FiNANCE FOR ENERGY MARKET RESEARCH CENTRE

Deep neural networks algorithms for stochastic control problems on finite horizon, part II : numerical applications

Achref BACHOUCH, Côme HURE, Nicolas LANGRENE and Huyên PHAM

Working Paper
RR-FiME-18-11

December 2018

DAUPHINE UNIVERSITÉ PARIS

CREST Centre de Recherche en Économie et Statistique

ÉCOLE POLYTECHNIQUE

EDF

Deep neural networks algorithms for stochastic control problems on finite horizon, Part 2: numerical applications

Achref Bachouch * Côme Huré † Nicolas Langrené ‡ Huyên Pham §

December 9, 2018

Abstract

This paper presents several numerical applications of deep learning-based algorithms that have been analyzed in [11]. Numerical and comparative tests using TensorFlow illustrate the performance of our different algorithms, namely control learning by performance iteration (algorithms NNcontPI and ClassifPI), control learning by hybrid iteration (algorithms Hybrid-Now and Hybrid-LaterQ), on the 100-dimensional nonlinear PDEs examples from [6] and on quadratic Backward Stochastic Differential equations as in [5]. We also provide numerical results for an option hedging problem in finance, and energy storage problems arising in the valuation of gas storage and in microgrid management.

1 Introduction

This paper is devoted to the numerical resolution of discrete-time stochastic control problem over a finite horizon. The dynamics of the controlled state process $X = (X_n)_n$ valued in $\mathbb{R}^d$ is given by

$$X_{n+1} \;=\; F(X_n, \alpha_n, \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \; X_0 = x_0 \in \mathbb{R}^d, \tag{1.1}$$

where $(\varepsilon_n)_n$ is a sequence of i.i.d. random variables valued in some Borel space $(E, \mathcal{B}(E))$, and defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ equipped with the filtration $\mathbb{F} = (\mathcal{F}_n)_n$ generated by the noise $(\varepsilon_n)_n$ ($\mathcal{F}_0$ is the trivial σ-algebra), the control $\alpha = (\alpha_n)_n$ is an $\mathbb{F}$-adapted process valued in $\mathbb{A} \subset \mathbb{R}^q$, and F is a measurable function from $\mathbb{R}^d \times \mathbb{R}^q \times E$ into $\mathbb{R}^d$. Given a running cost function f defined on $\mathbb{R}^d \times \mathbb{R}^q$ and a terminal cost function g defined on $\mathbb{R}^d$, the cost functional associated with a control process α is

$$J(\alpha) \;=\; \mathbb{E}\Big[\sum_{n=0}^{N-1} f(X_n, \alpha_n) + g(X_N) \Big]. \tag{1.2}$$

*Department of Mathematics, University of Oslo, Norway. The author's research is carried out with support of the Norwegian Research Council, within the research project Challenges in Stochastic Control, Information and Applications (STOCONINF), project number 250768/F20 achrefb at math.uio.no

†LPSM, University Paris Diderot hure at lpsm.paris

‡CSIRO, Data61, RiskLab Australia Nicolas.Langrene at data61.csiro.au

§LPSM, University Paris-Diderot and CREST-ENSAE, pham at lspm.paris The work of this author is supported by the ANR project CAESARS (ANR-15-CE05-0024), and also by FiME and the "Finance and Sustainable Development" EDF - CACIB Chair

The set $\mathcal{A}$ of admissible controls is the set of control processes α satisfying some integrability conditions ensuring that the cost functional $J(\alpha)$ is well-defined and finite. The control problem, also called Markov decision process (MDP), is formulated as

$$V_0(x_0) \;:=\; \inf_{\alpha\in\mathcal{A}} J(\alpha), \tag{1.3}$$

and the goal is to find an optimal control $\alpha^* \in \mathcal{A}$, i.e., attaining the optimal value: $V_0(x_0) = J(\alpha^*)$. Notice that problem (1.1)-(1.3) may also be viewed as the time discretization of a continuous time stochastic control problem, in which case, F is typically the Euler scheme for a controlled diffusion process.

It is well-known that the global dynamic optimization problem (1.3) can be reduced to local optimization problems via the dynamic programming (DP) approach, which allows to determine the value function in a backward recursion by

$$\begin{array}{rcl} V_N(x) &=& g(x), \quad x \in \mathbb{R}^d, \\ V_n(x) &=& \inf\limits_{a\in\mathbb{A}} Q_n(x,a), \\ \text{with } \; Q_n(x,a) &=& f(x,a) + \mathbb{E}\big[V_{n+1}(X_{n+1})\big|X_n = x, \alpha_n = a\big], \quad (x,a) \in \mathbb{R}^d \times \mathbb{A}. \end{array} \tag{1.4}$$

Moreover, when the infimum is attained in the DP formula (1.4) at any time n by $a_n^*(x) \in \arg\min_{a\in\mathbb{A}} Q_n(x,a)$, we get an optimal control in feedback form (policy) given by: $\alpha^* = (a_n^*(X_n^*))_n$ where X^* is the Markov process defined by

$$X_{n+1}^* \;=\; F(X_n^*, a_n^*(X_n^*), \varepsilon_{n+1}), \quad n = 0, \ldots, N-1, \;\; X_0^* = x_0.$$

The practical implementation of the DP formula may suffer from the curse of dimensionality and large complexity when the state space dimension d and the control space dimension are high. In [11], we proposed algorithms relying on deep neural networks for approximating/learning the optimal policy and then eventually the value function by performance/policy iteration or hybrid iteration with Monte Carlo regressions now or later. This research led to three algorithms, namely algorithms NNcontPI, Hybrid-Now and Hybrid-LaterQ that are recalled in Section 2. In Section 3, we perform some numerical and comparative tests for illustrating the efficiency of our different algorithms, on 100-dimensional nonlinear PDEs examples as in [6] and quadratic Backward Stochastic Differential equations as in [5]. We also provide numerical results for an option hedging problem in finance, and energy storage problems arising in the valuation of gas storage and in microgrid management. Finally, we conclude in Section 4 with some comments about possible extensions and improvements of our algorithms.

Remark 1.1 The proposed algorithms can deal with state and control constraints at any time, which is useful in several applications:

$$(X_n^\alpha, \alpha_n) \;\in\; \mathcal{S} \;\; a.s., \;\; n \in \mathbb{N},$$

where $\mathcal{S}$ is some given subset of $\mathbb{R}^d \times \mathbb{R}^q$. In this case, in order to ensure that the set of admissible controls is not empty, we assume that the sets

$$\mathbb{A}(x) \;:=\; \Big\{a \in \mathbb{R}^q : (F(x,a,\varepsilon_1), a) \in \mathcal{S} \; a.s.\Big\}$$

are non empty for all $x \in \mathcal{S}$, and the DP formula now reads

$$V_n(x) \;=\; \inf_{a\in\mathbb{A}(x)} \big[f(x,a) + P^a V_{n+1}(x)\big], \quad x \in \mathcal{S}.$$

From a computational point of view, it may be more convenient to work with unconstrained state/control variables, hence by relaxing the state/control constraint and introducing into the running cost a penalty function $L(x,a)$: $f(x,a) \leftarrow f(x,a) + L(x,a)$, and $g(x) \leftarrow g(x) + L(x,a)$. For example, if the constraint set $\mathcal{S}$ is in the form: $\mathcal{S} = \{(x,a) \in \mathbb{R}^d \times \mathbb{R}^q : h_k(x,a) = 0, k = 1,\ldots,p,\ h_k(x,a) \geq 0, k = p+1,\ldots,q\}$, for some functions h_k, then one can take as penalty functions:

$$L(x,a) \;=\; \sum_{k=1}^{p} \mu_k |h_k(x,a)|^2 + \sum_{k=p+1}^{q} \mu_k \max(0, -h_k(x,a)).$$

where $\mu_k > 0$ are penalization coefficients (large in practice). □

2 Algorithms

This section recalls the DNN-based algorithms we propose to solve the discrete-time stochastic control problem (1.1)-(1.2)-(1.3)-(1.4). These algorithms have been described and analyzed in detail in our companion paper [11]. We also introduce a quantization and k-nearest-neighbor-based algorithm (knn) to be used as benchmark when testing our algorithms on low-dimensional control problems.

We are given a class of deep neural networks (DNN) for the control policy represented by the parametric functions $x \in \mathbb{R}^d \mapsto A(x;\beta) \in \mathbb{A}$, with parameters $\beta \in \mathbb{R}^q$, and a class of DNN for the value function represented by the parametric functions: $x \in \mathbb{R}^d \mapsto \Phi(x;\theta) \in \mathbb{R}$, with parameters $\theta \in \mathbb{R}^p$. Recall that these DNN functions A and Φ are compositions of linear combinations and nonlinear activation functions, see [8].

2.1 Control Learning by Performance Iteration (NNContPI & ClassifPI)

- For $n = N-1,\ldots,0$, keep track of the approximated optimal policies $\hat{a}_k$, $k = n+1,\ldots,N-1$, and compute the approximated optimal policy at time n by

$$\begin{aligned} \hat{a}_n \;&=\; A(.;\hat{\beta}_n) \quad \text{with} \\ \hat{\beta}_n \;&\in\; \operatorname*{argmin}_{\beta\in\mathbb{R}^q} \mathbb{E}\big[f\big(X_n, A(X_n;\beta)\big) + \sum_{k=n+1}^{N-1} f\big(X_k^\beta, \hat{a}_k\big(X_k^\beta\big)\big) + g\big(X_N^\beta\big)\big] \end{aligned} \tag{2.1}$$

  where X_n is distributed according to a training probability measure μ on $\mathbb{R}^d$, and with $\big(X_k^\beta\big)_{k=n+1}^N$ defined by induction, for $m = 1,\ldots,M$, as:

$$\begin{cases} X_{n+1}^\beta \;=\; F\big(X_n, A\big(X_n;\beta\big), \varepsilon_{n+1}\big) \\ X_{k+1}^\beta \;=\; F\big(X_k^\beta, \hat{a}_k\big(X_k^\beta;\beta\big), \varepsilon_{k+1}\big), \quad \text{for } k = n+1,\ldots,N-1. \end{cases}$$

  We later refer to this algorithm as the NNContPI algorithm.

Remark 2.1 In practice, we use the Adam algorithm, implemented in TensorFlow, to compute $\hat{\beta}_n$ in (2.1), and refer to section 3.3 of [11] for a discussion on the choice of the training measure. □

Remark 2.2 (Case of finite control space) In the case where the control space $\mathbb{A}$ is finite, i.e., $\mathrm{Card}(\mathbb{A}) = L < \infty$ with $\mathbb{A} = \{a_1, \ldots, a_L\}$, a classification method can be used: consider a DNN that takes state x as input and returns a probability vector $p(x;\beta) = (p_\ell(x;\beta))_{\ell=1}^L$ with parameters β; the algorithm reads:

- For $n = N-1, \ldots, 0$, keep track of the approximated optimal policies $\hat{a}_k$, $k = n+1, \ldots, N-1$, and compute the approximated optimal policy at time n by

$$\begin{aligned}\hat{a}_n(x) &= a_{\hat{\ell}_n(x)} \text{ with } \hat{\ell}_n(x) \in \arg\max_{\ell=1,\ldots,L} p_\ell(x;\hat{\beta}_n)\\ \hat{\beta}_n &\in \arg\min_{\beta\in\mathbb{R}^q} \mathbb{E}\Big[\sum_{\ell=1}^{L} p_\ell(X_n;\beta)\Big(f(X_n,a_\ell) + \sum_{k=n+1}^{N-1} f\big(X_k^\ell, \hat{a}_k(X_k^\ell)\big) + g(X_N^\ell)\Big)\Big],\end{aligned}$$

  where X_n is distributed according to a training probability measure μ on $\mathbb{R}^d$, and $X_{n+1}^\ell = F(X_n, a_\ell, \varepsilon_{n+1})$, $X_{k+1}^\ell = F(X_k^\ell, \hat{a}_k(X_k^\ell), \varepsilon_{k+1})$, for $k = n+1, \ldots, N-1$, $\ell = 1, \ldots, L$.

In the numerical applications of the next section, we refer to this classification-based algorithm as the ClassifPI algorithm. □

2.2 Double DNN with Regress Now (Hybrid-Now)

- Initialize $\hat{V}_N = g$
- For $n = N-1, \ldots, 0$,

(i) compute the approximated policy at time n

$$\begin{aligned}\hat{a}_n &= A(.;\hat{\beta}_n) \text{ with}\\ \hat{\beta}_n &\in \operatorname*{argmin}_{\beta\in\mathbb{R}^q} \mathbb{E}\Big[f\big(X_n, A(X_n;\beta)\big) + \hat{V}_{n+1}(X_{n+1}^\beta)\Big]\end{aligned} \tag{2.2}$$

where X_n is distributed according to a training probability measure μ on $\mathbb{R}^d$, and $X_{n+1}^\beta = F\big(X_n, A(X_n;\beta)\big), \varepsilon_{n+1})$.

(ii) estimate the value function at time n

$$\begin{aligned}\hat{V}_n &= \Phi(.;\hat{\theta}_n) \text{ with}\\ \hat{\theta}_n &\in \operatorname*{argmin}_{\theta\in\mathbb{R}^p} \mathbb{E}\Big[f(X_n, \hat{a}_n(X_n)) + \hat{V}_{n+1}(X_{n+1}^{\hat{\beta}_n}) - \Phi(X_n;\theta)\Big]^2.\end{aligned} \tag{2.3}$$

Remark 2.3 Once again, we use the Adam algorithm, natively implemented in TensorFlow, to compute $\hat{\beta}_n$ in (2.2) and $\hat{\theta}_n$ in (2.3). Once again, we refer to section 3.3 of [11] for a discussion on the choice of the training measure. □

2.3 Double DNN with Regress Later and Quantization (Hybrid-LaterQ)

We are given in addition an L-optimal quantizer of the noise ε_n via a discrete random variable $\hat{\varepsilon}_n$ valued in a grid $\{e_1, \ldots, e_L\}$ of L points in E, and with weights $p_1, \ldots, p_L$.

- Initialize $\hat{V}_N = g$
- For $n = N-1, \ldots, 0$,

(i) compute the approximated policy at time n

$$\begin{aligned} \hat{a}_n &= A(.; \hat{\beta}_n) \quad \text{with} \\ \hat{\beta}_n &\in \operatorname*{argmin}_{\beta \in \mathbb{R}^q} \mathbb{E}\big[f\big(X_n, A(X_n;\beta)\big) + \hat{V}_{n+1}(X_{n+1}^{\beta})\big], \end{aligned} \tag{2.4}$$

where X_n is distributed according to a training probability measure μ on $\mathbb{R}^d$, and where $X_{n+1}^{\beta} = F\big(X_n, A(X_n;\beta), \varepsilon_{n+1}\big)$.

(ii) approximate the value function at time $n+1$

$$\begin{aligned} \tilde{V}_{n+1} &= \Phi(.; \hat{\theta}_{n+1}) \quad \text{with} \\ \hat{\theta}_{n+1} &\in \operatorname*{argmin}_{\theta \in \mathbb{R}^p} \mathbb{E}\Big[\hat{V}_{n+1}(X_{n+1}^{\hat{\beta}_n}) - \Phi\big(X_{n+1}^{\hat{\beta}_n};\theta\big)\Big]^2. \end{aligned} \tag{2.5}$$

(iii) estimate analytically by quantization the value function at time n:

$$\hat{V}_n(x) = f(x, \hat{a}_n(x)) + \sum_{\ell=1}^{L} p_\ell \tilde{V}_{n+1}\big(F(x, \hat{a}_n(x), e_\ell)\big).$$

Remark 2.4 • We use the Adam algorithm to compute $\hat{\beta}_n$ in (2.4) and $\hat{\theta}_{n+1}$ in (2.5).

- Observe that step (ii) is an interpolation step, which means that all kind of loss functions can be chosen to compute $\hat{\theta}_{n+1}$. In (2.5), we decide to take the $\mathbb{L}^2$-loss, mainly because of its smoothness.
- we refer to section 3.3 of [11] for a discussion on the choice of the training measure.

□

2.4 Quantization with k-nearest-neighbors (Qknn-algorithm)

We now present a simple version of the Qknn algorithm, based on the quantization and k-nearest neighbors methods, which will be the benchmark for all the low-dimensional control problems that will be considered in the next section. We refer to [2] for a detailed presentation of a more sophisticated version of this algorithm, and comparisons to other well-known algorithms on various control problems.

We are given an L-optimal quantizer of the noise ε_n via a discrete random variable $\hat{\varepsilon}_n$ valued in a grid $\{e_1, \ldots, e_L\}$ of L points in E, and with weights $p_1, \ldots, p_L$; as well as grids Γ_k, $k = 0, \ldots, N$ of points in $\mathbb{R}^d$, which are assumed to cover the region of $\mathbb{R}^d$ that is likely to be visited by the optimally driven process X at time $k = 0, \ldots, N-1$.

- Initialize $\hat{V}_N = g$
- For $n = N-1, \ldots, 0$,

(i) compute the approximated Q-value at time n

$$\hat{Q}_n(z,a) \;=\; f(z,a) + \sum_{\ell=1}^{L} p_\ell \widehat{V}_{n+1}\Big(\mathrm{Proj}_{\Gamma_{n+1}}\big(F(z,a,e_\ell)\big)\Big), \quad \forall (z,a) \in \Gamma_n \times A,$$

where $\mathrm{Proj}_{\Gamma_{n+1}}$ is the Euclidean projection over Γ_{n+1}.

(ii) compute the optimal control at time n

$$\hat{A}_n(z) \in \operatorname*{argmin}_{a \in A} \big[\hat{Q}_n(z,a)\big], \quad \forall z \in \Gamma_n,$$

using classical algorithms for optimization of deterministic functions.

(iii) estimate analytically by quantization the value function:

$$\widehat{V}_n(z) \;=\; \hat{Q}_n\big(z, \hat{A}_n(z)\big), \quad \forall z \in \Gamma_n.$$

3 Numerical applications

3.1 A semilinear PDE

We consider the following semilinear PDE with quadratic growth in the gradient:

$$\left\{ \begin{array}{rcl} \dfrac{\partial v}{\partial t} + \Delta_x v \;-\; |D_x v|^2 &=& 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d, \\ v(T,x) &=& g(x), \quad x \in \mathbb{R}^d. \end{array} \right. \tag{3.1}$$

By observing that for any $p \in \mathbb{R}^d$, $-|p|^2 = \inf_{a \in \mathbb{R}^d}[|a|^2 + 2a.p]$, the PDE (3.1) can be written as a Hamilton-Jacobi-Bellman equation

$$\left\{ \begin{array}{rcl} \dfrac{\partial v}{\partial t} + \Delta_x v \;+\; \inf\limits_{a \in \mathbb{R}^d} \big[|a|^2 + 2a.D_x v\big] &=& 0, \quad (t,x) \in [0,T) \times \mathbb{R}^d, \\ v(T,x) &=& g(x), \quad x \in \mathbb{R}^d, \end{array} \right. \tag{3.2}$$

hence associated with the stochastic control problem

$$v(t,x) \;=\; \inf_{\alpha \in \mathcal{A}} \mathbb{E}\big[\int_t^T |\alpha_s|^2 ds + g(X_T^{t,x,\alpha})\big], \tag{3.3}$$

where $X = X^{t,x,\alpha}$ is the controlled process governed by

$$dX_s \;=\; 2\alpha_s ds \;+\; \sqrt{2} dW_s, \quad t \le s \le T,\; X_t = x,$$

W is a d-dimensional Brownian motion, and the control process α is valued in $A = \mathbb{R}^d$. The time discretization (with time step $h = T/N$) of the control problem (3.3) leads to the discrete-time control problem (1.1)-(1.2)-(1.3) with

$$X_{n+1}^\alpha \;=\; X_n^\alpha + 2\alpha_n h + \sqrt{2h}\varepsilon_{n+1} \;=:\; F(X_n^\alpha, \alpha_n, \varepsilon_{n+1}), \quad n = 0, \ldots, N-1,$$

where $(\varepsilon_n)_n$ is a sequence of i.i.d. random variables of law $\mathcal{N}(0, \mathbb{I}_d)$, and a cost functional,

$$J(\alpha) \quad = \quad \mathbb{E}\Big[\sum_{n=0}^{N-1} h|\alpha_n|^2 \; + \; g(X_N^\alpha)\Big].$$

On the other hand, it is known that an explicit solution to (3.1) (or equivalently (3.2)) can be obtained via a Hopf-Cole transformation (see e.g. [5]), and is given by

$$v(t,x) \quad = \quad -\ln\Big(\mathbb{E}\Big[\exp\big(-g(x+\sqrt{2}W_{T-t})\big)\Big]\Big), \quad (t,x) \in [0,T]\times\mathbb{R}^d.$$

We choose to run tests on two different examples that have already been considered in the literature:

Test 1 Some recent numerical results have been obtained in [6] (see Section 4.3 in [6]) when $T = 1$ and $g(x) = \ln(\frac{1}{2}(1 + |x|^2))$ in dimension $d = 100$ (see Table 2 and Figure 3 in [6]). Their method is based on neural network regression to solve the BSDE representation of the control problem, and provide estimations of the value function at time 0 and state 0 for different values of a coefficient γ. We plotted the results of the Hybrid-Now algorithm in Figure 1. Hybrid-Now achieves a relative error of 0,13% in a bit less than 2hours using a 4-cores 3GHz intel Core i7 CPU, which is very close to the result found by [6], and reaches a relative error of 0,09% in a bit more than 4 hours. We want to highlight the fact that the algorithm presented in [6] only needs hundreds of seconds to provide a relative error of 0,17%, which is not comparable to the time required by Hybrid-Now to converge. However, we believe that the computation time can easily be alleviated; some ideas in that direction are discussed in section 4.

We also considered the same problem in dimension $d = 2$, for which we plotted the first component of X w.r.t. time in Figure 2, for five different paths of the Brownian motion, where for each ω, the agent follows either the naive ($\alpha = 0$) or the Hybrid-Now strategy. On can see that both strategies are very similar when the terminal time is far; but the Hybrid-Now strategy clearly forces X to get closer to 0 when the terminal time gets closer, in order to reduce the terminal cost.

Test 2 Tests of the algorithms have also been run in dimension 1 with the terminal cost $g(x) = -x^\gamma \mathbb{1}_{0\leq x\leq 1} - \mathbb{1}_{1\leq x}$ and $\gamma \in (0,1)$. This problem has already been considered in [13], where the author used the BSDE-based algorithm presented in [14]. Their results for the value function estimation at time 0 and state 0, when $\gamma = 1, 0.5, 0.1, 0$, are available in [13], and have been reported in column $Y\&R$ of Table 1. Also, the exact values for the value function have been computed for these values of γ, using the closed-form formula and running a Monte Carlo, and are reported in the column Bench of Table 1. Tests of the Hybrid-Now and Hybrid-LaterQ algorithms have been run, and the estimations of the value function at time 0 and state $x = 0$ are reported in the Hybrid-Now and Hybrid-LaterQ columns. We also tested the Qknn algorithm based on quantization of the exogenous noise (ε_n) and k-nearest neighbors method, and reported the results in column Qknn. Qknn does not regress on neural networks, but rather uses k-nearest neighbors (knn) estimates

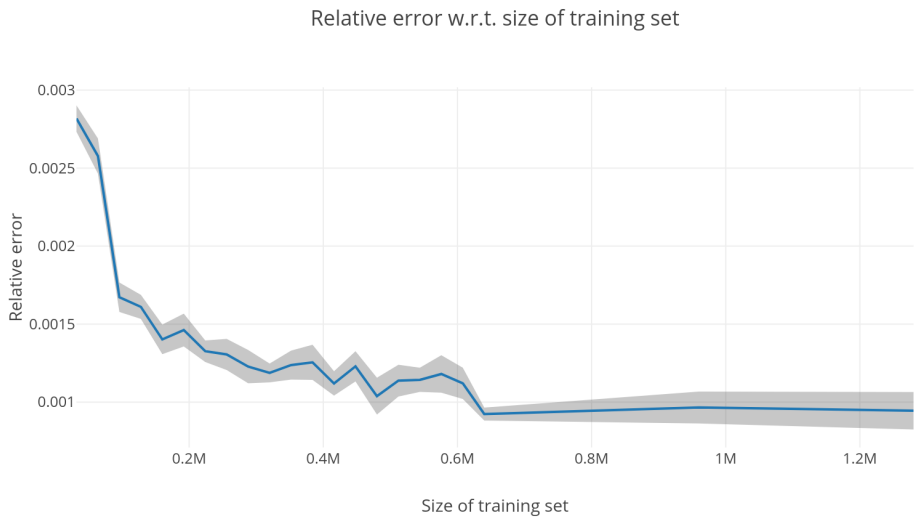

Figure 1: Relative error of the Hybrid-Now estimation of the value function at time 0 w.r.t the number of mini-batches used to build the Hybrid-Now estimators of the optimal strategy. The value functions have been computed running three times a forward Monte Carlo with a sample of size 10 000, following the optimal strategy estimated by the Hybrid-Now algorithm.

Table 1: Value function at time 0 and state 0 w.r.t. γ, computed with the Y&R, Hybrid-Now, Hybrid-Later and Qknn algorithms. Bench reports the MC estimations from the exact closed-form solution.

γ	Y&R	Hybrid-LaterQ	Hybrid-Now	Qknn	Bench
1.0	-0.402	-0.456	-0.460	-0.461	-0.464
0.5	-0.466	-0.495	-0.507	-0.508	-0.509
0.1	-0.573	-0.572	-0.579	-0.581	-0.586
0.0	-0.620	-1.000	-1.000	-1.000	-1.000

to approximate the Q-value. See [2] for a presentation, more details and several different tests of the Qknn algorithm. Note that Qknn is particularly well-suited to 1-dimensional control problems. In particular, it is not time-consuming since the dimension of the state space d is 1. Actually, it provides the fastest results, which is not surprising since the other algorithms need time to learn the optimal strategies and value functions through gradient-descent methods at each time step $n = 0, \dots, N-1$. Moreover, Table 1 reveals that Qknn is the most accurate algorithm on this example, probably because it uses local methods in space to estimate the conditional expectation that appears in the expression of the Q-value.

We end this paragraph by giving some implementation details for the different algorithms as part of **Test 2**.

- *Implementation details of Y&R algorithm:* Y&R algorithm requires to approximate the

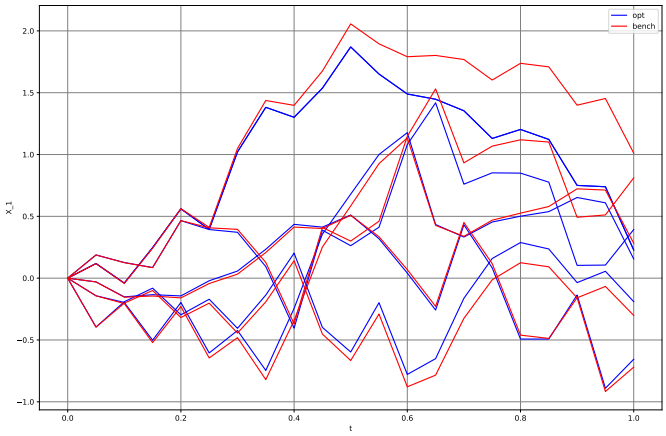

Figure 2: Pathwise comparison of the first component of X w.r.t. time when the agent follows the optimal strategy estimated by the Hybrid-Now algorithm (opt) and the naive strategy $\alpha = 0$ (bench). The dimension of the semilinear control problem has been set to d=2. Observe that, as expected, the strategy designed by Hybrid-Now algorithm is not to influence the diffusion of X when the terminal time is far in order to avoid any running cost, and try to make X small when terminal time gets close in order to minimize the terminal cost.

control problem by using a Lipschitz version of g like the following:

$$g_N(x) = \begin{cases} g(x) & \text{if } x \notin [0, N^{\frac{-1}{1-\gamma}}] \\ -Nx & \text{otherwise.} \end{cases}$$

- *Implementation details of Hybrid-Now algorithm:* We use $N = 40$ time steps for the discretization of the time interval $[0, 1]$. The value functions and optimal controls at time $n = 0, \ldots, N-1$ are estimated using neural networks with 3 hidden layers and 10+5+5 neurons.

- *Implementation details of Hybrid-LaterQ algorithm:* We use $N = 40$ time steps for the discretization of the time interval $[0, 1]$. The value functions and optimal controls at time $n = 0, \ldots, N-1$ are estimated using neural networks with 3 hidden layers containing 10+5+5 neurons; and 51 points for the quantization of the exogenous noise.

- *Implementation details of Qknn algorithm:* We use $N = 40$ time steps for the discretization of the time interval $[0, 1]$. We take 51 points to quantize the exogenous noise, $\varepsilon_n \sim \mathcal{N}(0, 1)$, for $n = 0, \ldots, N$; and 200 points for the space discretization. See [2] for more details on the Qknn algorithm.

The main conclusion regarding the numerical implementations and comparisons of this semilinear PDE is that the Hybrid-Now algorithm performs well in the control problem of dimension d=100, and outperforms the Hybrid-LaterQ algorithm in dimension d=2.

3.2 Option hedging

Our second example comes from a classical hedging problem in finance. We consider an investor who trades in q stocks with (positive) price process $(P_n)_n$, and we denote by (α_n) valued in $\mathbb{A} \subset \mathbb{R}^q$ the amount held in these assets on the period $(n, n+1]$. We assume for simplicity that the price of the riskless asset is constant equal to 1 (zero interest rate). It is convenient to introduce the return process as: $R_{n+1} = \mathrm{diag}(P_n)^{-1}(P_{n+1} - P_n)$, $n = 0, \ldots, N-1$, so that the self-financed wealth process of the investor with a portfolio strategy α, and starting from some capital w_0, is governed by

$$\mathcal{W}^\alpha_{n+1} \;=\; \mathcal{W}^\alpha_n + \alpha_n.R_{n+1}, \quad n = 0, \ldots, N-1, \;\; \mathcal{W}^\alpha_0 = w_0.$$

Given an option payoff $h(P_N)$, the objective of the agent is to minimize over her portfolio strategies α her expected square replication error

$$V_0 \;=\; \inf_{\alpha \in \mathcal{A}} \mathbb{E}\Big[\ell\big(h(P_N) - \mathcal{W}^\alpha_N\big)\Big],$$

where ℓ is a convex function on $\mathbb{R}$. Assuming that the returns R_n, $n = 1, \ldots, N$ are i.i.d, we are in a $(q+1)$-dimensional framework of Section 1 with $X^\alpha = (\mathcal{W}^\alpha, P)$ with $\varepsilon_n = R_n$ valued in $E \subset \mathbb{R}^q$, with the dynamics function

$$F(w,p,a,r) \;=\; \begin{cases} w + a.r \\ p + \mathrm{diag}(p)r, \end{cases} \qquad x = (w,p) \in \mathbb{R} \times \mathbb{R}^q, \; a \in \mathbb{R}^q, \; r \in E,$$

the running cost function $f = 0$ and the terminal cost $g(w,p) = \ell(h(p) - w)$. We test our algorithm in the case of a square loss function, i.e. $\ell(w) = w^2$, and when there is no portfolio constraints $\mathbb{A} = \mathbb{R}^q$, and compare our numerical results with the explicit solution derived in [3]: denote by $\nu(dr)$ the distribution of R_n, by $\bar\nu = \mathbb{E}[R_n] = \int r\nu(dr)$ its mean, and by $\bar M_2 = \mathbb{E}[R_n R_n^\intercal]$ assumed to be invertible; we then have

$$V_n(w,p) \;=\; K_n w^2 - 2Z_n(p)w + C_n(p)$$

where the functions $K_n > 0$, $Z_n(p)$ and $C_n(p)$ are given in backward induction, starting from the terminal condition

$$K_N \;=\; 1, \;\; Z_N(p) \;=\; h(p), \;\; C_N(p) \;=\; h^2(p),$$

and for $n = N-1, \ldots, 0$, by

$$\begin{aligned}
K_n \;&=\; K_{n+1}\big(1 - \bar\nu^\intercal \bar M_2^{-1} \bar\nu\big), \\
Z_n(p) \;&=\; \int Z_{n+1}(p + \mathrm{diag}(p)r)\nu(dr) - \bar\nu^\intercal \bar M_2^{-1} \int Z_{n+1}(p + \mathrm{diag}(p)r) r \nu(dr), \\
C_n(p) \;&=\; \int C_{n+1}(p + \mathrm{diag}(p)r)\nu(dr) \\
&\qquad - \frac{1}{K_{n+1}}\Big(\int Z_{n+1}(p + \mathrm{diag}(p)r) r\nu(dr)\Big)^\intercal \bar M_2^{-1}\Big(\int Z_{n+1}(p + \mathrm{diag}(p)r) r\nu(dr)\Big),
\end{aligned}$$

so that $V_0 = K_0 w_0^2 - 2Z_0(p_0)w_0 + C_0(p_0)$, where p_0 is the initial stock price. Moreover, the optimal portfolio strategy is given in feedback form by $\alpha_n^* = a_n^*(\mathcal{W}_n^*, P_n)$, where $a_n^*(w,s)$ is the function

$$a_n^*(w,p) = \bar{M}_2^{-1}\left[\frac{\int Z_{n+1}(p+\mathrm{diag}(p)r)r\nu(dr)}{K_{n+1}} - \bar{\nu}w\right],$$

and $\mathcal{W}^*$ is the optimal wealth associated with α^*, i.e., $\mathcal{W}_n^* = \mathcal{W}_n^{\alpha^*}$. Moreover, the initial capital w_0^* that minimizes $V_0 = V_0(w_0, p_0)$, and called (quadratic) hedging price is given by

$$w_0^* \quad = \quad \frac{Z_0(p_0)}{K_0}.$$

Test Take $N = 6$, and consider one asset $q = 1$ with returns modeled by a trinomial tree:

$$\nu(dr) \quad = \quad \pi_+\delta_{r_+} + \pi_0\delta_0 + \pi_-\delta_{r_-}, \quad \pi_0 + \pi_+ + \pi_- = 1,$$

with $r_+ = 5\%$, $r_- = -5\%$, $\pi_+ = 60\%$, $\pi_- = 30\%$. Take $p_0 = 100$, and consider the call option $h(p) = (p-\kappa)_+$ with $\kappa = 100$. The price of this option is defined as the initial value of the portfolio that minimizes the terminal quadratic loss of the agent when the latter follows the optimal strategy associated with the initial value of the portfolio. In this test, we want to determine the price of the call and the associated optimal strategy using different algorithms.

Numerical results In Figure 3, we plot the value function at time 0 w.r.t w_0, the initial value of the portfolio, when the agent follows the theoretical optimal strategy (benchmark), and the optimal strategy estimated by the Hybrid-Now or Hybrid-LaterQ algorithms. We perform forward Monte Carlo using 10,000 samples to approximate the lower bound of the value function at time 0 (see [9] for details on how to get an approximation of the upper-bound of the value function via duality). One can observe that while all the algorithms give a call option price approximately equal to 4.5, Hybrid-LaterQ clearly provides a better strategy than Hybrid-Now to reduce the quadratic risk of the terminal loss.

We plot in Figure 4 three different paths of the value of the portfolio w.r.t the time n, when the agent follows either the theoretical optimal strategy (red), or the estimated one using the Hybrid-Now algorithm (blue) or Hybrid-LaterQ algorithm (green). We set $w_0 = 100$ for these simulations. Note that for such a big value of w_0, it is not obvious that Hybrid-LaterQ is better than Hybrid-Now.

Comments on the Hybrid-Now and Hybrid-LaterQ algorithms The Option Hedging problem belongs to the class of the linear quadratic control problems for which we expect the optimal control to be affine in w and the value function to be quadratic in w. It is then natural to consider the following classes of controls $\mathcal{A}_M$ and functions $\mathcal{F}_M$ to properly approximate the optimal controls and the values functions at time n=$0, \ldots, N-1$:

$$\mathcal{A}_M := \left\{(w,p) \mapsto A(x;\beta)\cdot\big(1,w\big)^\intercal; \quad \beta \in \mathbb{R}^p\right\}, \tag{3.4}$$

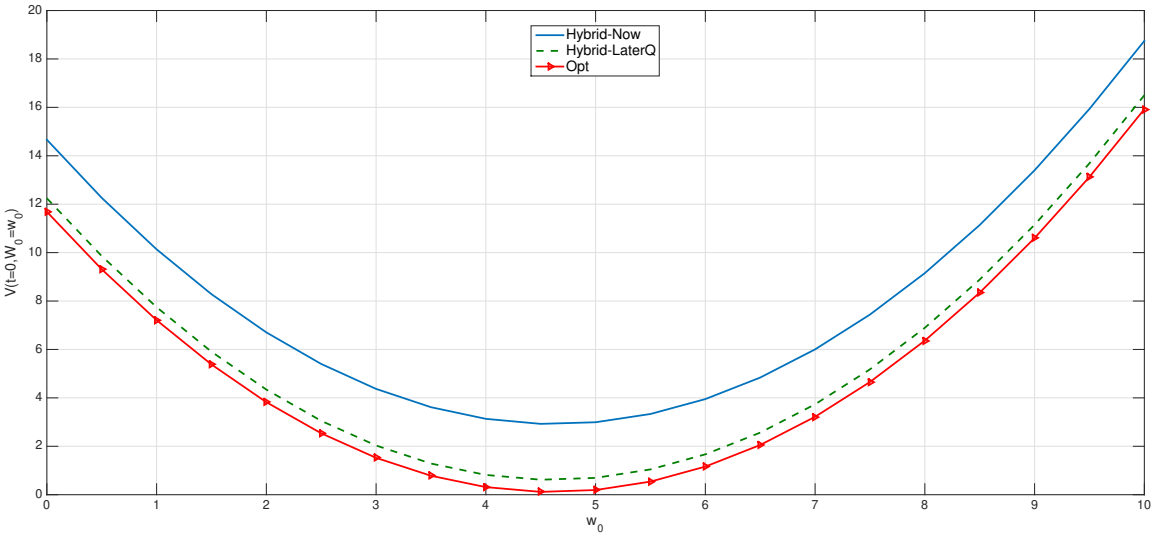

Figure 3: Estimations of the value function at time 0 w.r.t. w_0 using the Hybrid-Now algorithm (blue line), Hybrid-LaterQ algorithm (green dashes). We draw the value function in red for comparison. One can observe that the price of the call given by all the algorithms is approximately equals to 4.5, but Hybrid-LaterQ is better than Hybrid-Now at reducing the quadratic risk.

$$\mathcal{F}_M := \left\{(w,p) \mapsto \Phi(x;\theta)\cdot\left(1,w,w^2\right)^\intercal; \quad \theta \in \mathbb{R}^p\right\}, \tag{3.5}$$

where β describes the parameters (weights+ bias) associated with the neural network A and θ describes those associated with the neural network Φ. The notation $^\intercal$ stands for the transposition, and $\cdot$ for the inner product. Note that there are 2 (resp. 3) neurons in the output layer of A (resp. Φ), so that the inner product is well-defined in (3.5) and (3.4). It is then natural to use gradient-descent-based algorithms to find the optimal parameter β (resp. θ) for which A (resp. Φ) coincides with the optimal control (resp. the value function) at time $n{=}0,\ldots,N-1$.

Remark 3.1 The option hedging problem is linear quadratic, hence belongs to the class of problems where the agent has ansatzes on the optimal control and the value function. For these kind of problems, the algorithms presented in [11] can easily be adapted so that the expressions of the estimators satisfy the ansatzes, see e.g. (3.4) and (3.5). □

3.3 Valuation of energy storage

We present a discrete-time version of the energy storage valuation problem studied in [4]. We consider a commodity (gas) that has to be stored in a cave, e.g. salt domes or aquifers. The manager of such a cave aims to maximize the real options value by optimizing over a finite horizon N the dynamic decisions to inject or withdraw gas as time and market conditions evolve. We denote by (P_n) the gas price, which is an exogenous real-valued Markov process modeled by the following mean-reverting process:

$$P_{n+1} \quad = \quad \bar{p}(1-\beta) + \beta P_n + \xi_{n+1}, \tag{3.6}$$

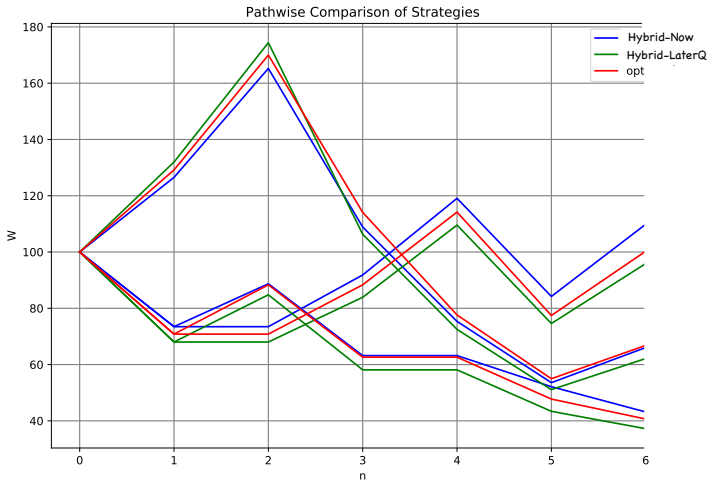

Figure 4: Three simulations of the agent's wealth w.r.t. the time n when, for each ω, the latter follows the theoretical optimal strategy (red), and the estimated one using the Hybrid-Now (blue) or Hybrid-LaterQ algorithm (green). We took $w_0 = 100$ for this simulation. Observe that for such a big value of w_0, the optimal strategy estimated by the Hybrid-LaterQ and the Hybrid-Now algorithms are similar to the theoretical optimal strategy.

where $\beta < 1$, and $\bar{p} > 0$ is the stationary value of the gas price. The current inventory in the gas storage is denoted by $(C_n^\alpha)_n$ and depends on the manager's decisions represented by a control process $\alpha = (\alpha_n)$ valued in $\{-1, 0, 1\}$: $\alpha_n = 1$ (resp. -1) means that she injects (resp. withdraws) gas with an injection (resp. withdrawal) rate $a_{in}(C_n^\alpha)$ (resp. $a_{out}(C_n^\alpha)$) requiring (causing) a purchase (resp. sale) of $b_{in}(C_n^\alpha) \geq a_{in}(C_n^\alpha)$ (resp. $b_{out}(C_n^\alpha) \leq a_{out}(C_n^\alpha)$), and $\alpha_n = 0$ means that she is doing nothing. The difference between b_{in} and a_{in} (resp. b_{out} and a_{out}) indicate gas loss during injection/withdrawal. The evolution of the inventory is then governed by

$$C_{n+1}^\alpha \;=\; C_n^\alpha + h(C_n^\alpha, \alpha_n), \quad n = 0, \ldots, N-1, \; C_0^\alpha \;=\; c_0, \tag{3.7}$$

where we set

$$h(c,a) \;=\; \begin{cases} a_{in}(c) & \text{for } \; a = 1 \\ 0 & \text{for } \; a = 0 \\ -a_{out}(c) & \text{for } \; a = -1, \end{cases}$$

and we have the physical inventory constraint:

$$C_n^\alpha \;\in\; [C_{min}, C_{max}], \quad n = 0, \ldots, N.$$

The running gain of the manager at time n is $f(P_n, C_n^\alpha, \alpha_t)$ given by

$$f(p,c,a) \;=\; \begin{cases} -b_{in}(c)p - K_1(c) & \text{for } \; a = 1 \\ -K_0(c) & \text{for } \; a = 0 \\ b_{out}(c)p - K_{-1}(c) & \text{for } \; a = -1, \end{cases}$$

and $K_i(c)$ represents the storage cost in each regime $i = -1, 0, 1$. The problem of the manager is then to maximize over α the expected total profit

$$J(\alpha) \;=\; \mathbb{E}\Big[\sum_{n=0}^{N-1} f(P_n, C_n^\alpha, \alpha_n) + g(P_N, C_N^\alpha)\Big],$$

where a common choice for the terminal condition is

$$g(p, c) \;=\; -\mu p(c_0 - c)_+,$$

which penalizes for having less gas than originally, and makes this penalty proportional to the current price of gas ($\mu > 0$). We are then in the 2-dimensional framework of Section 1 with $X^\alpha = (P, C^\alpha)$, and the set of admissible controls in the dynamic programming loop is given by:

$$A_n(c) = \big\{a \in \{-1, 0, 1\} : c + h(c, a) \in [C_{min}, C_{max}],\ c \in [C_{min}, C_{max}]\big\},\ \ n = 0, \ldots, N-1.$$

Test As in [4], we consider the example

$$a_{in}(c) \;=\; b_{in}(c) \;=\; 0.06, \qquad a_{out}(c) \;=\; b_{out}(c) \;=\; 0.25$$
$$K_i(c) \;=\; 0.01c$$

$C_{max} = 8$, $C_{min} = 0$, $c_0 = 4$, $\bar{p} = 5$, $\beta = 0.5$, $\xi_{n+1} \rightsquigarrow \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = 0.05$, and $\mu = 2$ in the terminal penalty function, $N = 10$, 20, 30.

Numerical results Figure 5 provides the value function estimates at time 0 w.r.t. a_{in} using Qknn algorithm, compared to the benchmark (Bench) defined as the naive do-nothing strategy $\alpha = 0$. As expected, the naive strategy performs well when a_{in} is small, since, in this case, it takes time to fill the cave, so that the agent is likely to do nothing so as not to be penalized at terminal time. When a_{in} is large, it is easy to fill up the cave, so the agent has more freedom to buy and sell gas in the market without worrying about the terminal cost. Observe that the value function is not monotone, due to the fact that the state space for the volume of gas in the cave is a bounded discrete set.

Table 2 provides the value function estimates obtained with the ClassifPI, Hybrid-Now and Hybrid-LaterQ algorithms. Observe first that the estimations provided by the Qknn algorithm are larger than those provided by the other algorithms, meaning that Qknn outperforms the other algorithms. The second best algorithm is ClassifPI, while the performance of Hybrid-Now is poor and clearly suffers from instability, due probably to the discontinuity of the running rewards w.r.t. the control variable.

Finally, Figures 6, 7, 8 provide the optimal decisions w.r.t. (P, C) at times 5, 10, 15, 20, 25, 29 estimated respectively by the Qknn, ClassifPI and Hybrid-Now algorithms. As expected, one can observe on each plot that the optimal strategy is to inject gas when the price is low, to sell gas when the price is high, and to make sure to have a volume of gas greater than c_0 in the cave when the terminal time is getting closer to minimize the terminal cost. Let us now comment on the implementation of algorithms:

Table 2: $V(0, P_0, C_0)$ estimates for different values of a_{in}, using the optimal strategy provided by the ClassifPI , Hybrid-Now and Qknn algorithms, with $a_{out} = 0.25$, $P_0 = 4$ and $C_0 = 4$.

a_{in}	Hybrid-Now	ClassifPI	Qknn	$\alpha = 0$
0.06	-0.99	-0.71	-0.66	-1.20
0.10	-0.70	-0.38	-0.34	-1.20
0.20	-0.21	0.01	0.12	-1.20
0.30	-0.10	0.37	0.37	-1.20
0.40	0.10	0.51	0.69	-1.20

- *Comments on the Qknn algorithm:* Table 2 shows that once again, due to the low-dimensionality of the problem, Qknn provides the best value function estimates. The estimated optimal strategies, shown on Figure 6, are very good estimations of the theoretical ones. The three decision regions on Figure 6 are natural and easy to interpret: basically it is optimal to sell when the price is high, and to buy when it is low. However, a closer look reveals that the waiting region (where it is optimal to do nothing) has an unusual triangular-based shape, which, while close to the theoretical one, can be expected to be very hard to reproduce with the DNN-based algorithms proposed in [11].

- *Comments on the ClassifPI algorithm:* As shown on Figure 7, the ClassifPI algorithm manages to provide stable estimates for the optimal controls at time $n = 0, \ldots, N-1$. However, the latter is not able to catch the particular triangular-based shape of the waiting region, which explains why Qknn performs better.

- *Comments on the Hybrid-Now algorithm:* As shown on Figure 8, the Hybrid-Now algorithm only manages to provide a weak estimation of the three different regions at time $n = 0, \ldots, N-1$. In particular, the regions suffer from instability.

We end this paragraph by providing some implementation details for the different algorithms we tested.

- *Implementation details for the Qknn algorithm:* We recall that the Qknn algorithm is based on the quantization and k-nearest neighbors methods to estimate the value functions at time $n = 0, \ldots, N-1$. We take the $k = 2$ closest neighbors for the estimation of the regression of the value functions, in order to insure continuity of the estimation w.r.t. the pair (p, c) of state variables. The optimal control is computed at each point of the grid using deterministic optimizers such as the Golden-section search or the Brent algorithm, which are classical optimization routines available in many numerical libraries.

- *Implementation details for the neural network-based algorithms:* We use neural networks with two hidden layers, ELU activation functions[a] and 20+20 neurons . The output layer contains 3 neurons with softmax activation function for the ClassifPI algorithm and no

[a]The Exponential Linear Unit (ELU) activation function is defined as $x \mapsto \begin{cases} \exp(x) - 1 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$.

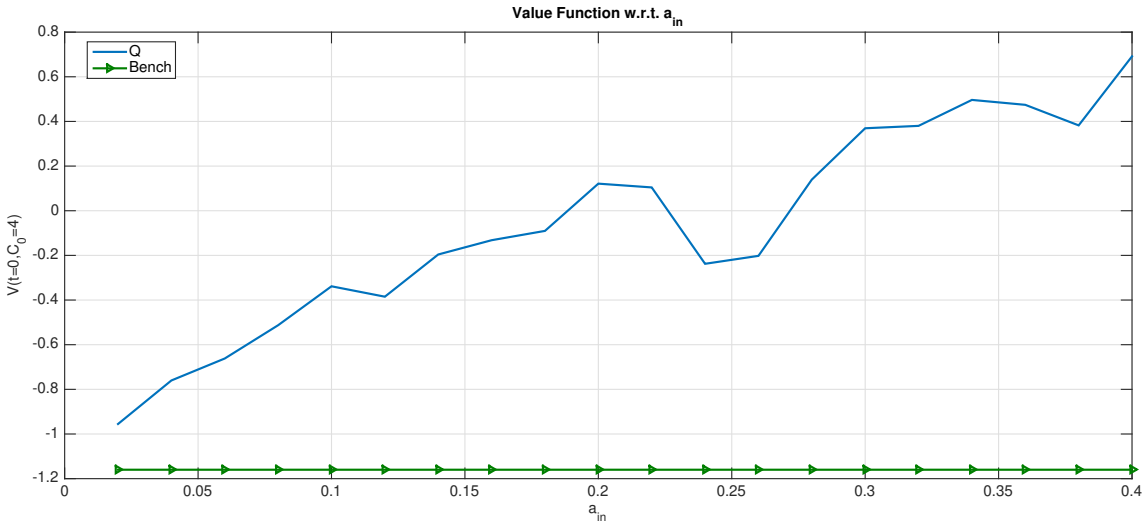

Figure 5: Value functions estimates at time 0 w.r.t. a_{in}, when the agent follows the optimal strategy estimated by the Qknn algorithm, by running a forward Monte Carlo with a sample of size 100,000 (blue). We also plotted the cost functional associated with the naive passive strategy $\alpha = 0$ (Bench). See that for small values of a_{in} such as 0.06, doing nothing is a reasonable strategy. In this case, the naive strategy is a good benchmark to test the algorithms.

activation function for the Hybrid-Now one. We use a training set of size $M = 60,000$ at each time step. Note that given the expression of the terminal cost, the ReLU activation functions (Rectified Linear Units) could have been deemed a better choice to capture the shape of the value functions, but our tests revealed that ELU activation functions provide better results.

The main conclusion of our numerical comparisons on this energy storage example is that ClassifPI, the DNN-based classification algorithm designed for stochastic control problems with discrete control space, appears to be more accurate than the more general Hybrid-Now. Nevertheless, ClassifPI was not able to capture the unusual triangle-based shape of the optimal control as well as Qknn did.

3.4 Microgrid management

Finally, we consider a discrete-time model for power microgrid inspired by the continuous-time models developed in [10] and [12]; see also [1]. The microgrid consists of a photovoltaic (PV) power plant, a diesel generator and a battery energy storage system (BES), hence using a mix of fuel and renewable energy sources. These generation units are decentralized, i.e., installed at a rather small scale (a few kW power), and physically close to electricity consumers. The PV produces electricity from solar panels with a generation pattern $(P_n)_n$ depending on the weather conditions. The diesel generator has two modes: on and off. Turning it on consumes fuel, and produces an amount of power α_n. The BES can store energy for later use but has limited capacity and power. The aim of the microgrid management is to find the optimal planning that meets the power demand, denoted by $(D_n)_n$,

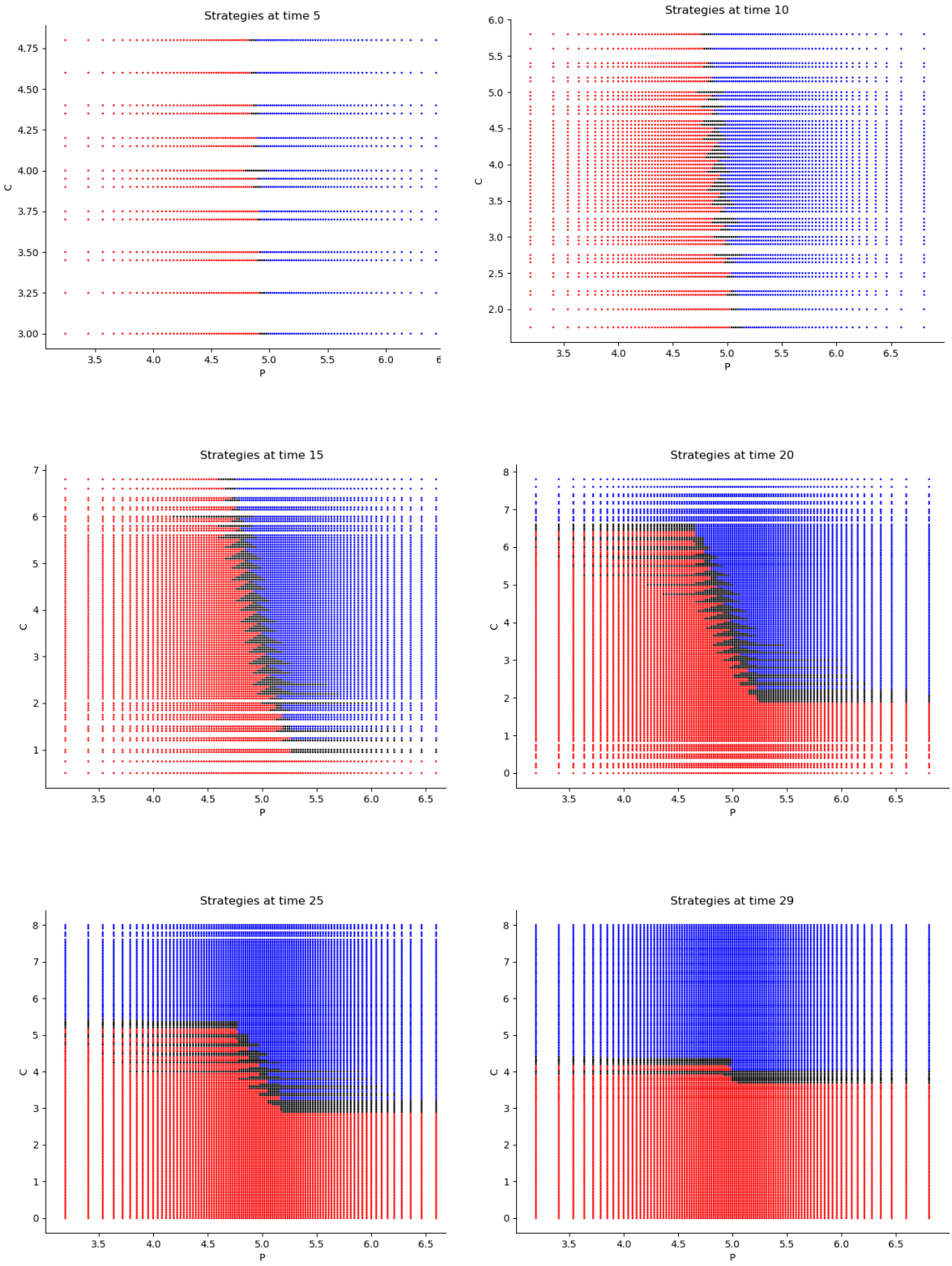

Figure 6: Estimated optimal decisions at times 5, 10, 15, 20, 25, 29 w.r.t. (P,C) for the energy storage valuation problem using the **Qknn** algorithm. Injection (a=-1) in red, store (a=0) in black and withdraw (a=1) in blue.

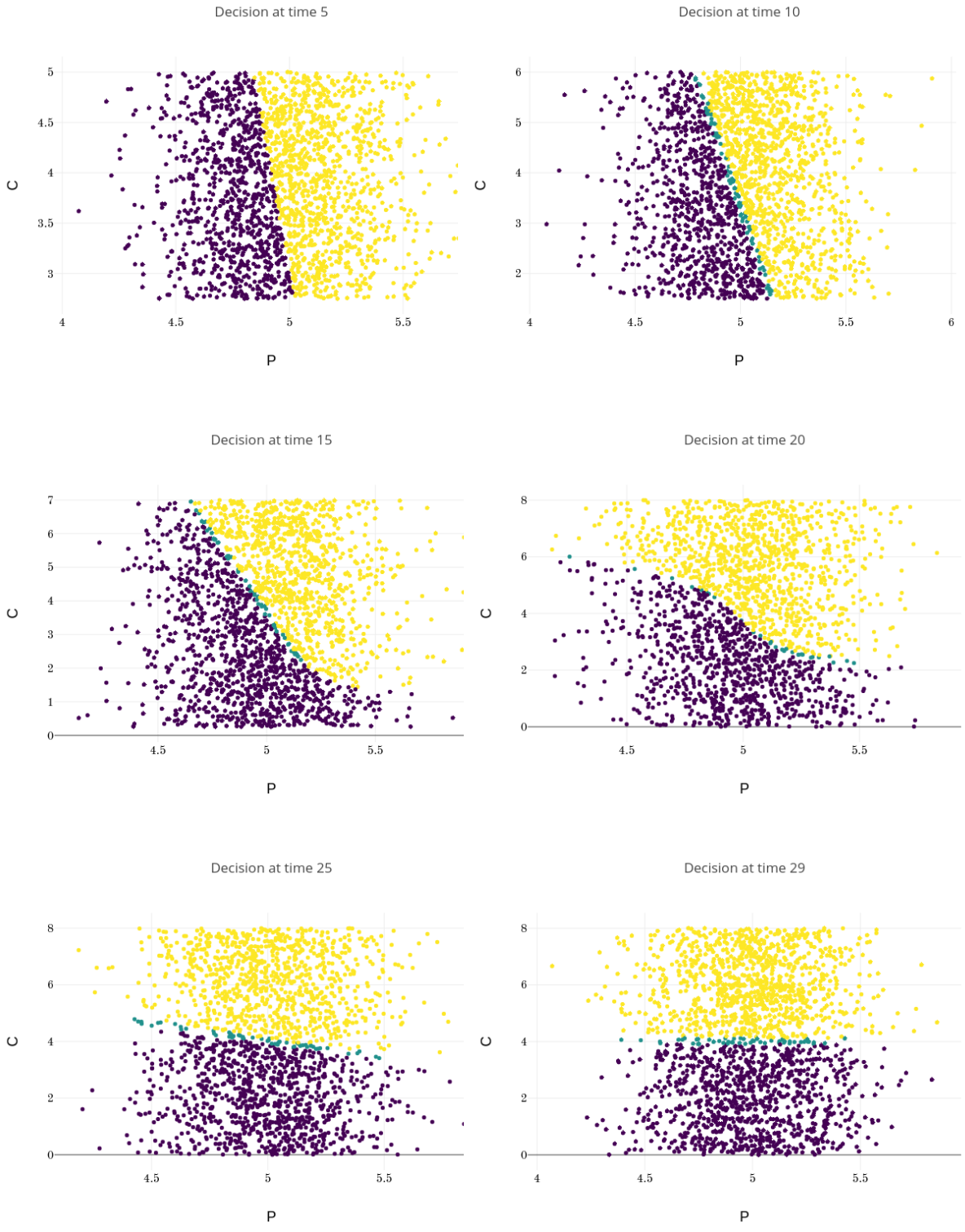

Figure 7: Estimated optimal decisions at times 5, 10, 15, 20, 25, 29 w.r.t. (P,C) for the energy storage valuation problem using the **ClassifPI** algorithm. Injection (a=-1) in purple, store (a=0) in blue and withdraw (a=1) in yellow.

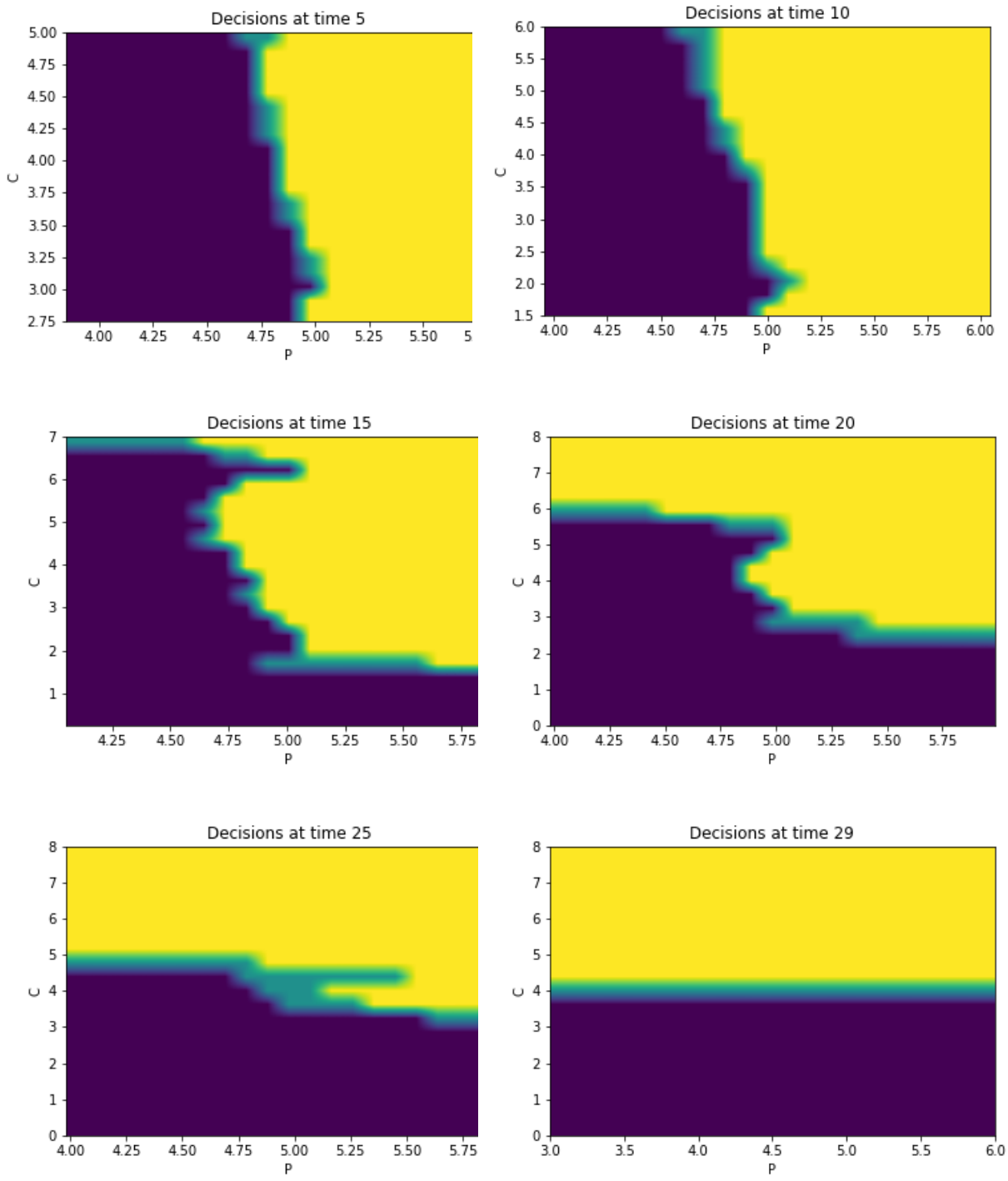

Figure 8: Estimated optimal decisions at times 5, 10, 15, 20, 25, 29 w.r.t. (P,C) for the energy storage valuation problem using the **Hybrid-Now** algorithm. Injection (a=-1) in purple, store (a=0) in blue and withdraw (a=1) in yellow.

while minimizing the operational costs due to the diesel generator. We denote by

$$R_n \;=\; D_n - P_n,$$

the residual demand of power: when $R_n > 0$, one should provide power through diesel or battery, and when $R_n < 0$, one can store the surplus power in the battery.

The optimal control problem over a fixed horizon N is formulated as follows. At any time $n = 0, \dots, N-1$, the microgrid manager decides the power production of the diesel generator, either by turning it off: $\alpha_n = 0$, or by turning it on, hence generating a power α_n valued in $[A_{min}, A_{max}]$ with $0 < A_{min} < A_{max} < \infty$. There is a fixed cost $\kappa > 0$ associated with switching from the on/off mode to the other one off/on, and we denote by M_n^α the mode valued in $\{0 = \text{off}, 1 = \text{on}\}$ of the generator right before time n, i.e., $M_{n+1}^\alpha = 1_{\alpha_n \neq 0}$.

When the diesel generator and renewable provide a surplus of power, the excess can be stored into the battery (up to its limited capacity) for later use, and in case of power insufficiency, the battery is discharged for satisfying the power demand. The input power process $\mathcal{I}^\alpha$ for charging the battery is then given by

$$\mathcal{I}_n^\alpha \;=\; (\alpha_n - R_n)_+ \wedge (C_{max} - C_n^\alpha),$$

where C_{max} is the maximum capacity of the battery with current charge C^α, while the output power process O^α for discharging the battery is given by

$$O_n^\alpha \;=\; (R_n - \alpha_n)_+ \wedge C_n^\alpha.$$

Here, we denote $p_+ = \max(p, 0)$. Assuming for simplicity that the battery is fully efficient, the capacity charge $(C_n^\alpha)_n$ of the BES, valued in $[0, C_{max}]$, evolves according to the dynamics

$$C_{n+1}^\alpha \;=\; C_n^\alpha + \mathcal{I}_n^\alpha - O_n^\alpha. \tag{3.8}$$

The imbalance process defined by

$$S_n^\alpha \;=\; R_n - \alpha_n + \mathcal{I}_n^\alpha - O_n^\alpha$$

represents how well we are doing for satisfying electricity supply: the ideal situation occurs when $S_n^\alpha = 0$, i.e., perfect balance between demand and generation. When $S_n^\alpha > 0$, this means that demand is not satisfied, i.e., there is missing power in the microgrid, and when $S_n^\alpha < 0$, there is an excess of electricity. In order to ensure that there is no missing power, we impose the following constraint on the admissible control:

$$S_n^\alpha \;\leq\; 0, \quad \text{i.e.} \quad \alpha_n \;\geq\; R_n - C_n^\alpha,$$

but penalize the excess of electricity when $S_n^\alpha < 0$ with a proportional cost $Q^- > 0$.

We model the residual demand as a mean-reverting process:

$$R_{n+1} \;=\; \bar{R}(1 - \varrho) + \varrho R_n + \varepsilon_{n+1},$$

where $(\varepsilon_n)_n$ are i.i.d., $\bar{R} \in \mathbb{R}$, and $\varrho < 1$. The goal of the microgrid manager is to find the optimal (admissible) decision α that minimizes the functional cost

$$J(\alpha) \;=\; \mathbb{E}\left[\sum_{n=0}^{N-1} \ell(\alpha_n) + \kappa 1_{\{M_n^\alpha \neq M_{n+1}^\alpha\}} + Q^-(S_n^\alpha)_-\right],$$

where $\ell(.)$ is the cost function for fuel consumption: $\ell(0) = 0$, and e.g. $\ell(a) = Ka^\gamma$, with $K > 0$, $\gamma > 0$. This stochastic control problem fits into the 3-dimensional framework of Section 1 (see also Remark 1.1) with control α valued in $\mathbb{A} = \{0\} \times [A_{min}, A_{max}]$, $X^\alpha = (C^\alpha, M^\alpha, R)$, noise ε_{n+1}, starting from an initial value $(C_0^\alpha, M_0^\alpha, R_0) = (c_0, 0, r_0)$ on the state space $[0, C_{max}] \times \{0, 1\} \times \mathbb{R}$, with dynamics function

$$F(x, a, e) \quad = \quad \begin{pmatrix} F^1(x, a) := c + (a - r)_+ \wedge (C_{max} - c) - (r - a)_+ \wedge c \\ 1_{a \neq 0} \\ \bar{R}(1 - \varrho) + \varrho r + e \end{pmatrix},$$

for $x = (c, m, r) \in [0, C_{max}] \times \{0, 1\} \times \mathbb{R}$, $a \in \{0\} \times [A_{min}, A_{max}]$, $e \in \mathbb{R}$, running cost function

$$\begin{aligned} f(x, a) &= \ell(a) + \kappa 1_{m=1_{a=0}} + Q^- S(x, a)_-, \\ S(x, a) &= r - a + (a - r)_+ \wedge (C_{max} - c) - (r - a)_+ \wedge c, \end{aligned}$$

zero terminal cost $g = 0$, and control constraint

$$\begin{aligned} \mathbb{A}_n(x) &= \Big\{ a \in \{0\} \times [A_{min}, A_{max}] : S(x, a) \leq 0 \Big\} \\ &= \Big\{ a \in \{0\} \times [A_{min}, A_{max}] : r - c \leq a \Big\}. \end{aligned}$$

Tests The state/space constraint is managed by introducing into the running cost a penalty function (see Remark 1.1): $f(x, a) \leftarrow f(x, a) + L(x, a)$

$$L(x, a) \quad = \quad Q^+ \Big(r - c - a \Big)_+$$

with large $Q^+ > 0$, much larger than Q^-. In practice we set $Q^+ = 1,000$.

The control space $\{0\} \cup [A_{\min}, A_{\max}]$ is a mix between a discrete space and a continuous space, which is challenging for algorithms with neural networks. We actually use a mixture of classification and standard DNN for the control: $(p_0(x; \theta), \pi(x; \beta))$ valued in $[0, 1] \times [A_{min}, A_{max}]$, where $p_0(x; \theta)$ is the probability of turning off in state x, and $\pi(x; \beta)$ is the amount of power when turning on with probability $1 - p_0(x; \theta)$. In other words,

$$X_{n+1} \quad = \quad \begin{cases} F(X_n, 0, \varepsilon_{n+1}) & \text{with probability } p_0(X_n; \theta_n) \\ F(X_n, \pi(X_n; \beta_n), \varepsilon_{n+1}) & \text{with probability } 1 - p_0(X_n; \theta_n) \end{cases}$$

Our algorithm is then written as

- For $n = N-1, \ldots, 0$, keep track of the optimal feedback control $\hat{a}_k(.)$, $k = n+1, \ldots, N-1$, valued in $\{0\} \times [A_{min}, A_{max}]$, and compute

$$\begin{aligned} (\hat{\theta}_n, \hat{\beta}_n) \quad \in \quad & \arg\max_{\theta, \beta} \mathbb{E}\Big[p_0(X_n; \theta) \Big(f(X_n, 0) + \sum_{k=n+1}^{N-1} f(\hat{X}_k^0, \hat{a}_k(\hat{X}_k^0) \Big) \\ & + (1 - p_0(X_n; \theta)) \Big(f(X_n, \pi(X_n; \beta)) + \sum_{k=n+1}^{N-1} f(\hat{X}_k^{1,\beta}, \hat{a}_k(\hat{X}_k^{1,\beta}) \Big) \Big], \end{aligned}$$

where $X_n \rightsquigarrow \mu_n$ a training distribution, $\hat{X}^0_{n+1} = F(X_n, 0, \varepsilon_{n+1})$, $\hat{X}^0_{k+1} = F(\hat{X}^0_k, \hat{a}_k(\hat{X}^0_k), \varepsilon_{k+1})$, for $k = n+1, \dots, N-1$, while $\hat{X}^{1,\beta}_{n+1} = F(X_n, \pi(X_n;\beta), \varepsilon_{n+1})$, $\hat{X}^{1,\beta}_{k+1} = F(\hat{X}^{1,\beta}_k, \hat{a}_k(\hat{X}^{1,\beta}_k), \varepsilon_{k+1})$, for $k = n+1, \dots, N-1$.

- Update the optimal feedback control at time n:

$$\hat{a}_n(x) \quad = \quad \begin{cases} 0 & \text{if } p_0(x;\hat{\theta}_n) > \frac{1}{2} \\ \pi(x;\hat{\beta}_n) & \text{if } p_0(x;\hat{\theta}_n) \leq \frac{1}{2} \end{cases}$$

Numerical results The microgrid management problem, which can be seen as a problem of dimension "almost" 2 as the M state component can only take two values 0 or 1, is particularly easy to solve using Monte Carlo-based or quantization-based algorithms. In [1], the authors propose several Monte Carlo-based methods to numerically solve a very similar problem. We choose to use the Qknn algorithm here, since it also provides accurate and fast results. We take the following parameters to test the Qknn algorithm:

$$\begin{array}{rclrclrclrcl} N & = & 30 \text{ or } 200, & \bar{R} & = & 0.1, & \varrho & = & 0.9, & \sigma & = & 0.2, \\ C_{\min} & = & 0, & C_{\max} & = & 1 \text{ or } 3, & C_0 & = & 0, & K & = & 2, \\ \gamma & = & 2, & \kappa & = & 0.2, & Q^- & = & 10, & R_0 & = & 0.1, \\ A_{\min} & = & 0.05, & A_{\max} & = & 10. & & & & & & \end{array}$$

Note that there is no need to use a penalization method with the Qknn-algorithm to constrain the control to stay in $\mathbb{A}_n(x)$, where x is the state at time n, since we simply look for the optimal control in $\mathbb{A}_n(x)$, using a deterministic Brent or Golden Search algorithm. Figure 10 shows the Qknn-estimated optimal decisions to take at times $n = 1, 10, 28$ in the cases where $m = M_n = 0$ and $m = M_n = 1$. If the generator is off at time n, i.e. $m = 0$, the blue curve separates the region where it is optimal to keep it off and the one where it is optimal to generate power. If the generator is on at time n, i.e. $m = 1$, the blue curve separates the region where it is optimal to turn it off and the one where it is optimal to generate power. A colorscale is available on the right to inform how much power it is optimal to generate in both cases. Observe that the optimal decisions are quite intuitive: for example, if the demand is high and the battery is empty, then it is optimal to generate a lot of energy. Moreover, it is optimal to turn the generator off if the demand is negative or if the battery is charged enough to meet the demand. Note that, once again, the plots in Figure 10 are very similar to the ones obtained in [1]. For comparison, we also plot in Figure 11 the estimated optimal decisions at times $n = 1, 10, 28$, using the NN-based algorithm, with $N = 30$ time steps.

Without tuning the parameters, we report in Table 3 the result for the estimation of the value function with N=200 time steps, obtained by running 20 times a forward Monte Carlo with 10 000 simulations using the Qknn-estimated optimal strategy. Figure 9 shows two simulations of (C, M, R) controlled using the Qknn-estimated optimal strategy, where $N = 200$ has been chosen. Observe in particular the natural behavior of the Qknn-decisions which consists in turning the generator on when the demand cannot be met by the battery, and turn it off when the demand is negative or when the battery is charged enough to meet

Table 3: Qknn-estimations of the value function at time 0 and state $(C_0 = 0, M_0 = 0, R_0 = 0.1)$, for $N = 200$.

Mean	Standard Deviation
231.8	1.2

the demand. Note that the plots are very similar to the ones obtained in [1].

The microgrid management problem is very challenging for our algorithms because the control space $\{0\} \cup [a_{\min}, a_{\max}]$ is a mix of discrete and continuous space, moreover the choice of the optimal control is subject to constraints. As expected, our algorithms perform well, but their results stay far from those obtained by the Qknn algorithm. Note that the microgrid management problem is definitely not challenging in low dimension for algorithms such as Qknn, so that we could provide easily very accurate results for the problem with N=200 time steps.

4 Discussion and conclusion

Our proposed algorithms are well-designed and provide accurate estimates of optimal control and value function associated with various high-dimensional control problems. We also tested their performances on low-dimensional problems, and concluded that they perform well, but remain far from the Monte Carlo-based or quantization-based methods which have proven their efficiency in low dimension, see e.g. [2] and [1]).

The presented algorithms suffer from a very high time-consumption due to the daunting training task of $2(N-1)$ neural networks to get approximation of the value functions and optimal controls at times $n = 0, \ldots, N-1$. We suggest different ideas to overcome this drawback: the first one is to reduce the number of neural networks to train by partially or totally ignoring the programming dynamic principle (DPP). See [6] for a method where the DPP is totally ignored. The use of one unique Recurrent Neural Networks (RNN) (in the case where the DPP is totally ignored) or a few of them (in the other case) can also be considered to learn the optimal controls, either all at the same time (first case), or group by group in a backward way (second case). Another idea consists in learning faster the value functions and optimal controls at times $n = 0, \ldots, N-1$ either by introducing Bayesian methods such as Gaussian Processes (GPs) and learning optimal parameters for the latter with neural networks methods, see [7] for more details in this direction; or by pre-training the neural networks so that less data will be needed to train the neural networks. One idea in that direction is to initialize at time n the weights and bias of value function estimator $\hat{V}_n$ to the ones of $\hat{V}_{n+1}$, and rely on the continuity of the value function w.r.t. the time n to make the training-task faster since it starts from a good guess.

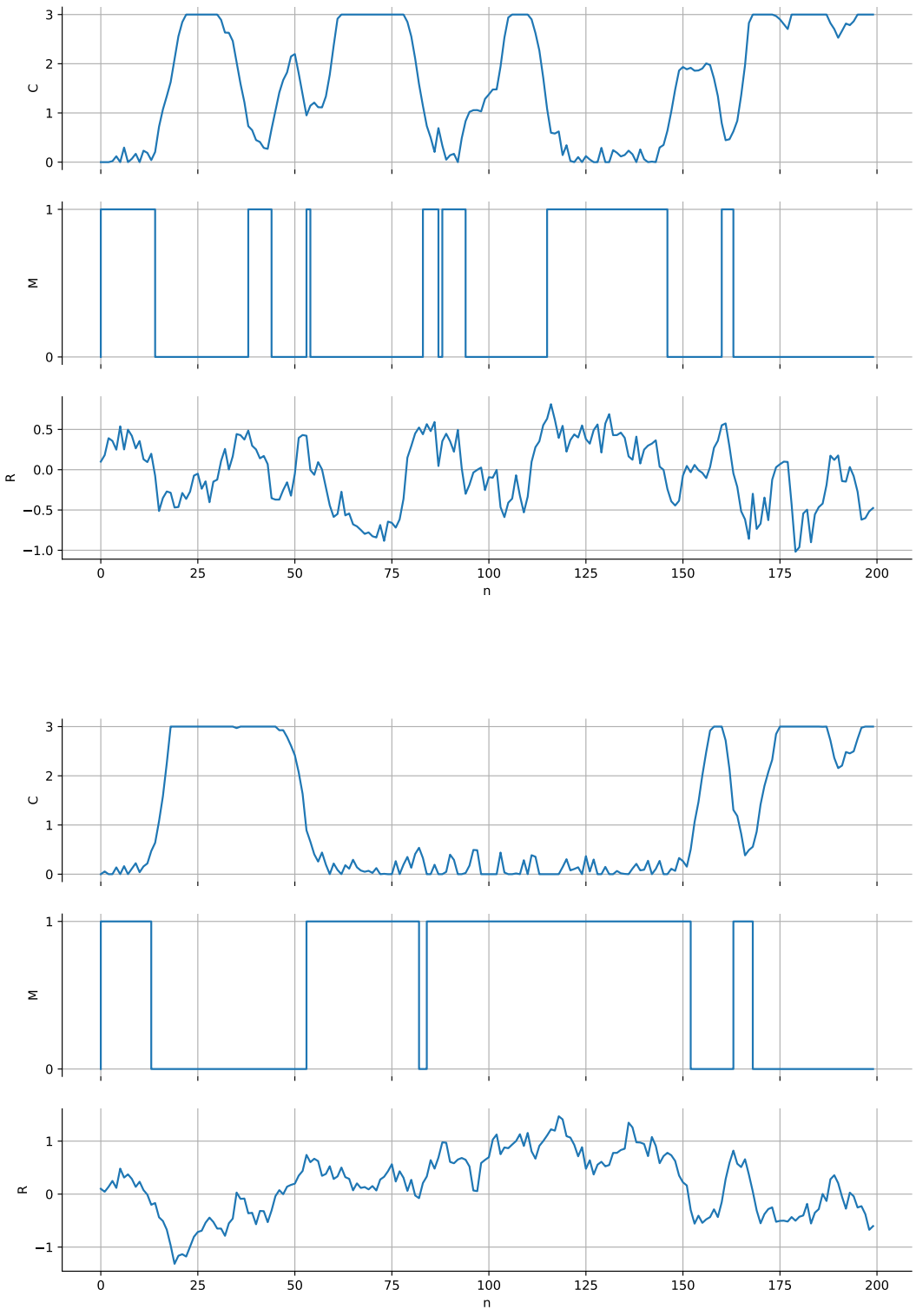

Figure 9: Two simulations of (C, M, R) optimally controlled using the Qknn algorithm, with $N = 200$ and $C_{\max} = 4$.

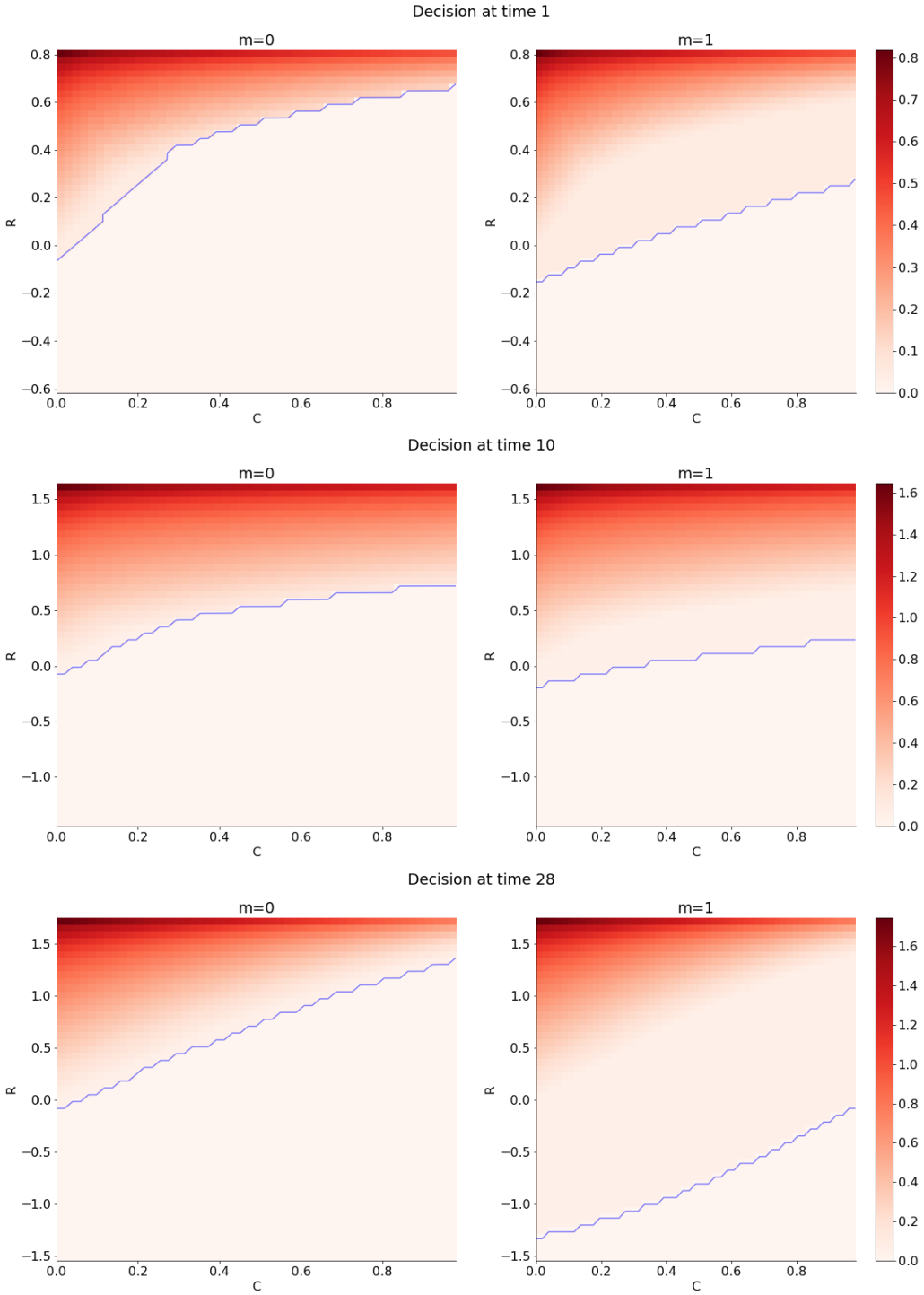

Figure 10: Estimated optimal decisions at time 1, 10 and 28, using the Qknn algorithm, with $N = 30$ time steps. The region under the blue line is the one where it is optimal to turn the generator off if m=1 (i.e. the generator was on at time n-1), or keep it off if $m = 0$ (i.e. the generator was off at time n-1).

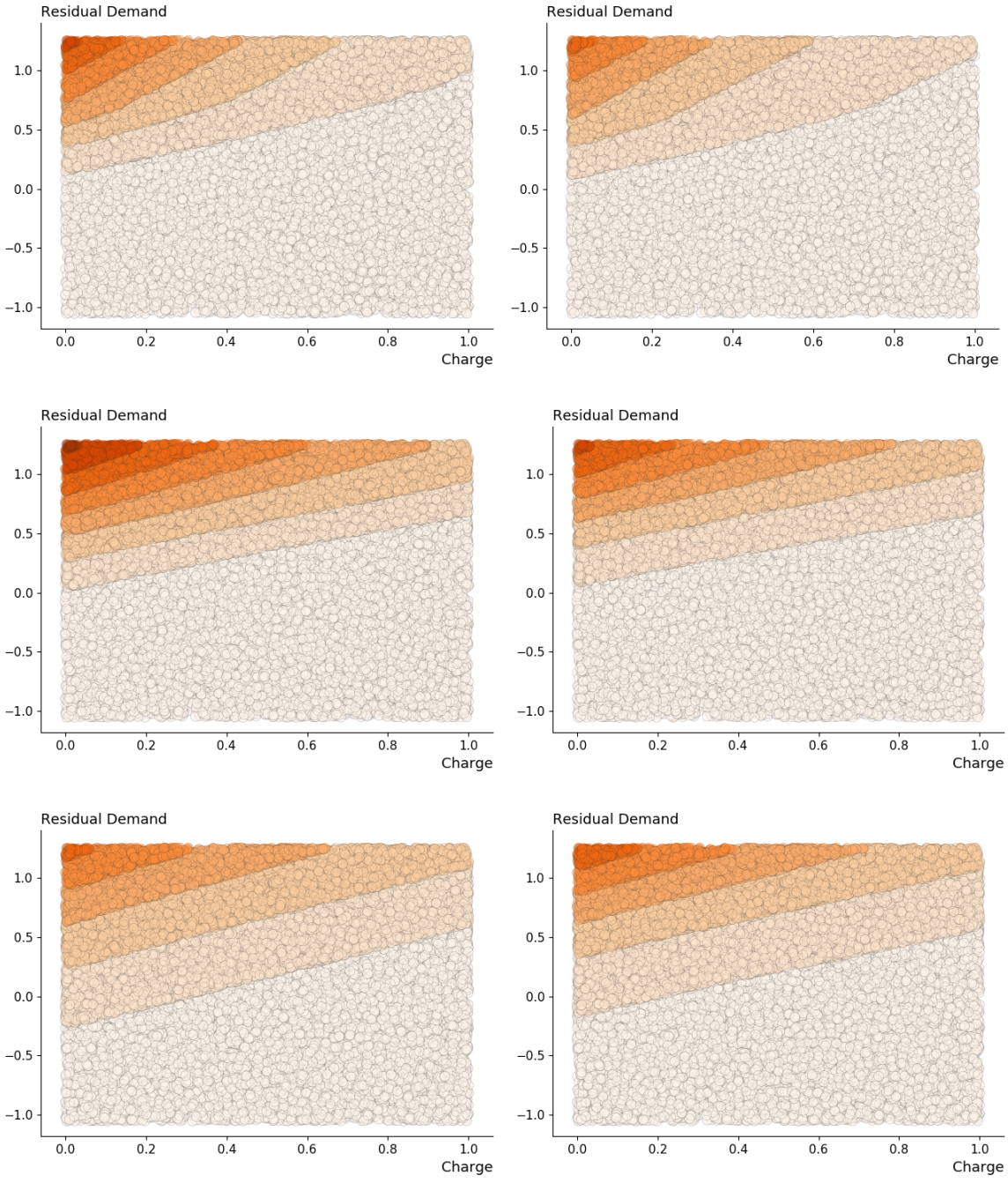

Figure 11: Estimated optimal decisions at time 1, 10 and 28, using the NN-based algorithm, with $N = 30$ time steps. The graphs on the left (resp. right) correspond to $m = 0$ (resp. $m = 1$). The control intensity is divided into intervals of length 0.33, where the lightest region corresponds to control taking values in $[0, 0.33]$.

References

[1] Clemence Alasseur, Alessandro Balata, Sahar Ben Aziza, Aditya Maheshwari, Peter Tankov, and Xavier Warin. Regression Monte Carlo for microgrid management. *arXiv:1802.10352v1*, 2018.

[2] Alessandro Balata, Côme Huré, Mathieu Laurière, Huyên Pham, and Isaque Pimentel. A class of finite-dimensional numerically solvable McKean-Vlasov control problems. *arXiv:1803.00445v2, to appear in ESAIM Proceedings and survey*, 2018.

[3] Dimitris Bertsimas, Leonid Kogan, and Andrew W. Lo. Hedging derivative securities and incomplete markets: an ε-arbitrage approach. *Operations Research*, 49(3):372–397, 2001.

[4] René Carmona and Mike Ludkovski. Valuation of energy storage: an optimal switching approach. *Quantitative Finance*, 26(1):262–304, 2010.

[5] Jean-Francois Chassagneux and Adrien Richou. Numerical simulation of quadratic BSDEs. *The Annals of Applied Probabilities*, 26(1):262–304, 2016.

[6] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics 5*, 5:349–380, 2017.

[7] Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes. Proceedings of the 35^{th} International Conference on Machine Learning, 2018.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[9] Pierre Henry-Labordere. Deep primal-dual algorithm for BSDEs: Applications of machine learning to CVA and IM. *SSRN:3071506*, 2017.

[10] Benjamin Heymann, J. Frédéric Bonnans, Pierre Martinon, Francisco J. Silva, Fernando Lanas, and Guillermo Jiménez-Estévez. Continuous optimal control approaches to microgrid energy management. *Energy Systems*, 9(1):59–77, 2018.

[11] Côme Huré, Huyên Pham, Achref Bachouch, and Nicolas Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis. 2018.

[12] Daniel R. Jiang and Warren B. Powell. An approximate dynamic programming algorithm for monotone value functions. *Operations Research*, 63(6):1489–1511, 2015.

[13] Adrien Richou. *Etude théorique et numérique des équations différentielles stochastiques rétrogrades*. PhD thesis, Université de Rennes 1, 2010.

[14] Adrien Richou. Numerical simulation of BSDEs with drivers of quadratic growth. *The Annals of Applied Probability*, 21(5):1933–1964, 2011.

Finance for Energy Market Research Centre

Institut de Finance de Dauphine, Université Paris-Dauphine

1 place du Maréchal de Lattre de Tassigny

75775 PARIS Cedex 16

www.fime-lab.org