)}\left(\mathring{\boldsymbol{X}},\bar{\boldsymbol{X}}\right)$. Other tuning parameters of the algorithm are provided in Table 9.

$M=\begin{pmatrix}0.06 & 0.00\\ 0.00 & 0.03\end{pmatrix}$	$\Sigma=\begin{pmatrix}4.00 & -1.00\\ -0.25 & 3.00\end{pmatrix}$
$Q_1=\begin{pmatrix}1.5 & 0.0\\ 0.0 & 1.0\end{pmatrix}$	$Q_2=\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{pmatrix}$
$\alpha=5\%$	
T from 2 to 8 years	

Table 8: Parameters defining the 2 risk processes of the linear-quadratic regulator problem.

$\mathring{t}$	uniform distribution $[0, T)$
$\mathring{y}^{(1)}, \mathring{y}^{(2)}$	uniform distribution $[0, 400)$
σ^*	10^{-4}
$\eta_0^{(0)}$	0.00
$\eta_t^{(0)}$	1.5×standard deviation of $\mathring{t}$, $\sqrt{T^2/12}$
$\eta_1^{(0)}$, $\eta_2^{(0)}$	1.5×standard deviations of $\mathring{y}^{(1)}$and $\mathring{y}^{(2)}$, $\sqrt{400^2/12}$
$c_t^{(1)(0)}$, $c_t^{(2)(0)}$	3.00

Table 9: Kernel and PI-CGPR parameters for the linear-quadratic regulator.

Table 10 reports the MREs, as defined in Equation (50), for the sample sets $\mathring{\boldsymbol{X}}$ of sizes ranging from 400 to 600. The exact value function is computed by solving numerically Equations (57) with backward finite differences. We observe that MREs rise with the time-horizon of the problem, while increasing the size of the sample set reduces the MREs. We will check later that the largest spreads between approximated and exact value functions are obtained with triplets close to the boundaries of the training set. The policy improvement algorithm needs more iterations to converge[5] than for the consumption-investment problem. The computational times, reported in Table 11, vary from 192s to 592s. We run 10 iterations of the gradient descent for optimizing hyper-parameters. The learning rate is $\rho = 0.01$ whereas the gradient with respect to $\boldsymbol{\eta}$ in Equation (41) is computed by finite differences with increments of $\boldsymbol{\eta}$ equal to 0.20 .

	m and $d = m/2$					
	400	500	600	400	500	600
T	MRE (%)			# of iterations		
2.0	0.04	0.02	0.02	3	3	3
4.0	0.27	0.23	0.16	5	5	5
6.0	1.09	1.27	0.83	7	6	7
8.0	5.12	4.93	3.92	9	10	9

Table 10: Mean relative errors (MREs) in % and number of iterations for converging.

	m and $d = m/2$		
	400	500	600
T	Time (seconds)		
2.0	191.92	250.73	336.33
4.0	268.35	339.75	444.56
6.0	344.17	426.01	559.63
8.0	382.46	475.47	591.85

Table 11: Computational time.

Table 12 shows the errors $\mathring{Err}$ and $\bar{Err}$, defined in Equations (29) and (30), on the inner and boundary domains. The errors increase with the time horizon and slightly decrease with the size of the sample sets.

[5]We use $\delta = 1.00$ as stopping criterion.

	m and $d = m/2$					
	400	500	600	400	500	600
T	$\mathring{Err}$			$\bar{Err}$		
2.0	0.0494	0.0458	0.0394	0.0375	0.0381	0.0334
4.0	0.0803	0.0788	0.0732	0.0401	0.0416	0.0372
6.0	0.1842	0.1801	0.1761	0.0711	0.0745	0.0826
8.0	0.5825	0.5682	0.5365	0.2363	0.1765	0.1899

Table 12: Errors of approximation on the inner and boundary domains.

Table 13 provides the hyper-parameters of the Gaussian kernel , obtained with gradient descent. We see that η_t increases with the time horizon T. Notice that we have imposed the constraint $\eta_1 = \eta_2$, to keep under control the computation time. The hyper-parameters are relatively stable in the different settings. η_0 is on average close to 2.00.

	η_1, η_2				η_t		
T/m	400	500	600	T/m	400	500	600
2.0	175.2519	175.2223	175.2165	2.0	2.9128	2.8834	2.8769
4.0	175.2248	175.2252	175.2093	4.0	3.7507	3.7522	3.7362
6.0	175.2176	174.4026	175.1943	6.0	4.6100	3.7955	4.5874
8.0	174.9879	174.5850	174.4005	8.0	5.2461	4.8441	4.6595

	η_0		
T/m	400	500	600
2.0	2.0587	2.0274	2.0202
4.0	2.0375	2.0357	2.0136
6.0	2.0453	1.2088	2.0069
8.0	1.8276	1.4087	1.2065

Table 13: Hyper-parameters obtained after 10 iterations of gradient descent.

To end this section, we focus on the optimal regulator problem with $T = 5$ and m=500. Table 14 presents the evolution of inner and boundary errors (Equations (29); (30)) for each PI iteration. We see that the PI algorithm converges and that errors decrease at each iteration.

	$\mathring{Err}$	$\bar{Err}$	$\frac{1}{m}\sum_{i=1}^{m}\left(\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}_i) - \widehat{V^{(n-1)}}(\mathring{\boldsymbol{x}}_i)\right)^2$
1	827.2925	0.0420	206401741.17
2	99.1432	0.0500	20423585.12
3	1.7486	0.0497	154424.51
4	0.1294	0.0496	593.59
5	0.1147	0.0476	36.77
6	0.1142	0.0500	2.47
7	0.1147	0.0509	0.20

Table 14: Evolution of the average inner and boundary errors by PI iterations.

Figure 5 presents the absolute spreads between the exact and approximated value functions, $\widehat{V^{(n)}}(\mathring{\boldsymbol{x}}) - V(\mathring{\boldsymbol{x}})$, on $[0,5] \times [50,250]^2$. We again observe local maxima, located in areas far from sampled points in $\mathring{\boldsymbol{X}}$. The largest errors are observed around the starting time and for the largest values of $y_0^{(1)}$ and $y_0^{(2)}$. As with the consumption-investment problem, the largest errors are found on the boundaries of the training domain.

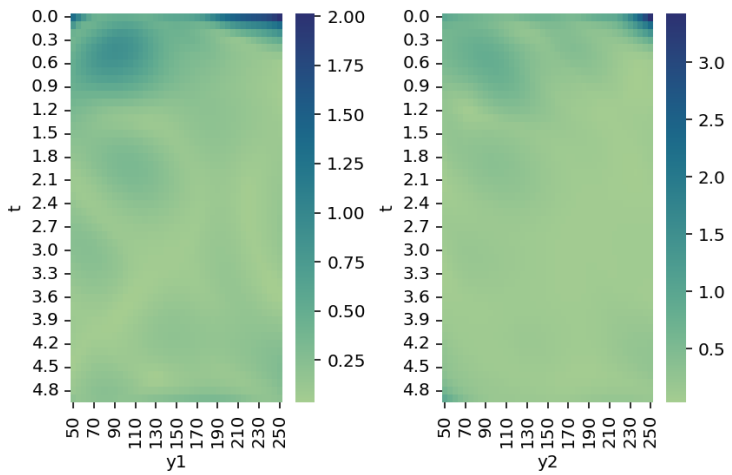

Figure 5: Inner errors for $T = 5$ years and m =500.

Figure 6 compares approximated and exact value functions and controls in a single scenario. The value function and controls are calculated with the assumption that $y_t^{(1)}$,$y_t^{(2)}$ remain equal to 100. The graphs clearly underline the accuracy of the PI-CGPR algorithm, at least in the central areas of the learning domain. This is confirmed by the heatmaps 7, of relative errors between exact and approximated value functions, on the inner subset $[0, 5) \times [50, 250]^2$. In this region of the domain, the MREs are nearly everywhere smaller than 1.00%, except for short-term maturities and the largest values of state variables. The same conclusions apply to the heatmaps 8, which show the relative spreads between approximated and exact controls. In practice, we therefore need to consider a sufficiently large training domain compared to the domain in which we employ the approximation. We can also consider Sobol sampling (1967) or non-uniform distributions for sampling more state variables in critical regions of the domain, as in Bachouch et al. (2022).

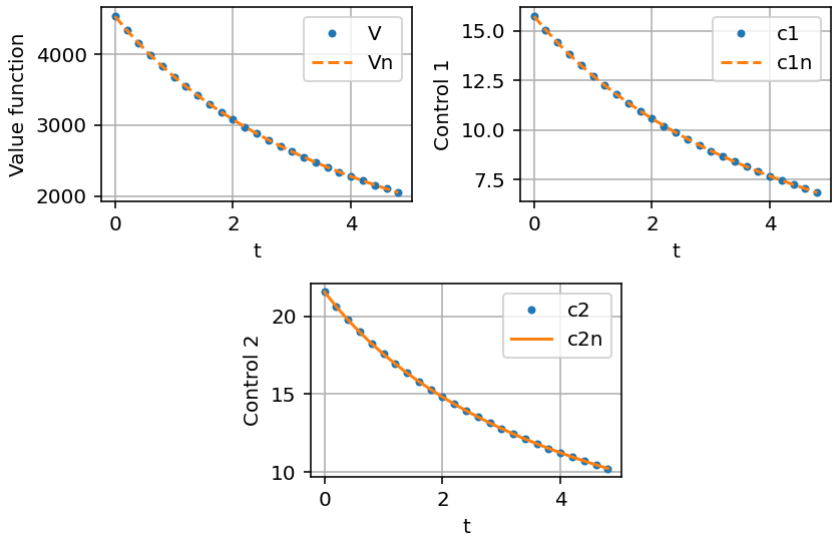

Figure 6: Comparison of exact and approximated value functions and controls in one sample path: $y_t^{(1)}, y_t^{(2)} = 100$ and $T = 5$ years.

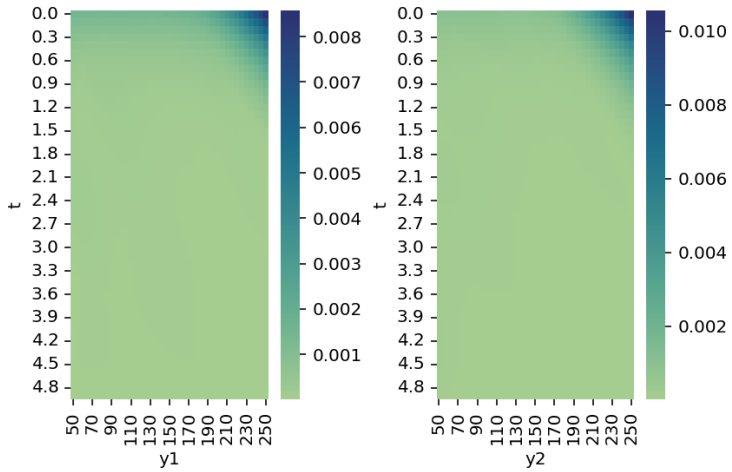

Figure 7: Heatmaps of $\left|\frac{\widehat{V^{(n)}}(t,y)-V(t,y)}{V(t,y)}\right|$, time versus $y^{(1)}$ and time versus $y^{(2)}$.

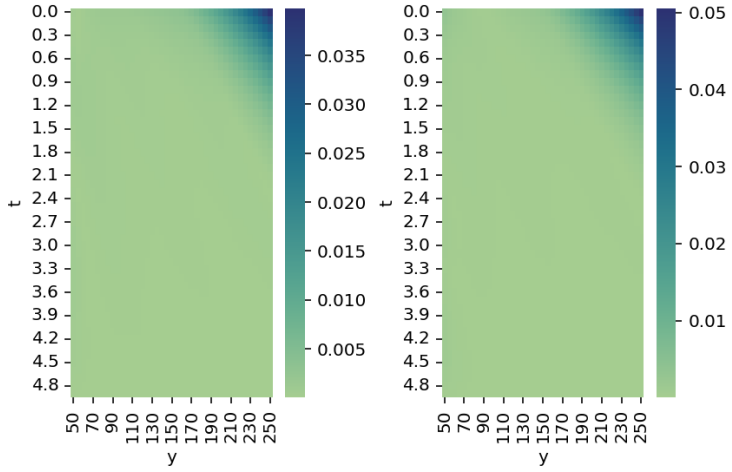

Figure 8: Left plot: heatmap of $\left|\frac{c_t^{(1)(n)}-c_t^{(1)*}}{c_t^{(1)*}}\right|$. Right plot: heatmap of $\left|\frac{c_t^{(2)(n)}-c_t^{(2)*}}{c_t^{(2)*}}\right|$.

6 Conclusions

This paper presents a novel iterative algorithm that combines policy improvements with constrained Gaussian process regressions (CGPR) to solve stochastic control problems. By leveraging the strengths of Gaussian processes for regression and the iterative nature of policy improvements, our method offers a robust and efficient approach to approximating value functions and optimizing controls.

Our numerical experiments, focusing on the consumption-investment and linear-quadratic regulator problems, demonstrate the efficacy and accuracy of the proposed PI-CGPR algorithm. The results indicate that the algorithm consistently improves the value function at each iteration and converges to the optimal solution within a reasonable number of iterations. The analytical tractability of Gaussian kernels further enhances the computational efficiency of our method.

However, the PI-CGPR algorithm is not without its limitations. One significant drawback

is its reliance on the choice of kernel and hyper-parameters, which can impact the accuracy and stability of the solution. The method also requires a careful balance between the size of the training set and the regularization parameter to avoid numerical instabilities. Additionally, the accuracy of the approximation tends to decline near the boundaries of the training domain.

Despite these limitations, the PI-CGPR algorithm's ability to solve control problems makes it a valuable tool for a wide range of applications in finance, engineering, and beyond. Future research could explore the extension of this framework to other types of kernels and the incorporation of additional constraints on controls to further enhance its applicability and performance.

Appendix A, partial derivatives of $\beta^{(n)}(\boldsymbol{x})$

In the policy improvements algorithm 1, we consider optimization problems which admit explicit optimal controls in terms of the gradient and Hessian of intermediates approximated payoffs:

$$\boldsymbol{c}^{(n+1)} = \boldsymbol{c}^{(n)}\left(\boldsymbol{x}, \partial_t \widehat{V^{(n-1)}}(\boldsymbol{x}), \nabla_{\boldsymbol{y}} \widehat{V^{(n)}}(\boldsymbol{x}), \mathcal{H}_{\boldsymbol{y}} \widehat{V^{(n)}}(\boldsymbol{x})\right),$$

where $\widehat{V^{(n)}}(\boldsymbol{x}) = \boldsymbol{\beta}^{(n)}(\boldsymbol{x})^\top C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix} \boldsymbol{z}^{(n)} \\ \boldsymbol{h} \end{pmatrix}$. The first and second order derivatives of $\widehat{V^{(n)}}$ ruling the control are

$$\partial_t \widehat{V^{(n)}}(\boldsymbol{x}) = \left(\partial_t \boldsymbol{\beta}^{(n)}(\boldsymbol{x})\right)^\top C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix} \boldsymbol{z}^{(n)} \\ \boldsymbol{h} \end{pmatrix}, \tag{59}$$

$$\partial_{y_k} \widehat{V^{(n)}}(\boldsymbol{x}) = \left(\partial_{y_k} \boldsymbol{\beta}^{(n)}(\boldsymbol{x})\right)^\top C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix} \boldsymbol{z}^{(n)} \\ \boldsymbol{h} \end{pmatrix}, \tag{60}$$

$$\partial_{y_l}\partial_{y_k} \widehat{V^{(n)}}(\boldsymbol{x}) = \left(\partial_{y_l}\partial_{y_k} \boldsymbol{\beta}^{(n)}(\boldsymbol{x})\right)^\top C^{(n)}\left(\mathring{\boldsymbol{X}}, \bar{\boldsymbol{X}}\right)^{-1}\begin{pmatrix} \boldsymbol{z}^{(n)} \\ \boldsymbol{h} \end{pmatrix}. \tag{61}$$

The derivative of $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ w.r.t. time is a $(m+d)$ vector

$$\partial_t \boldsymbol{\beta}^{(n)}(\boldsymbol{x}) = \begin{pmatrix} \left(\partial_t \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k\left(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i\right)\right)_{i=1,\dots,m} \\ \left(\partial_t k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right)\right)_{j=1,\dots,d} \end{pmatrix},$$

where

$$\begin{cases} \partial_t \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k\left(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i\right) & = \frac{k(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i)}{\eta_t^2} - \frac{\left(t - \mathring{t}_i\right)}{\eta_t^2} \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k\left(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i\right) \\ \partial_t k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right) & = \frac{\left(\bar{t}_j - t\right)}{\eta_t^2} k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right). \end{cases}$$

The derivative of $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ w.r.t. y_k in Eq. (60) is a $(m+d)$ vector

$$\partial_{y_k} \boldsymbol{\beta}^{(n)}(\boldsymbol{x}) = \begin{pmatrix} \left(\partial_{y_k} \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k\left(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i\right)\right)_{i=1,\dots,m} \\ \left(\partial_{y_k} k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right)\right)_{j=1,\dots,d} \end{pmatrix},$$

with elements equal to

$$\begin{cases} \partial_{y_k} \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k\left(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i\right) & = \left[\boldsymbol{e}_k^\top \Sigma_{\boldsymbol{Y}}(\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}) \Sigma_{\boldsymbol{Y}}(\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)})^\top \boldsymbol{d}_{\mathring{\boldsymbol{y}}_i}(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i)\right. \\ & \quad \left. + \boldsymbol{\mu}_{\boldsymbol{Y}}(\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)})^\top \boldsymbol{e}_k\right] \frac{k(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i)}{\eta_k^2} \\ & \quad - \mathcal{L}_{\mathring{\boldsymbol{x}}_i, \boldsymbol{c}^{(n)}} k\left(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i\right) \boldsymbol{e}_k^\top \boldsymbol{d}_{\mathring{\boldsymbol{y}}_i}(\boldsymbol{x}, \mathring{\boldsymbol{x}}_i), \\ \partial_{y_k} k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right) & = \frac{\left(\bar{y}_{j,k} - y_k\right)}{\eta_k^2} k\left(\bar{\boldsymbol{x}}_j, \boldsymbol{x}\right). \end{cases}$$

The Hessian of $\boldsymbol{\beta}^{(n)}(\boldsymbol{x})$ is the $(p-1)\times(p-1)$ matrix of second order derivatives

$$\partial_{y_l}\partial_{y_k}\boldsymbol{\beta}^{(n)}(\boldsymbol{x}) \;=\; \begin{pmatrix} \left(\partial_{y_l}\partial_{y_k}\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\mathring{\boldsymbol{x}}_i\right)\right)_{i=1,\dots,m} \\ \left(\partial_{y_l}\partial_{y_k}k\left(\bar{\boldsymbol{x}}_j,\boldsymbol{x}\right)\right)_{j=1,\dots,d} \end{pmatrix},$$

where

$$\begin{cases} \partial_{y_l}\partial_{y_k}\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\mathring{\boldsymbol{x}}_i\right) & = \left(\boldsymbol{e}_k^\top\Sigma_{\boldsymbol{Y}}(\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)})\Sigma_{\boldsymbol{Y}}(\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)})^\top\boldsymbol{e}_l\right)\frac{k(\boldsymbol{x},\mathring{\boldsymbol{x}}_i)}{\eta_k^2\eta_l^2} \\ & \quad -\left(\partial_{y_k}\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\mathring{\boldsymbol{x}}_i\right)\right)\boldsymbol{e}_l^\top\boldsymbol{d}_{\mathring{\boldsymbol{y}}_i}(\boldsymbol{x},\mathring{\boldsymbol{x}}_i) \\ & \quad -\left(\partial_{y_l}\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\mathring{\boldsymbol{x}}_i\right)\right)\boldsymbol{e}_k^\top\boldsymbol{d}_{\mathring{\boldsymbol{y}}_i}(\boldsymbol{x},\mathring{\boldsymbol{x}}_i) \\ & \quad -\mathcal{L}_{\mathring{\boldsymbol{x}}_i,\boldsymbol{c}^{(n)}}k\left(\boldsymbol{x},\mathring{\boldsymbol{x}}_i\right)\left(\boldsymbol{e}_l^\top\boldsymbol{H}_{\boldsymbol{y}}(\boldsymbol{x},\mathring{\boldsymbol{x}}_i)\boldsymbol{e}_k\right), \\ \partial_{y_l}\partial_{y_k}k\left(\bar{\boldsymbol{x}}_j,\boldsymbol{x}\right) & = \left(\frac{(\bar{y}_{j,k}-y_k)(\bar{y}_{j,l}-y_l)}{\eta_k^2\eta_l^2}-\mathbf{1}_{\{k=l\}}\frac{1}{\eta_k^2}\right)k\left(\bar{\boldsymbol{x}}_j,\boldsymbol{x}\right). \end{cases}$$

Acknowledgement

Donatien Hainaut thanks the FNRS (Fonds de la recherche scientifique) for the financial support through the EOS project Asterisk (research grant 40007517).

Statements and declarations

Authors have seen and agree with the contents of the manuscript and there is no financial interest to report.

References

[1] Al-Aradi A., Correia A., Jardim G., de Freitas Naiff D., Saporito Y. 2022. Extensions of the deep Galerkin method. App. Math. and Computation, 430, 127287.

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

[3] Banerjee A., Dunson D.B., Tokdar S.T., 2013. Efficient Gaussian process regression for large datasets. Biometrika 100 (1), 75–89.

[4] Bertsekas D.P., 2011. Approximate policy iteration: a survey and some new methods. Journal of Control Theory and Technology, 9, 310-335.

[5] Choi J., Son Y. , Jeong M.K. 2024. Gaussian kernel with correlated variables for incomplete data. Annals of Operational Research 341, 223–244.

[6] Deringer V., Bartok A., Bernstein N., Wilkins D., Ceriotti M., 2021. Gaussian Process Regression for Materials and Molecules. Chemical Reviews 121 (16), 10073-10141.

[7] De Spiegeleer J., Madan D.B., Reyners S., Schoutens W. 2018. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. Quantitative finance, 18 (10), 1635–1643.

[8] Dupret J.L., Hainaut D., 2024. Deep learning for high-dimensional continuous-time stochastic optimal control without explicit solution. UCLouvain working paper. https://dial.uclouvain.be/pr/boreal/object/boreal:287848.

[9] Fine C.H., Porteus, E.L. 1989. Dynamic process improvement. Operations research 37 (4), 514-673.

[10] Glau K., Wunderlich L. 2022. The deep parametric PDE method and applications to option pricing. App. Math. and Computation, 432, 127355

[11] Glau K., Wunderlich L. 2024. Neural network expression rates and applications of the deep parametric PDE method in counterparty credit risk. Annals of Operations Research 336, 331–357.

[12] Hainaut D., Casas A., 2024.Option pricing in the Heston model with physics inspired neural networks. Annals of finance. https://doi.org/10.1007/s10436-024-00452-7

[13] Hainaut D., Vrins F., 2024. European option pricing with model constrained Gaussian process regressions. UCLouvain ISBA discussion paper. https://dial.uclouvain.be/pr/boreal/object/boreal:292395

[14] Hainaut D. 2024. American option pricing with model constrained Gaussian process regressions. UCLouvain ISBA discussion paper. https://dial.uclouvain.be/pr/boreal/object/boreal:292665

[15] Howard R.A. 1960. Dynamic programming and Markov processes. MIT press and J. Wiley & Sons, New-York-London.

[16] Jacka S.D., Mijatovic A. 2017. On the policy improvement algorithm in continuous time. Stochastics 89(1), 348–359.

[17] Murphy K.P. 2012. Machine learning: a probabilistic perspective, MIT press.

[18] Price I., Fowkes J., Hopman D. 2019. Gaussian processes for unconstrained demand. European Journal of Operational Research, 275 (2), 621-634.

[19] Rasmussen C.E., Williams C.K.I. 2006. Gaussian processes for machine learning, MIT press.

[20] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686–707.

[21] Roberts S, Osborne M, Ebden M, Reece S, Gibson N, Aigrain S., 2013. Gaussian processes for time-series modelling. Phil Trans R Soc A 371: 20110550.

[22] Sacks J., Welch W.J., Mitchell T.J., Wynn. H.P. 1989. Design and analysis of computer experiments. Statistical Science, 409–423.

[23] Santner T.J., Williams B.J., Notz W.I. 2003. The design and analysis of computer experiments. Springer Series in Statistics.

[24] Sobol, I.M. 1967. Distribution of points in a cube and approximate evaluation of integrals. Zh. Vych. Mat. Mat. Fiz. 7: 784–802 (in Russian); U.S.S.R Comput. Maths. Math. Phys. 7: 86–112.

[25] Sirignano J., Spiliopoulos K., 2018. DGM: a deep learning algorithm for solving partial differential equations, Journal of Computational Physics, 375, 1339–1364.

[26] Swiler L., Gulian M., Frankel A., Safta C., Jakeman J. 2020. A survey of constrained Gaussian process regression: approaches and implementation challenges. Journal of Machine Learning for Modeling and Computing 1(2) 119-156.

[27] Touzi, N. (2012). Optimal stochastic control, stochastic target problems, and backward SDE, volume 29. Springer Science & Business Media.