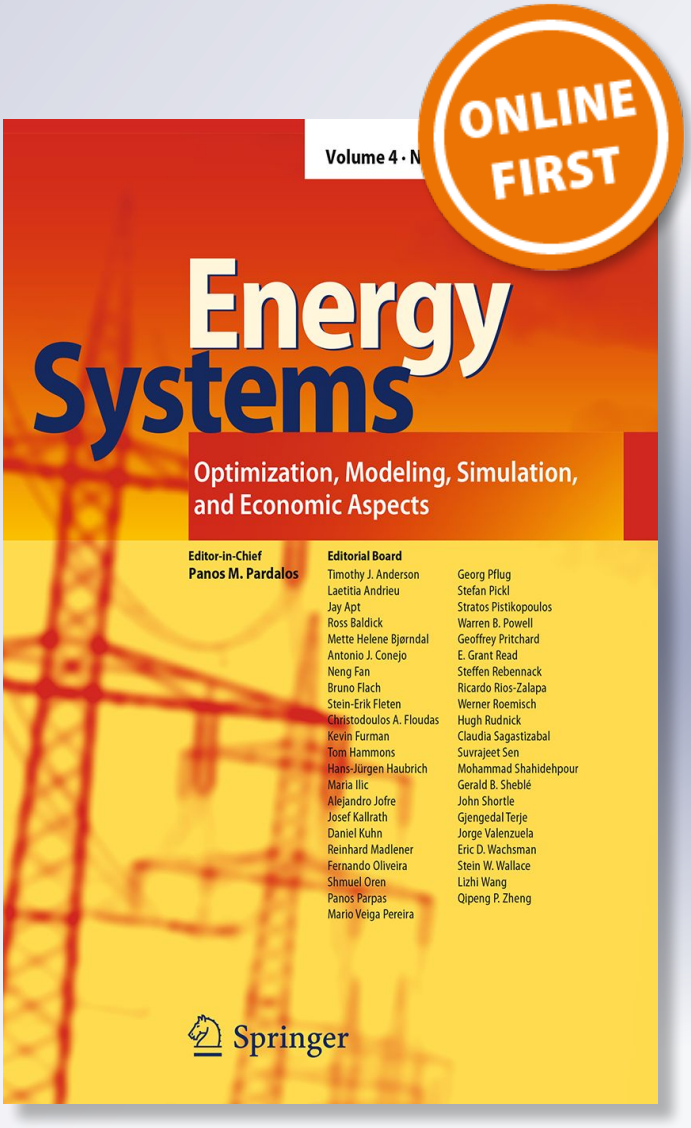
Simulation methods for stochastic storage problems: a statistical learning perspective

Michael Ludkovski & Aditya Maheshwari

Energy Systems
Optimization, Modeling, Simulation, and Economic Aspects

ISSN 1868-3967

Energy Syst
DOI 10.1007/s12667-018-0318-4



Springer

Energy Systems
https://doi.org/10.1007/s12667-018-0318-4

ORIGINAL PAPER

Simulation methods for stochastic storage problems: a statistical learning perspective

Michael Ludkovski[1] · **Aditya Maheshwari**[1]

Received: 29 March 2018 / Accepted: 7 December 2018

Abstract
We consider solution of stochastic storage problems through regression Monte Carlo methods. Taking a statistical learning perspective, we develop the dynamic emulation algorithm (DEA) that unifies the different existing approaches in a single modular template. We then investigate the two central aspects of regression architecture and experimental design that constitute DEA. For the regression piece, we discuss various non-parametric approaches, in particular introducing the use of Gaussian process regression in the context of stochastic storage. For simulation design, we compare the performance of traditional design (grid discretization), against space-filling, and several adaptive alternatives. The overall DEA template is illustrated with multiple examples drawing from natural gas storage valuation and optimal control of back-up generator in a microgrid.

Keywords Regression Monte Carlo · Simulation design · Gaussian process regression · Natural gas storage · Microgrid control

1 Introduction

Stochastic storage problems concern the optimal use of a limited inventory capacity under uncertainty, motivated by models from commodity management, energy supply, operations research and supply chains. The common thread in all these settings is deciding how to optimally add and reduce inventory as the system state stochastically fluctuates over time. For example, in the gas storage version [5,7,8,10,11,30,31,36,

This work is partially supported by NSF DMS-1736439.

✉ Aditya Maheshwari
maditya0310@gmail.com

Michael Ludkovski
ludkovski@pstat.ucsb.edu

[1] Department of Statistics and Applied Probability, South Hall, University of California, Santa Barbara, CA 93106, USA

Published online: 03 January 2019

38,39], the objective is to manage an underground cavern through buying and selling natural gas, with the principal stochastic factor being the commodity price. In the microgrid context [20–22], the objective is to deliver electricity at the lowest cost by maximizing inter-temporal storage linked to a renewable intermittent power source (say from solar or wind), so as to minimize the use of non-renewable backup generator. In the hydropower pumped storage setup, the objective is to match upstream inflows and downstream energy demands at the lowest cost [1,13,41].

To study a stochastic storage problem, one postulates the dynamics of the stochastic risk factors and defines the objective functional. The solution is then obtained by Dynamic Programming, reducing the global optimization into an iterative sequence of local problems, solved via backward recursion in time. There are two main approaches used in the literature to handle these control problems: probabilistic simulation based methods [5,7,8,10,14,17,31,39] and partial differential equations (PDE) [11,35].

In this study, we focus on simulation strategies, specifically the regression Monte Carlo (RMC) framework. Introduced for American option pricing [28,37], RMC was adapted to storage problems in [4,7,10,12,14,32]. The main complication is the presence of the endogenous inventory variable whose trajectory depends on the control. Since the storage problem is solved backward in time and the controller does not know which states will be reached, the original path-based Longstaff–Schwartz strategy in [28] is not feasible. To overcome this hurdle, Carmona and Ludkovski [10] suggested to back-propagate the inventory trajectories backward in time. This direction was further explored in [4,14]. A different approach is to view the overall storage problem as a continuum collection of switching problems, indexed by the current level of inventory I_t. One may then reduce to a finite number of 1D switching problems by discretizing the inventory I, which are then solved in parallel, employing interpolation in the I-direction [7,31].

An alternative is to view the stochastic risk factor P and the inventory I jointly in a multivariate setup, either globally [4,31], or locally [9,39] by partitioning the (P, I) domain into sub-domains. Langrene et al. [26,27] extended this approach by incorporating the control c into the state in order to handle problems where inventory dynamics feature further exogenous shocks. To deal with the unknown state distribution, the authors suggested multiple iterations of the backward procedure to improve the quality of the solution.

Beyond RMC, other Monte Carlo-based alternatives include the recent proposal in Van-Ackooij and Warin [38] to apply Stochastic Dual Dynamic Programming and the Markov Decision Process approach using binomial or multinomial trees [5,17].

The RMC valuation algorithms can be divided into three sub-problems:

- Approximation of the conditional expectation/continuation value.
- Evaluation of the optimal control.
- Estimation of the pathwise value.

In the storage setup most models consider discrete storage regimes, so that the optimal control is simply the maximizer (obtained generally via brute force comparison) of the continuation value. As for the conditional expectation, within the umbrella of RMC, there are three main methods popular in the literature to approximate it: Regress Now Monte Carlo (RNMC) [7,8,10,28,31,37,39], Regress Later Monte Carlo (RLMC) [4,

12,24,25,32] and Control Randomization (CR) [26,27]. Computation of the pathwise values can be based on 1-step look-ahead (the so-called TvR scheme introduced in [37]), w-step look-ahead utilizing the partial trajectories as proposed by Egloff in a series of papers [15,16] or a full $[t, T]$-trajectory as introduced by Longstaff and Schwartz [28].

Contributions

In this article we present a unified treatment of stochastic storage problems from the statistical learning perspective. We recast simulation-based dynamic programming approaches as an iterative sequence of machine learning tasks, corresponding to approximating the value functions, indexed by the time-step parameter t. Equivalently, the tasks can be thought of as learning the underlying q-values, or the optimal feedback control c_t.

With the machine learning perspective, the storage model is viewed as a stochastic simulator that produces noisy pathwise observations, and the aim is to recover the latent *average* behavior, i.e. the conditional expectation, by judiciously selecting which simulations to run. This framework naturally emphasizes computational complexity and offers an abstract modular template that accommodates a variety of approximation techniques. Indeed, our template involves three major pieces: (i) experimental design to determine which simulations to run; (ii) approximation technique for the conditional expectation; (iii) optimization step to recover optimal feedback control. While these sub-problems have been treated variously elsewhere, to our knowledge we are the first ones to fully modularize and distill them in this context. In particular, emphasizing the design aspect of RMC methods was only recently taken up in [18,29]. We borrow the design framework introduced by the first author in [29] for American options, and to our knowledge, this is the first article to explore experimental designs in context of storage problems.

We show that existing proposals for Regression Monte Carlo for storage problems all fit neatly into this template, and moreover our setup furnishes a variety of further improvements. Specifically, we address the following three enhancements:

1. New simulation designs. To better explore the input space we consider space-filling designs based on quasi Monte Carlo sequences or Latin hypercube sampling. To focus the emulators on the regions of interest, we consider probabilistic design. We also explore various adaptive versions, such as mixtures of space-filling and probabilistic designs, and designs that vary across t;
2. Non-parametric regression architectures for learning the value function. In particular, we document the advantages of using Gaussian Process regression;
3. Implementations that vary RMC ingredients across time-steps, either deterministically or adaptively, during the backward recursion. This includes alternating between joint-(P, I) and discretized-inventory regression schemes, as well as changing the design size as t changes. We showcase one such approach, addressing a non-smooth terminal condition.

We emphasize the mixing-and-matching aspect, which is especially attractive for implementation in a broad-scope software library where the different methods are expressed as subroutines accessible via a uniform interface.

The developed Dynamic Emulation Algorithm is applicable in a wide range of stochastic storage settings, being scalable in the dimension of the state variable and potential state dynamics. We illustrate DEA with four different extended case-studies. The first three case-studies consider valuation of natural gas storage facilities, starting from a standard benchmark first introduced in Forsyth and Chen [11]. This benchmark is then extended to add switching costs (that make control regime part of the state), and to consider simultaneously optimizing two storage facilities (leading to a 3D state space). Last but not least, we consider an example from microgrid management, solving for the optimal dispatch of backup generator to balance an intermittent renewable power source coupled to a limited battery.

Our developments parallel the recent literature on *optimal stopping*, especially in the context of Bermudan option pricing, where a wealth of strategies have been proposed and investigated [29]. It has been a well known folklore result that storage problems, especially with discrete controls, are "essentially" optimal stopping as far as computational methods are concerned. Nevertheless, the respective knowledge transfer is non-trivial and there remain substantive gaps, which we address herein, between the respective numerical algorithms. Thus, the present article "lifts" optimal stopping techniques to the setting of stochastic storage (i.e. optimal switching), and can be seen as a step towards similar treatment of further stochastic control problems, such as optimal impulse, or continuous control.

The rest of the paper is organized as follows. Section 2 describes the classical storage problem and the key ingredients to its solution. Section 3 describes the algorithm developed to modularize the solution steps for regression Monte Carlo into a sequence of statistical learning tasks. Section 4 discus the mathematics for Gaussian process regression and other popular regression methods used in the literature in the context of storage problems. In Sect. 5, we introduce different design alternates such as space-filling, adaptive and dynamic to exploit the spatial information and efficiently implement non-parametric regression methods (particularly, Gaussian process regression) utilizing batched design. Sections 6 and 7 are devoted to numerical illustrations, taking up the gas storage and microgrid management, respectively. Finally, Section 8 concludes.

2 Problem description

A storage problem is exemplified by the presence of *stochastic risk factors* together with an inventory state variable. The risk factors have autonomous dynamics, while the inventory is (fully) controlled by the operator via the storage policy; thus the latter dynamics are endogenized. A second feature of storage problems we consider is their *switching* property: the controller actions consist of directly toggling the storage *regime*. In turn the storage regime drives the dynamics of the inventory. Depending on the setup, the regime is either a control variable, or a part of the system state.

To make our presentation concrete, we focus on the classical storage problem with a stochastic price. Namely, there are two main state variables: P_t and I_t. $P_t \in \mathbb{R}_+$ represents the price of the stored commodity; $I_t \in [0, I_{\max}]$ is the inventory level. We present the storage dynamics in discrete-time on a time interval $[0, T]$ discretized into a finite grid $0 = t_0 < \ldots t_k \ldots < t_K = T$, such that $t_k = k\Delta t = k\frac{T}{K}$.

For the price, we assume exogenous Markovian dynamics of the form

$$P_{t_{k+1}} = P_{t_k} + b(t_k, P_{t_k})\Delta t + \sigma(t_k, P_{t_k})\,\Delta W_{t_k}, \tag{1}$$

where (ΔW_t) are exogenous i.i.d. stochastic shocks. For the rest of the article we take $\Delta W_{t_k} \sim \mathcal{N}(0, \sqrt{\Delta t})$ representing Brownian motion dynamics; any other (time-dependent) shocks could be straightforwardly utilized too. We denote by $\mathcal{F}_{t_k}$ the σ-algebra generated by price process up until time t_k and by $\mathbb{F} = (\mathcal{F}_{t_k})$ the corresponding filtration. The inventory level I_t follows,

$$I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\,\Delta t, \tag{2}$$

where c_{t_k} is the inventory control, representing the rate of storage injection $c > 0$, withdrawal $c < 0$ or holding $c = 0$. The control is linked to the storage regime m_{t_k}. We assume that there are three regimes $m_{t_k} \in \mathcal{J} := \{+1, -1, 0\}$ representing injection, withdrawal and do-nothing respectively. The regime and control are determined by the joint state:

$$m_{t_{k+1}} = \mathcal{M}(t_k, P_{t_k}, I_{t_k}, m_{t_k}), \tag{3}$$

$$c_{t_k}(m_{t_{k+1}}) = \mathcal{C}(t_k, P_{t_k}, I_{t_k}, m_{t_k}; m_{t_{k+1}}). \tag{4}$$

Note that the above form implies that at each time-step t_k and state $(P_{t_k}, I_{t_k}, m_{t_k})$ the controller picks her next regime $m_{t_{k+1}}$ which in turn determines her control $c_{t_k}(m_{t_{k+1}})$. It also directly restricts controls to be of Markovian feedback form, making the policies $(c_{t_k}, m_{t_{k+1}})$ $(\mathcal{F}_{t_k})$-adapted. As a result, $(P_{t_k}, I_{t_k}, m_{t_k})$ is a Markov process, adapted to the price filtration $(\mathcal{F}_{t_k})$.

Let $\pi(P, c)$ be the instantaneous profit rate earned by using control c when price is P, and $K(i, j) \geq 0$ be the switching cost for switching from regime i to j. Then

$$\pi^{\Delta}(P_{t_k}, m_{t_k}, m_{t_{k+1}}) := \pi(P_{t_k}, c_{t_k}(m_{t_{k+1}}))\Delta t - K(m_{t_k}, m_{t_{k+1}}),$$

is the net profit earned during one time-step $[t_k, t_{k+1})$. To denote the cumulative profit of the controller on $[t_k, T]$ along the path specified by $\mathbf{P}_{t_k} = P_{t_k:t_K}$ and a selected sequence of regimes $\mathbf{m}_{t_k} := (m_{t_k:t_K})$ (and consequently the control $\mathbf{c}_{t_k} := (c_{t_k:t_K})$) we use

$$v(t_k, \mathbf{P}_{t_k}, I_{t_k}, \mathbf{m}_{t_k}) := \sum_{s=k}^{K-1} e^{-r(t_s - t_k)}\pi^{\Delta}(P_{t_s}, m_{t_s}, m_{t_{s+1}}) + e^{-r(T-t_k)}W(P_T, I_T), \tag{5}$$

where $r \geq 0$ is the discount rate and $W(P, I)$ is the terminal condition (typically concerning the final inventory I) at the contract expiration. Note that I_T is determined

recursively based on $\mathbf{m}_{t_k}$ using (2). Similarly, because $v(\cdot)$ depends on the initial regime m_{t_k} through the switching costs $K(m_{t_k}, m_{t_{k+1}})$, m_{t_k} is part of the current state, impacting the next decision to be made, cf. Fig. 7 in Sect. 6.4. The goal of the controller is to maximize discounted expected profits on the horizon $[t_k, T]$,

$$V(t_k, P, I, m) = \sup_{\mathbf{m}_{t_k}} \mathbb{E}\left[v(t_k, \mathbf{P}_{t_k}, I_{t_k}, \mathbf{m}_{t_k}) \middle| P_{t_k} = P, I_{t_k} = I, m_{t_k} = m \right], \tag{6}$$

$$\text{subject to } \; I_{t_s} \in [I_{min}, I_{max}] \quad \forall s. \tag{7}$$

Problem (6) belongs to the class of stochastic optimal control, and satisfies the dynamic programming principle (DPP, also known as the Bellman equation) [33]. The DPP implies that $V(t_k, \cdot)$ satisfies the one-step recursion

$$V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) = \max_{m \in \mathcal{J}} \mathbb{E}\left[\pi^{\Delta}(P_{t_k}, m_{t_k}, m) + e^{-r\Delta t} V(t_{k+1}, P_{t_{k+1}}, I_{t_{k+1}}, m) \middle| P_{t_k} \right], \tag{8}$$

where the expectation is over the random variable $P_{t_{k+1}}$ since the inventory $I_{t_{k+1}}$ is fully determined by I_{t_k} and the chosen regime m on $[t_k, t_{k+1})$. See, [2,4,10] for further discussion and proof of DPP in related storage problems. Note that due to the inventory constraints, some of the controls might not be admissible for different initial conditions, so that formally the maximum in (8) is over $\mathcal{J} = \mathcal{J}(t_k, P_{t_k}, I_{t_k}, m_{t_k}) \subseteq \{+1, -1, 0\}$. For instance, if the inventory is zero, $I_{t_k} = 0$ further withdrawal is ruled out and $\mathcal{J}(t_k, P_{t_k}, 0, m_{t_k}) = \{0, 1\}$. Such constraints could even be time- or price-dependent, for instance in hydropower management.

Remark 1 In our main setup the choice of the control is pre-determined given the stochastic state and the regime. More generally, conditional on the regime there might be a set of admissible controls $\mathcal{A}(t_k, m_{t_{k+1}})$, adding an additional optimization substep. For example, we might have $\mathcal{A}(t_k, +1) = (0, c_{\max}(I_{t_k})]$ and $\mathcal{A}(t_k, -1) = [c_{\min}(I_{t_k}), 0)$, where $c_{max} > 0$ is the maximum injection rate and $c_{min} < 0$ is the largest withdrawal rate (i.e., minimal, negative injection rate). When $|\mathcal{A}| > 1$, the optimization problem (8) requires first to find the optimal regime $m_{t_{k+1}}$, and secondly to find the optimal control $c_{t_k}(m_{t_{k+1}})$ admissible to this regime. The original formulation corresponds to $\mathcal{A}$ being a singleton and can be interpreted as a trivial optimization over $\mathcal{A}$, e.g. due to a bang-bang structure. Abstractly, we may always write $c_{t_k}(m_{t_{k+1}}) = \mathcal{C}(t_k, P_{t_k}, I_{t_k}, m_{t_k}; m_{t_{k+1}})$ subsuming the inner optimizer.

2.1 Solution structure

Due to the Markovian structure, at each time-step t_k and state $(P_{t_k}, I_{t_k}, m_{t_k})$ the controller picks her next regime $m_{t_{k+1}}$ (consequently control $c_{t_k}(m_{t_{k+1}})$) according to

$$\begin{aligned} m_{t_{k+1}} &= m^*(t_k, P, I, m) \\ &= \arg\max_{j \in \mathcal{J}} \left\{ \pi^{\Delta}(P, m, j) + e^{-r\Delta t} q(t_k, P, I + a(c_{t_k}(j))\Delta t, j) \right\}, \end{aligned} \tag{9}$$

where the continuation value of switching to regime m is

$$q(t_k, P, I, m) := \mathbb{E}\left[V(t_{k+1}, P_{t_{k+1}}, I, m) \middle| P_{t_k} = P\right]. \tag{10}$$

Conceptually, we have a map from the value function V to m^* and c^*, encoded as $m^* : (t, P, I, m) \mapsto \mathcal{J}$. The dependence of the continuation value q, value function V and the control map m^* on the current regime m is a consequence of the switching costs $K(m_{t_k}, m_{t_{k+1}})$, see Remark 4.

Conversely, Eq. (6) provides the representation of the value function as a conditional expectation of future profits based on an optimal policy $\mathbf{m}^*$. Therefore, any estimate $\widehat{m} : (t, P, I, m) \mapsto \mathcal{J}$ of the control map naturally induces a corresponding estimate $\hat{V}$ of the value function. Specifically, $\widehat{m}$ yields the dynamics

$$\hat{I}_{t_{k+1}} = \hat{I}_{t_k} + a\big(c_{t_k}(\widehat{m}_{t_{k+1}}(t, P, I, \widehat{m}_{t_k}))\big)\Delta t,$$

and induces

$$\hat{V}(0, P_0, I_0, m_0) = \mathbb{E}\left[\sum_{k=0}^{K-1} e^{-rt_k}\pi^{\Delta}(P_{t_k}, \widehat{m}_{t_k}, \widehat{m}_{t_{k+1}}) + e^{-rT}W(P_T, \hat{I}_T)\right]. \tag{11}$$

While $\hat{I}_{t_k}$ does not appear explicitly above, it is crucially driving $\widehat{m}_{t_{k+1}}(t, P_{t_k}, \hat{I}_{t_k}, \widehat{m}_{t_k})$.

Figure 1 illustrates this dual link by showing a trajectory of (P_t) and several corresponding trajectories of $(\hat{I}_t)$ indexed by their initial inventories I_0 (viewed as an external parameter) for a gas storage facility (see Sect. 6.1 for more details). One interesting observation is that the dependence of time-t inventory $\hat{I}_t$ on $I_0 = i$ is rather weak, i.e. the inventory levels coalesce: $\hat{I}_t^i = \hat{I}_t^{i'}$ after an initial "transient" time period. Note that because the controls are specified in feedback form, once $\hat{I}_t^i = \hat{I}_t^{i'}$ we have $\widehat{m}_s^i = \widehat{m}_s^{i'}$ for all $s \geq t$ and the inventory paths will stay together forever. The Figure also illustrates the underlying maxim of "buy low, sell high": when P_t is low, controlled inventory $\hat{I}_t$ is high (and increasing), and when P_t is high, $\hat{I}_t$ is low (and shrinking). As a result, we see a clustering of $\hat{I}_t$ around the minimum and maximum storage levels I_{min}, I_{max}, indicating the strong constraint imposed by the bounded storage capacity.

To visualize the estimated optimal policy $\widehat{m}$, Fig. 2a plots the control map $(P, I) \mapsto \widehat{m}(t, P, I)$ at a fixed time step t, namely with $T - t = 0.3$ years for the gas storage example of Sect. 6.1. (In that example, there is no dependence on current regime m.) The state space is divided into three regions: when P is high it is optimal to withdraw: $\widehat{m} = -1$; if P is low, it is optimal to inject $\widehat{m} = +1$; in the middle, or if inventory is very large, it is optimal to do nothing $\widehat{m} = 0$. Typically, the control map is interpreted by fixing current inventory I and looking at $\widehat{m}$ as a function of P. We can then summarize the resulting policy in terms of the *injection/withdrawal boundaries* $B_{inj}(I, t)$ and $B_{wdr}(I, t)$:

$$\begin{cases} B_{inj}(I, t) := \sup\{P : \widehat{m}(t, P, I) = +1\}, \\ B_{wdr}(I, t) := \inf\{P : \widehat{m}(t, P, I) = -1\}. \end{cases} \tag{12}$$

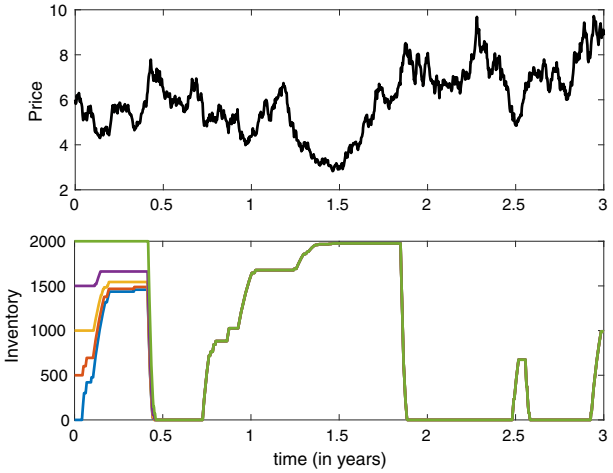


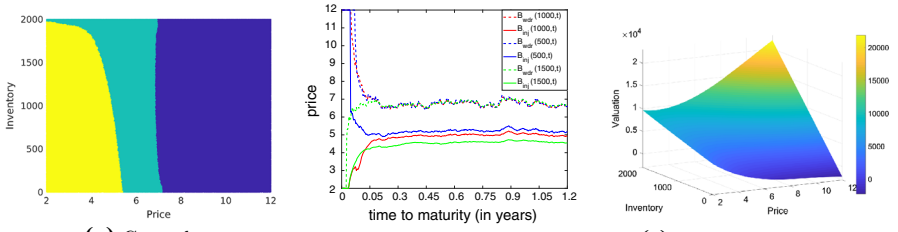
Fig. 1 *Top panel:* a given trajectory of commodity price (P_t) following logarithmic mean reverting process in (27). *Lower panel:* corresponding trajectories of controlled inventory $\hat{I}_t$ starting at $\hat{I}_0 \in \{0, 500, 1000, 2000\}$. This figure is associated with the gas storage example of Sect. 6.1 using the PR-1D solution scheme

Since injection becomes profitable with low prices, $B_{inj}(I,t)$ represents the maximum price for which injection $(\widehat{m}(t,P_t,I) = +1)$ is the optimal policy. Similarly, $B_{wdr}(I,t)$ represents the minimum price for which withdrawal $(\widehat{m}(t,P_t,I) = -1)$ is the optimal policy. The interval $[B_{inj}(I,t), B_{wdr}(I,t)]$ is the no-action region. These boundaries are plotted as a function of $T-t$ in Fig. 2b at three different inventory levels. One prominent feature is the boundary layer as $T-t \to 0$ whereby the policy is primarily driven by the terminal penalty $W(P,I)$ than immediate profit considerations. In this example the "hockey-stick" $W(P_T, I_T)$ forces the controller to target the inventory level $I = 1000$, as $T-t \to 0$, so that injection becomes the optimal policy for $I < 1000$ independent of the price, and withdrawal becomes optimal for $I > 1000$ (the no-action region effectively disappears). Conversely, for large $T-t$, the boundaries $t \mapsto B_{inj}(I,t)$ and $t \mapsto B_{wdr}(I,t)$ are essentially time-stationary.

Finally, Fig. 2c shows the value function $\hat{V}(0,P,I)$ as a 2-D surface in the price and inventory coordinates. As noted by previous studies, for a fixed price P, we observe a linear relationship between value and inventory. However, as a function of P, $V(0,\cdot,I)$ is non-linearly decreasing at low inventory levels, and increasing for large inventory.

3 Dynamic emulation algorithm

The numerical algorithms we consider provide approximations, denoted as $\hat{q}$, to the continuation value in (10). This is done via backward induction to estimate $\hat{q}(t_k,\cdot)$ as the conditional expectation of $\hat{V}(t_{k+1},\cdot)$ for $k = K-1$ to $k = 0$. For this induction, the simulation-based framework relies on a Monte Carlo approximation which consists of generating *pathwise* profits $v_{k+1}(\mathbf{P}_{t_{k+1}}, \mathbf{I}_{t_{k+1}}, \mathbf{m}_{t_{k+1}})$ and then inferring the input-output relationship between $(P_{t_k}, I_{t_k}, m_{t_k})$ and $v_{k+1}(\cdot)$ via a statistical regression.



(a) Control map **(b)** Prod./Inject. boundaries **(c)** Value function

Fig. 2 *Left panel:* snapshot of the control map $\widehat{m}(t, P, I)$ at $t = 2.7$ years. *Middle:* injection $B_{inj}(I, \cdot)$ and withdrawal $B_{wdr}(I, \cdot)$ boundaries as a function of time for $I \in \{500, 1000, 1500\}$. *Right:* value function $\hat{V}(0, P, I)$ at $t = 0$. Results were obtained with conventional design and PR-1D regression

Indeed, a conditional expectation, which is an L^2-projection, is naturally approximated as learning the mean response with a squared-loss criterion. Specifically, regression is implemented as empirical projection onto an approximation space $\mathcal{H}_k$.

Because these pathwise simulations of $P_{t_k} \rightarrow P_{t_{k+1}}$ are themselves part of the algorithm, we must also choose the respective experimental design $\mathcal{D}_k$, i.e. the inputs "$\mathbf{x}$" of the regression (with v_{k+1}'s being the corresponding "$\mathbf{y}$"). The core loop, which we dub Dynamic Emulation, thus consists of the following sequence of learning tasks that must be done at each time-step t:

$$\text{Generate design} \rightarrow \text{Generate pathwise profits} \rightarrow \text{Estimate the continuation function } \hat{q}.$$

The nomenclature of the Dynamic Emulation algorithm emphasizes the associated recursive learning that can be dynamically adjusted. One non-standard feature is that while at each step DEA targets the emulation of the continuation value $q(t, \cdot)$, the overall performance measure is given by the quality of the final answer $\hat{V}(0, \cdot, \cdot, \cdot)$ on an out-of-sample simulations utilizing the estimates of the continuation value $q(t, \cdot)$.

The above perspective distills the following key properties of the RMC framework:

- Picking the simulation designs $\mathcal{D}_k$ underlying the emulation.
- Generating the pathwise profits v_{k+1} underlying the regression.
- Picking the projection spaces $\mathcal{H}_k$ for the regression.
- Modularity in terms of t, which allows dynamic selections for the above sub-steps as the backward recursion unfolds.

The resulting template for solving the storage problem brings multiple advantages. First, DEA offers a wide latitude in selecting the approximation space $\mathcal{H}_k$. Existing schemes largely concentrated on parametric regression with a fixed set of basis functions. In contrast, DEA allows for any number of regression frameworks, including non-parametric approaches that are more expressive. It also includes the possibility to (adaptively) pick different $\mathcal{H}_k$ across timesteps t_k. Second, DEA eliminates the requirement to store global price paths in memory, rather all simulations are performed "online". Third, DEA introduces arbitrary experimental designs; since the latter is primal to maximizing the learning efficiency, there are significant gains to be exploited from judiciously selecting the shape of $\mathcal{D}_k$. Heretofore the literature concentrated on

what we call "probabilistic" designs where the values of P_{t_k} came from pre-computed price paths. Fourth, DEA allows to vary the number of simulations N_k at time step t_k.

The DEA is based on the concept of stochastic simulation. Assuming that we have estimated the continuation function $\hat{q}(t_{k+1}, \cdot), \ldots, \hat{q}(t_{K-1}, \cdot)$, the objective is to estimate $\hat{q}(t_k, \cdot)$ using a simulation design $\mathcal{D}_k := (P^n_{t_k}, I^n_{t_{k+1}}, m^n_{t_{k+1}}, n = 1, \ldots, N_k)$. Note that for conceptual clarity we re-label the current state via the next-step $I_{t_{k+1}}$ which is deterministic based on $I_{t_k}, m_{t_{k+1}}$. The simulator first generates one-step paths $P^n_{t_k} \mapsto P^n_{t_{k+1}}, n = 1, \ldots, N_k$. It then computes the one-step-ahead pathwise profits (cf. [37])

$$v^n_{k+1} := \max_{j \in \mathcal{J}} \left\{ \pi^\Delta(P^n_{t_{k+1}}, m^n_{t_{k+1}}, j) + e^{-r\Delta t} \hat{q}(t_{k+1}, P^n_{t_{k+1}}, I^n_{t_{k+1}} + a(c(j))\Delta t, j) \right\} \tag{13}$$

$$= \pi^\Delta(P^n_{t_{k+1}}, m^n_{t_{k+1}}, m^n_{t_{k+2}}) + e^{-r\Delta t} \hat{q}(t_{k+1}, P^n_{t_{k+1}}, I^n_{t_{k+2}}, m^n_{t_{k+2}}), \tag{14}$$

along each path $n = 1, \ldots, N_k$ where

$$m^n_{t_{k+2}} = \arg\max_{j \in \mathcal{J}} \left\{ \pi^\Delta(P^n_{t_{k+1}}, m^n_{t_{k+1}}, j) + e^{-r\Delta t} \hat{q}(t_{k+1}, P^n_{t_{k+1}}, I^n_{t_{k+1}} + a(c(j))\Delta t, j) \right\}$$
$$I^n_{t_{k+2}} = I^n_{t_{k+1}} + a(c(m^n_{t_{k+2}}))\Delta t.$$

Observe that while (13) is based on the definition of (10) in terms of the value function, $\hat{V}$ actually never makes an appearance, and v_{k+1} is defined solely from $\hat{q}(t_{k+1}, \cdot)$. The approximation task is then to learn (by fitting a statistical model) the input-output relationship between $(P_{t_k}, I_{t_{k+1}}, m_{t_{k+1}})^{N_k}_{n=1}$ and $(v_{k+1})^{N_k}_{n=1}$ to extract the expected response (i.e. the mathematical expectation of v_{k+1}). Note that to evaluate the term on the RHS of (13) we need to predict continuation values at arbitrary next-step states $\hat{q}(t_{k+1}, P_{t_{k+1}}, I_{t_{k+1}}, m_{t_{k+1}})$.The underlying operations of `fit` and `predict` are thus the two main workhorses of the statistical approximation procedure. In the `fit` step, we seek to L^2-project v_{k+1} onto an approximation space $\mathcal{H}_k$:

$$\check{q}(t_k, \cdot) := \arg\min_{h_{t_k} \in \mathcal{H}_k} \|h_{t_k} - v_{k+1}\|_2. \tag{15}$$

As a canonical setup, $\mathcal{H}_k = \text{span}(\phi_1, \ldots, \phi_R)$ is the linear space generated by basis functions ϕ_i and the approximation $\check{q}(t_k, \cdot) = \sum_{i=1}^R \beta_i \phi_i(\cdot)$ is described through its coefficient vector $\vec{\beta}$. To estimate $\vec{\beta}$ we solve a discrete optimization problem based on experimental design $\mathcal{D}_k$ of size N_k and the corresponding realized pathwise values $v^n_{k+1}, n = 1, \ldots, N_k$ from trajectories started at $(P^n_{t_k}, I^n_{t_{k+1}}, m^n_{t_{k+1}})$:

$$\hat{q}(t_k, \cdot, \cdot, \cdot) = \arg\min_{h_{t_k} \in \mathcal{H}_k} \sum_{n=1}^N \left| h_{t_k}(P^n_{t_k}, I^n_{t_{k+1}}, m^n_{t_{k+1}}) - v^n_{k+1} \right|^2. \tag{16}$$

Thus, $\hat{q}$ is an empirical approximation of the projection $\check{q}$, with the corresponding finite-sample error. In particular, subject to mild conditions on $\mathcal{D}_k$, we have $\hat{q} \to \check{q}$ as $N_k \to \infty$.

Since the regime $m_{t_{k+1}}$ is a discrete covariate, we repeat the above emulation sub-problem separately for each $m \in \mathcal{J}$. In particular, we select a separate simulation design for different regimes, labeled as $\mathcal{D}_{k,m}$. To emphasize the resulting link between the design and the pathwise simulation we then write $P_k^{n,\mathcal{D}_{k,m}} \mapsto P_{k+1}^{n,\mathcal{D}_{k,m}}$. Also note that both the design and the simulators can be carried out *on-the-fly* during the backward recursion; there is no need to do any pre-simulations.

To complete the description of the algorithm, it remains to address the initialization and post-processing steps. Recall that DEA runs backward over t_k. It is initialized with the known terminal condition

$$\hat{q}(t_K, P_{t_K}, I_{t_K}, m) = W(P_T, I_T) \quad \forall m.$$

The recursive construction then ensures that at step t_k we know the continuation functions $\hat{q}(t_k, \cdot, \cdot, \cdot)$ and hence can find $\widehat{m}(t_{k+1}, P, I, m)$ for any (P, I, m) in the state space as in (9), as well as

$$\hat{V}(t_k, P, I, m) = \pi^{\Delta}(P, m, \widehat{m}_{t_{k+1}}) + e^{-r\Delta t}\hat{q}(t_k, P, I + a(c(\widehat{m}_{t_k}))\Delta t, \widehat{m}(t_{k+1}, P, I, m)). \tag{17}$$

In the last step of DEA, we compute an *out-of-sample* estimate of the value function at $t_0 = 0, P_0, I_0, m_0$ by generating N' new paths $(\mathbf{P}_{0:T}^{n'}, \hat{\mathbf{I}}_{0:T}^{n'}, \widehat{\mathbf{m}}_{0:T}^{n'}), n' = 1, \ldots, N'$, where the pathwise inventory $\hat{I}$ is based on the just-estimated control map $\widehat{m}(t_{k+1}, \cdot, \cdot, \cdot)$ matching (9), and consequently the control $c(\widehat{m})$, cf. Fig. 1. Thus, $\hat{V}(0, P_0, I_0, m_0) = \frac{1}{N'}\sum_{n'=1}^{N'} v_{0:K}(0, \mathbf{P}_{0:T}^{n'}, \hat{\mathbf{I}}_{0:T}^{n'}, \widehat{\mathbf{m}}_{0:T}^{n'})$, where $v_{0:K}(0, \mathbf{P}_{0:T}^{n'}, \hat{\mathbf{I}}_{0:}T^{n'}, \widehat{\mathbf{m}}_{0:T}^{n'})$ is the total cumulative discounted profit over the K time-steps using $\widehat{m}$ for each sample path, cf. (11). Since the policy $\widehat{\mathbf{m}}$ is necessarily sub-optimal, $\hat{V}(0, \cdot)$ is a lower bound for the true V, modulo the Monte Carlo error from the N' trajectories used in the last averaging.

Algorithm 1 presents the overall template for solving the storage problem. Assuming K time-steps and a fixed design size of $|\mathcal{D}_k| = N \forall k$, its overall complexity is $\mathcal{O}(KN)$. It is purposely short and abstract, stressing the modular setup and the associated looping. Lines 1–3 initialize with the terminal condition; Line 6 is the emulation step, returning a fitted regression model for $\hat{q}(t_k, \cdot)$. Line 7 is the experimental design sub-step. Line 8 is the stochastic simulator which is combined with Lines 10–13 to create the one-step-ahead profits v_{k+1}. We conclude this section with several remarks. In the next two sections we then provide menus for the two principal steps in DEA: selecting the approximation space $\mathcal{H}_k$ and the design $\mathcal{D}_{k,m}$.

Remark 2 (Defining the Pathwise Profits). Above we view the continuation value as the conditional expectation of one-step-ahead value function, matching the original Tsitsiklis–van Roy [37] scheme. More generally, we can unroll the Dynamic Programming Eq. (8) using the optimal regime choices $m_{t_{k+1}}^*$ and corresponding controls $c_{t_k}^*$ for any $w \geq 1$ as

Algorithm 1: Dynamic Emulation Algorithm (DEA) - $\mathcal{O}(KN)$

Data: K (time steps), (N_k) (simulation budgets per step);

Generate design $\mathcal{D}_{K-1,m} := (\mathbf{P}_{K-1}^{\mathcal{D}_{K-1,m}}, \mathbf{I}_K^{\mathcal{D}_{K-1,m}})$ of size N_{K-1} for each $m \in \mathcal{J}$.

Generate one-step paths $P_{K-1}^{n,\mathcal{D}_{K-1,m}} \mapsto P_K^{n,\mathcal{D}_{K-1,m}}$ for $n = 1, \ldots, N_{K-1}$ and $m \in \mathcal{J}$

Terminal condition: $v_{K,m}^n \leftarrow W(P_K^{n,\mathcal{D}_{K-1,m}}, I_K^{n,\mathcal{D}_{K-1,m}})$ for $n = 1, \ldots, N_{K-1}$ and $m \in \mathcal{J}$

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

  for $m \in \mathcal{J}$ **do**

    Fit: $\hat{q}(k, \cdot, \cdot, m) \leftarrow \arg\min_{h_k \in \mathcal{H}_k} \sum_{n=1}^{N_k} |h_k(P_k^{n,\mathcal{D}_{k,m}}, I_{k+1}^{n,\mathcal{D}_{k,m}}) - v_{k+1,m}^n|^2$

    Generate design $\mathcal{D}_{k-1,m} := (\mathbf{P}_{k-1}^{\mathcal{D}_{k-1,m}}, \mathbf{I}_k^{\mathcal{D}_{k-1,m}})$ of size N_{k-1} for each $m \in \mathcal{J}$

    Generate one-step paths $P_{k-1}^{n,\mathcal{D}_{k-1,m}} \mapsto P_k^{n,\mathcal{D}_{k-1,m}}$ for $n = 1, \ldots, N_{k-1}$

  end

  for $n = 1, \ldots, N_{k-1}$ *and* $m \in \mathcal{J}$ **do**

    Predict: $m' \leftarrow$

    $\arg\max_{j \in \mathcal{J}} \{\pi^\Delta(P_k^{n,\mathcal{D}_{k-1,m}}, m, j) + e^{-r\Delta t} \hat{q}(k, P_k^{n,\mathcal{D}_{k-1,m}}, I_k^{n,\mathcal{D}_{k-1,m}} + a(c_k(j))\Delta t, j)\}$

    $v_{k,m}^n \leftarrow \pi^\Delta(P_k^{n,\mathcal{D}_{k-1,m}}, m, m') + e^{-r\Delta t} \hat{q}(k, P_k^{n,\mathcal{D}_{k-1,m}}, I_k^{n,\mathcal{D}_{k-1,m}} + a(c_k(m'))\Delta t, m')$

  end

end

return $\{\hat{q}(k, \cdot, \cdot, m)\}_{k=1, m \in \mathcal{J}}^{K-1}$

$$
\begin{aligned}
V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) &= \mathbb{E}[\pi^\Delta(P_{t_k}, m_{t_k}, m^*_{t_{k+1}}) + e^{-r\Delta t} q(t_k, P_{t_k}, I_{t_{k+1}}, m^*_{t_{k+1}}) \Big| P_{t_k}] \\
&= \mathbb{E}\Big[\pi^\Delta(P_{t_k}, m_{t_k}, m^*_{t_{k+1}}) + e^{-r\Delta t} \pi^\Delta(P_{t_{k+1}}, m^*_{t_{k+1}}, m^*_{t_{k+2}}) \\
&\quad + e^{-2r\Delta t} q(t_{k+1}, P_{t_{k+1}}, I^*_{t_{k+2}}, m^*_{t_{k+2}}) \Big| P_{t_k}\Big] \\
&\quad \cdots \\
&= \mathbb{E}\Big[v_{k:k+w}(\pi, q)(\mathbf{P}_{t_k}, \mathbf{I}_{t_k}, \mathbf{m}_{t_k}) \Big| P_{t_k}, I_{t_k}, m^*_{t_k}\Big] \qquad (18)
\end{aligned}
$$

in terms of the pathwise gains

$$
\begin{aligned}
v_{k:k+w}(\pi, q)(\mathbf{P}_{t_k}, \mathbf{I}_{t_k}, \mathbf{m}_{t_k}) &:= \sum_{s=k}^{k+w-1} e^{-r(s-k)\Delta t} \pi^\Delta(P_{t_s}, m_{t_s}, m^*_{t_{s+1}}) \\
&+ e^{-rw\Delta t} q(t_{k+w}, P_{t_{k+w}}, I_{t_{k+w+1}}, m_{t_{k+w+1}}). \qquad (19)
\end{aligned}
$$

Similarly, the continuation value $q(t_k, P_{t_k}, I_{t_{k+1}}, m_{t_{k+1}})$ can be written as

$$
q(t_k, P_{t_k}, I_{t_{k+1}}, m_{t_{k+1}}) = \mathbb{E}\Big[v_{k+1:k+w}(\pi, q)(\mathbf{P}_{t_{k+1}}, \mathbf{I}_{t_{k+1}}, \mathbf{m}_{t_{k+1}}) | P_{t_k}, I_{t_{k+1}}, m_{t_{k+1}}\Big]. \qquad (20)
$$

Then one can use the $v_{k:k+w}$ to construct alternative stochastic simulators that would lead to further ways for estimating $\hat{q}(t_k, \cdot)$. The partial-path construction in (20) can

be traced back to [15,16]. It encompasses the TvR choice $w = 1$ that we employ in this article, and the Longstaff–Schwartz (CLS) algorithm [28] where $w = K - k - 1$. Note that (19) also nests the final pathwise profits $v_{0:K}$ used for estimating $\hat{V}(0, \cdot)$.

Remark 3 (No More Global Paths) Traditionally RMC is implemented by generating N *global* paths (P_t^n) for the exogenous (price) process starting at $t = 0$ until maturity T, which are then stored permanently in memory for the entire backward induction, introducing significant overhead. Algorithm 1 replaces this with a design $\mathcal{D}_k$ and the associated one-step trajectories $(P_{t:t+1}^n, I_{t+1}^n)$.

Remark 4 (Dimension of the Regression Problem) In the case where there are no switching costs $K(i, j) \equiv 0\ \forall i, j$, the dimensionality of the regression problem can be reduced from 3 to 2. Indeed, since the inventory process I_t is completely controlled, the continuation value only depends on the inventory and regime (I_{t_k}, m_{t_k}) through the next-step inventory $I_{t_{k+1}} = I_{t_k} + a(c(m_{t_k}))\Delta t$. Analogously, the value function is then independent of the current regime, $V(t_k, P_{t_k}, I_{t_k})$. With a slight abuse of notation we then write $q(t_k, P_{t_k}, I_{t_{k+1}})$, working with the projection subspace generated by $(P_{t_k}, I_{t_{k+1}})$.

When switching costs are present, the same reduction is possible during the regression, but the present regime m_{t_k} remains a part of the state since it affects the continuation value q, so no overall dimension savings are achieved. Practically, this is handled by solving a distinct regression for each $m \in \mathcal{J}$ as in (16).

Remark 5 (Regress Now vs Regress Later) An alternative to (16) is to use as state variables $(P_{t_{k+1}}, I_{t_{k+1}})$ during the projection, and then take the conditional expectation analytically:

$$\check{q}(t_k, P, I, m) = \mathbb{E}\left[\arg\min_{h_{t_{k+1}} \in \mathcal{H}_{k+1}} \sum_{n=1}^{N} |h_{t_{k+1}}(P_{t_{k+1}}^n, I_{t_{k+1}}^n) - v_{k+1}^n|^2 \,\middle|\, P_{t_k} = P, I_{t_k} = I, m_{t_k} = m\right]. \tag{21}$$

This is known as Regress Later Monte Carlo (RLMC) and lowers the variance in the estimated $\hat{q}$ [32]. However, the requirement of closed form expressions for the conditional expectation generally rules out use of RLMC with non-parametric approximation spaces.

Remark 6 (Role of the Testing Set) Note that the final estimate $\hat{V}(0, P_0, I_0, m_0)$ depends on the out-of-sample simulations $\mathbf{P}_{0:T}^{1:N'}$, as well as the in-sample simulations $(\mathbf{P}_{t:t+1})$. To facilitate comparison of different methods, where possible we fix the test scenario database $(\mathbf{P}_{0:T}^{n'})$, whereby we can directly evaluate the different controls/cumulative revenues obtained along a given sample path of the price process.

Remark 7 (DEA vs. Conventional RMC for Storage Problems) The DEA template nests essentially all existing RMC approaches to (6). For example, inventory backpropagation can be interpreted as a specific recipe how to build $\mathcal{D}_k$ given $\mathcal{D}_{k+1}$ and the control map at step t_{k+1}. Inventory discretization is a specific combination of $\mathcal{D}_k$ and $\mathcal{H}_k$ that applies piecewise regression plus interpolation, see Sect. 4.1.1. In particular, the DEA emphasizes the unified treatment of P and I dimensions, with their

stochastic/endogenous dynamics only implicitly appearing as a structural property of the pathwise v_{k+1} simulator. Hence, we decouple the backward induction inherent to DPP from the emulation task that is based on the forward stochastic simulator. In turn, the template offers many new strategies (see below) that can (i) improve statistical efficiency, i.e. better use of the computational resources, such as speed, memory, and simulation budget; (ii) lead to more automated solvers that require less fine-tuning (e.g. no need to directly specify basis functions) and adapt to the problem structure; (iii) facilitate application of RMC on new problem instances through schemes that are less sensitive to dimensionality, model dynamics, and payoff format; (iv) create new links between RMC and existing emulation strategies in machine learning (including the ability to re-use existing code).

4 Approximation spaces

In this section we fix a given time step t_k and consider the problem of approximating the continuation value $q(t_k, P, I, m)$ viewed as a function $h_{t_k}(P, I)$, with m treated as a discrete parameter ("factor" level). Below we generically use $\mathbf{x} = (P^n_{t_k}, I^n_{t_{k+1}})_{n=1}^N$ and $\mathbf{y} = (v^n_{k+1})_{n=1}^N$ to represent the dataset used during the regression.

The statistical assumption is that the input–output relationship is described by

$$y^n = h(x^n) + \sigma^2 \xi, \text{ where } \xi \sim \mathcal{N}(0, 1), \tag{22}$$

where $h \in \mathcal{H}_k$ is the unknown function to be learned, and $\sigma^2 \xi$ is the noise. In our case, the noise is due to the stochastic shocks in $(P_{t_{k+1}})$ which induce variation in the realized pathwise profit relative to the continuation value.

Selection of $\mathcal{H}_k$ is key because the intrinsic approximation error, i.e. the distance between the true $q(t_k, \cdot)$ and the closest element in $\mathcal{H}_k$, strongly affects the quality of the solution. Among schemes that have been explored in the literature are global polynomial regression [4,7,10], radial basis functions [31], support vector regressions [30], kernel regressions [27], neural networks [19], and piecewise linear regression [39].

All of the above can be straightforwardly implemented within the DEA template, so that Algorithm 1 nests all these proposals. Below, we provide more details on three representative schemes which in our experience have been most promising. After reviewing the piecewise regression and the inventory-discretization approaches, we discuss nonparametric regressions where $\mathcal{H}_k$ is characterized through the design sites in $\mathcal{D}_k$. In particular, inspired by the recent work [29] on Bermudan options, we introduce Gaussian process (GP) regression for solving storage problems. To our knowledge, ours is the first paper to use GPs in such context.

4.1 Bivariate piecewise approximation

The classical regression framework is a linear parametric model where $\mathcal{H}_k$ is the vector space spanned by some basis functions (ϕ_i). Then the prediction at a generic x_* is

controlled by the regression coefficients $\vec{\beta}$: $h(x_*) = \vec{\beta}^T \vec{\phi}(x_*)$. A simple choice dating back to Longstaff–Schwartz's seminal work [28] is to use polynomial bases. Such polynomial regression (PR) has also been employed for storage problems in [4,7,10]. A degree-r global polynomial approximation $h_{t_k} = \sum_i \beta_i \phi_i$, has $(r+1)(r/2+1)$ basis functions and takes $\phi_i(P, I) = P^{\alpha_1(i)} I^{\alpha_2(i)}$, where the total degree of the basis function is $\alpha_1 + \alpha_2 \leq r$. For example, a global quadratic approximation ($r = 2$) has six basis functions $\{1, P, P^2, I, I^2, P \cdot I\}$, and a cubic PR has ten basis functions. Our experiments indicate that typically PR leads to poor performance due to the resulting stringent constraints on the shape of the continuation value and consequent back-propagation of error.

A popular alternative is to use piecewise approximations based on partitioning the space of (P, I), restricted to $[\min_{1\leq n\leq N} P_{t_i}^n, \max_{1\leq n\leq N} P_{t_i}^n] \times [I_{\min}, I_{\max}]$, into $M = M_P \times M_I$ rectangular sub-domains $\mathcal{D}_{i_1,i_2}$, $i_1 = 1, 2, \ldots, M_P$; $i_2 = 1, 2, \ldots, M_I$. Piecewise regression allows to localize the projection errors (while global regression is apt to oscillate) and tends to be more stable empirically. Relative to PR, piecewise regressions are also more "robust" to fitting arbitrary shapes of the continuation value. We then consider basis functions of the form $\{\phi_g^{i_1,i_2}\}$, with support restricted to $\mathcal{D}_{i_1,i_2}$. For example, in piecewise linear approximation we have $g = 1, 2, 3$ with

$$\begin{aligned}
\phi_1^{i_1,i_2}(P, I) &= \mathbf{1}_{(P,I)\in\mathcal{D}_{i_1,i_2}} \\
\phi_2^{i_1,i_2}(P, I) &= P \cdot \mathbf{1}_{(P,I)\in\mathcal{D}_{i_1,i_2}} \\
\phi_3^{i_1,i_2}(P, I) &= I \cdot \mathbf{1}_{(P,I)\in\mathcal{D}_{i_1,i_2}}.
\end{aligned}$$

Overall we then have $3M_P M_I$ coefficients to be estimated. Higher-degree terms could also be added, e.g. a cross-term $P \cdot I$ or quadratic terms P^2, I^2. Piecewise regression offers a divide-and-conquer advantage with the overall fitting done via a loop across $\mathcal{D}_{i_1,i_2}$'s; in each instance only a small subset of the data is selected to learn a few coefficients. This decreases the overall workload of the regression substep, and allows for parallel processing. The main disadvantage of this approach is the inherent discontinuity in $\hat{q}$ at the sub-domain boundaries, and the need to specify M_P, M_I and then construct the rectangular sub-domains $\mathcal{D}_{i_1,i_2}$. We refer to [39] for several adaptive constructions, including equal-weighted and equi-gridded.

4.1.1 Piecewise continuous approximation

Another approach is to construct a piecewise approximation that is continuous through a linear interpolation. For example, after discretizing the endogenous inventory variable into $M_I + 1$ levels $I_0, I_1, \ldots, I_{M_I}$, we fit an independent degree-M_P monomial in P for each level, i.e. optimize for $\hat{h}_j(P) := \sum_i \beta_{ij} \phi_i(P)$ for $j = 0, \ldots, M_I$, giving a total of $(M_I + 1)M_P$ regression coefficients. (Conversely, one could also fit a piecewise linear model with M_P sub-domains in the P-dimension.) The final interpolated prediction for arbitrary $I \in (I_j, I_{j+1})$ is then piece-wisely defined as

$$\widehat{h}_{t_k}(P, I) := \delta(I)\, \hat{h}_j(P) + (1 - \delta(I))\, \hat{h}_{j+1}(P), \tag{23}$$

where $\delta(I) = \frac{I-I_j}{I_{j+1}-I_j}$. This scheme leverages the fact that the stochastic shocks are only in P, and effectively replaces the problem of learning $\hat{q}$ over (P, I) with a collection of one-dimensional (hence simpler) regressions in P only. In principle this allows to re-use P-simulations across different I-levels. It is intrinsically smooth in P and piecewise linear in I.

4.2 Local polynomial regression (LOESS)

Local regressions minimize the worry regarding the choice of basis functions by constructing a non-parametric fit that solves an optimization problem at each predictive site. Given a dataset $\{\mathbf{x}, \mathbf{y}\}$ (as a reminder, $x \equiv (P, I)$ is two-dimensional throughout this section), the prediction using LOESS at x_* is $h(x_*) = \sum_{i=1}^{r} \beta_i(x_*)\phi_i(x_*)$ where the *local* coefficients $\beta_i(x_*)$ are determined from the weighted least-squares regression

$$\vec{\beta}(x_*) = \arg\min_{\vec{\beta}\in\mathbb{R}^r} \frac{1}{N}\sum_{n=1}^{N} \kappa(x_*, x^n)\left[y^n - \vec{\beta}^T\vec{\phi}(x^n)\right]^2. \tag{24}$$

The weight function $\kappa(x_*, x) \in [0, 1]$ gives more weight to y^n's from inputs close to x_*, akin to kernel regression. Since a separate optimization is needed for each x_*, to make N' predictions (24) has complexity of $\mathcal{O}(NN')$. Implemented in the context of real options for mining by [27], LOESS was enhanced through a "sliding trick" that reduces complexity to $\mathcal{O}((N + N')\log N)$.

4.3 Gaussian process regression (GPR)

Gaussian process regression is a non-parameteric technique widely used in spatial statistics and more recently for emulation tasks in machine learning. Besides its ability to efficiently reconstruct non-linear input–output relationships, its popularity is attributed to very few tunable hyperparameters, intrinsic smoothness of the obtained approximation, and symbiotic links to adaptive experimental designs. In our context, GPR offers a flexible alternative to parametric regression, obviating the need to directly specify regression bases or manually construct the mesh within the piecewise approach. GPR is superficially similar to LOESS, in that the prediction $h(x_*)$ is a weighted average of sampled outputs $\mathbf{y}$. The underlying relationship $h : x \mapsto y$ is taken to be a realization of a Gaussian random field, i.e. $\{h(x^i)\}_{i=1}^N$ is a sample from the multivariate Gaussian distribution with mean $\{m(x^i)\}_{i=1}^N$ and covariance function $\{\kappa(x^i, x^j)\}_{i,j=1}^N$. Among many options in the literature, arguably the most popular is the squared exponential kernel,

$$\kappa(x^i, x^j) = \sigma_f^2 \exp\Big(-\frac{1}{2}(x^i - x^j)^T\boldsymbol{\Sigma}^{-1}(x^i - x^j)\Big),$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix. Traditionally, diagonal elements of the matrix $\boldsymbol{\Sigma}$ are known as the lengthscale parameters, and σ_f^2 as the signal variance. Together they

are often referred to as hyper-parameters. While the lengthscale parameters determine the smoothness of the surface in the respective dimension, σ_f^2 determines the amplitude of the fluctuations. In Fig. 2 we observed that for a fixed price, the continuation value function is linear in the inventory dimension, and has non-linear behavior in the price dimension. GPR captures this difference through its scale parameters, which will be "large" for inventory (slow decay of correlation, so little curvature) and "small" for price dimension (fast correlation decay allow for wiggles in terms of P). The hyperparameters are estimated through likelihood maximization. For the prior mean we take $m(x) = \beta_0$ where β_0 is learned together with the other hyperparameters.

For any site x_*, $h(x_*)$ is a random variable whose conditional distribution given $\{\mathbf{x}, \mathbf{y}\}$ is:

$$h(x_*)|\mathbf{y} \sim \mathcal{N}\Big(m(x_*) + H_*\mathbf{H}^{-1}(\mathbf{y} - m(\mathbf{x})),\, H_{**}(x_*) - \mathbf{H}_*(x_*)\mathbf{H}^{-1}\mathbf{H}_*(x_*)^T\Big), \tag{25}$$

where the $N \times N$ matrix covariance matrix $\mathbf{H}$ and the $N \times 1$ vector $\mathbf{H}_*(x_*)$ are

$$\begin{aligned}
\mathbf{H} &:= \begin{bmatrix} \kappa'(x^1, x^1) & \kappa(x^1, x^2) & \dots & \kappa(x^1, x^N) \\ \kappa(x^2, x^1) & \kappa'(x^2, x^2) & \dots & \kappa(x^2, x^N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(x^N, x^1) & \kappa(x^N, x^2) & \dots & \kappa'(x^N, x^N) \end{bmatrix}, \\
\mathbf{H}_*(x_*)^T &:= \begin{bmatrix} \kappa(x_*, x^1) \\ \kappa(x_*, x^2) \\ \vdots \\ \kappa(x_*, x^N) \end{bmatrix}, \quad H_{**}(x_*) = \kappa'(x_*, x_*),
\end{aligned} \tag{26}$$

where $\kappa'(x^i, x^j) = \kappa(x^i, x^j) + \sigma^2$. Consequently, the prediction at x_* is $\hat{h}(x_*) = m(x_*) + \mathbf{H}_*(x_*)\mathbf{H}^{-1}(\mathbf{y} - m(\mathbf{x}))$ and the posterior GP variance $s^2(x_*) := H_{**}(x_*) - \mathbf{H}_*(x_*)\mathbf{H}^{-1}\mathbf{H}_*(x_*)^T$ provides a measure of uncertainty (akin to standard error) of this prediction. GPR generally has $\mathcal{O}(N^3 + NN'^2)$ complexity, similar to kernel regression. GPR usually performs extremely well but becomes prohibitively expensive for $N \gg 1000$.

Remark 8 The above is the most basic version of GP emulation. There is an extensive GP ecosystem containing numerous extensions for optimizing GPR in a specific context. Some of the relevant aspects include more advanced prior mean specification for $m(\cdot)$; other (including adaptive) kernel families $\kappa(\cdot, \cdot)$; heteroskedastic models that can handle state-dependent simulation noise $\sigma^2(\cdot)$; further techniques for selecting GP hyperparameters; and piecewise models for allow for spatially non-stationary covariance. See [6,29] and references therein.

5 Simulation design

The second central piece of Algorithm 1 concerns the designs $\mathcal{D}_k$. The choice of $\mathcal{D}_k$ directly affects the quality of $\hat{q}(t_k, \cdot)$: the approximation will generally be better in regions where $\mathcal{D}_k$ has many input sites, and worse where $\mathcal{D}_k$ is sparse. Trying to make predictions of $\hat{q}(t_k, \cdot)$ beyond $\mathcal{D}_k$, i.e. extrapolating, is especially prone to large errors. Thus, the shape of $\mathcal{D}_k$ is akin to introducing weights within the projection (15), emphasizing some sub-domains of the input space and de-emphasizing others. This effect is particularly strong for non-parametric or piecewise regression schemes where there is a close link between $\mathcal{D}_k$ and $\mathcal{H}_k$. Spatially, a good simulation design should (i) cover all regions containing potential (P_{t_k}, I_{t_k}) pairs; (ii) target the region of interest that is most relevant for selecting the optimal action $\widehat{m}(t_k, P_{t_k}, I_{t_k})$. Statistically, $\mathcal{D}_k$ ought to maximize the learning rate of $\hat{q}(t_k, \cdot)$ and lead to stable regressions, i.e. low empirical sensitivity of $\hat{q}(t_k, \cdot)$ across algorithm runs.

Like the regression sub-problem, the template gives the user a wide latitude in selecting $\mathcal{D}_k$ given a simulation budget N. We identify four relevant aspects of potential designs: joint vs. product; adaptive vs. space-filling; deterministic vs. stochastic; and unique vs. replicated. Last but not least, we discuss the design size N. To summarize the range of simulation designs we use a short-hand nomenclature. Product designs are identified as $\mathcal{D}_P \times \mathcal{D}_I$, while joint designs are denoted with a single symbol. Subscripts are used where necessary to identify the dimensionality of the respective design. Different design types are identified by different letters.

To set the stage, let us summarize the "conventional" design [4,7,31]. Traditionally, the design to solve the storage problem relied on a mix of global paths together with inventory discretization. In the price dimension, it consists of choosing N^P initial conditions for the price process P_{t_0} at time t_0 and sampling $n = 1, \ldots, N^P$ paths $P^n_{t_{k+1}}$ following the conditional density $p(t_{k+1}, .|t_k, P_{t_k})$, until terminal date T. The resulting collection $(P^n_{t_k})$ is used for the design "mesh" at each t_k. For the endogenous inventory dimension, I is discretized into N^I levels: $I^l = l\Delta I$, $\Delta I = \frac{I_{\max} - I_{\min}}{N^I - 1}$, $l = 0, 1, \ldots, N^I - 1$. The overall design $\mathcal{D}_k$ has $N = N^P N^I$ sites and is constructed as the Cartesian product $\{P^1_{t_k}, P^2_{t_k}, \ldots, P^{N^P}_{t_k}\} \times \{I^0, I^1, \ldots, I^{N^I - 1}\}$. In our terminology, this is a product design that is adaptive and stochastic in P, space-filling and deterministic in I, and has no replication. We label it as $\mathcal{P} \times \mathcal{G}$, representing a density-based "probabilistic" design $\mathcal{P}$ in the first coordinate of x, and a gridded design $\mathcal{G}$ in the second coordinate.

The shape of a $\mathcal{P} \times \mathcal{G}$ design is convenient for the piecewise continuous regression scheme of Sect. 4.1.1 that treats the P and I coordinates separately, yielding stable empirical $\hat{q}(t_k, \cdot)$. While this recipe can lead to competitive results, it is clearly more prescriptive than adaptive to the particular problem instance. The fact that it relies on a grid in the I-coordinate makes it poorly suited for scaling into higher dimension, while using a random sample drawn from the density of $P_{t_{k+1}}$ is prone to requiring extrapolation in subsequent `predict` calls. Through DEA we are able to search numerous other feasible approaches to maximize performance.

5.1 Space filling designs

To achieve the goal of learning $(P, I) \mapsto q(t_k, P, I, m)$ we need to explore continuation values throughout the input domain. A simple mechanism to achieve this is to spread out the design sites to fill the space. A gridded design, with design sites uniformly selected using a mesh size Δ like in the conventional approach above, is an example of such *space-filling* sequences. Exploration through "spreading out" $\mathcal{D}$ can be supported by more rigorous criteria, such as A- or D-optimality that quantify the optimal way to reduce the global $L^2(Leb)$ approximation error.

Space-filling can be done either deterministically or randomly. For the deterministic case, besides the grid $\mathcal{G}$ one may employ various Quasi Monte Carlo (QMC) sequences, for example the Sobol sequences $\mathcal{S}$. Sobol sequences are useful in dimension $d > 1$ where they can produce a d-variate space filling design of any size N, whereas a grid is limited to rectangular constructions of size $N_P \times N_I$. QMC sequences are also theoretically guaranteed to provide a good "uniform" coverage of the specified rectangular domain. We experimented with the following two setups:

- A 1-D Sobol sequence $\mathcal{S}_1$ of size N^P for the price dimension, restricted to $[P_{\min}, P_{\max}]$ at each time-step t_k, $k = 0, \ldots, K$. We then discretize the inventory dimension as $\{I^1, I^2, \ldots, I^{N^I}\}$, similar to conventional design, and the final $\mathcal{D}$ is the product $\mathcal{S}_1 \times \mathcal{G}$.
- Alternatively, we generate N design sites from the 2-D Sobol sequence $\mathcal{S}_2 = \{P^n, I^n\}_{n=1}^N$ on the restricted domain $[P_{\min}, P_{\max}] \times [I_{\min}, I_{\max}]$.

An example of a randomized space filling design is taking $P_{t_k}^n \sim \text{Unif}(P_{\min}, P_{\max})$ i.i.d. Because i.i.d. uniforms tend to cluster, there are variance-reduced versions, such as Latin hypercube sampling (LHS). In two dimensions, the LHS design $\mathcal{L}_2$ stratifies the input space into a rectangular array and ensures that each row and column has exactly one design site.

Note that if the same deterministic space-filling method is employed across time-steps, the design $\mathcal{D}_{t_k} \equiv \mathcal{D}$ becomes identical in t_k. This may generate "aliasing" effects from the regression scheme, i.e. approximation artifacts around (P^n, I^n) due to the repeated regressions and respective error back-propagation. Changing or randomizing $\mathcal{D}_{t_k}$ across t_k's is one remedy and often preferred as an implementation default. Another challenge with space-filling designs is the need to specify the bounding box $[P_{\min}, P_{\max}] \times [I_{\min}, I_{\max}]$, which is easy in I but not obvious in the unbounded P-coordinate.

5.2 Adaptive designs

In contrast to space-filling designs that aim to explore the input space, adaptive designs exploit the observation that the quality of $\hat{V}_0$ depends on the correct prediction of storage actions along *controlled* paths $(P_{t_k}^{n'}, \hat{I}_{t_k}^{n'})$. Therefore, the region of interest at t_k where we should target the best estimation of $\hat{q}(t_k, \cdot)$ is the region where the $(P_{0:T}^{n'}, \hat{I}_{0:T}^{n'})$ trajectory is most likely to be. This suggests to use the distribution of $(P_{t_k}, \hat{I}_{t_k})$ when constructing $\mathcal{D}_k$, cf. Fig. 3c. Information about the distribution of the

system state was already leveraged in the conventional approach which used randomized, *probabilistic* design for $\mathcal{D}_P$. Similarly, [31] used a non-uniform discretization in I to refine the mesh closer to $I_{\min}$ and $I_{\max}$, since inventory tends to be either 0% or 100% full, see Fig. 1.

An ideal adaptive design would reflect the *bivariate* distribution of $(P, \hat{I})$. Of course, this is not directly feasible since $\hat{I}_{t_k}$ is endogenous to the controls on $[0, t_k]$. One strategy is to use a small part of the simulation budget to first create a policy using a traditional/space-filling design. In the second step, one runs a forward simulation of this policy to construct a proxy for the joint distribution of $(P_{t_k}, \hat{I}_{t_k})$, and hence link $\mathcal{D}_k$ to the joint probabilistic design $\mathcal{P}_2$.

There are numerous variations on generating adaptive designs that reflect some target density $\mathfrak{p}(\cdot)$ (either bivariate or univariate for P and I). Instead of a joint design, one could build a product design $\mathcal{P} \times \mathcal{P}$ that matches the respective marginal densities $P^n_{t_k} \sim \mathfrak{p}^P(t_k, \cdot)$ and $I^n_{t_k} \sim \mathfrak{p}^I(t_k, \cdot)$. Yet another approach is to deterministically quantize the target density $\mathfrak{p}$, similar to numerical quadrature methods, to return a discrete representation with N^P sites.

5.3 Batched designs

Non-parametric techniques like GPR and LOESS improve accuracy at the cost of increased regression overhead. Indeed, for both methods the complexity is at least quadratic in the number of sites N which can become prohibitive for $N \gg 1000$. One solution to overcome this hurdle is to use replicates, i.e. re-use the same design site for multiple simulations. Thus, rather than having thousands of distinct design sites equivalent to the simulation budget N, we select only a few 100 distinct sites N_s and generate $N_b := N/N_s$ paths from each design site. Formally this means that we distinguish between the N initial conditions $(P^n_{t_k}, I^n_{t_k})_{n=1}^N$ for simulating pathwise continuation values and the *unique* design sites $N_s \ll N$ which comprise the design $\bar{\mathcal{D}}$. The latter can then be of any type, space-filling, adaptive, etc. The use of such replicated designs is common in the design of experiments literature, but has been little explored in the RMC context.

Given a design site $(P^n_{t_k}, I^n_{t_{k+1}})_{n=1}^{N_s}$, we make N_b draws $P^{n,(m)}_{t_{k+1}}$ and evaluate the corresponding pathwise continuation value $v^{(m)}_{k+1}(P^{n,(m)}_{t_{k+1}}, I^n_{t_{k+1}})$, $m = 1, \ldots, N_b$. For kernel-based techniques like GP and LOESS one may then work with pre-averaged values, i.e. first evaluate the empirical average:

$$\bar{v}^n_{k+1}(P^n_{t_k}, I^n_{t_{k+1}}) = \frac{1}{N_b} \sum_{m=1}^{N_b} v^{(m)}_{k+1},$$

across the N_b replicates. One then feeds the resulting dataset $(P^n_{t_k}, I^n_{t_{k+1}}, \bar{v}^n_{k+1})_{n=1}^{N_s}$ into the regression equations to estimate the continuation function.

Besides reducing the overall time spent in regressions (which can easily be several orders of magnitude), a batched design has the advantage of reduced simulation variance of $\bar{v}$ at each design site, thus improving the signal-to-noise ratio. While a

replicated design is sub-optimal (in terms of maximizing the quality of the statistical approximation), in practice for large N (which are frequently in the 100 of 1000) the loss of fidelity is minor and is more than warranted given the substantial computational gains.

Beyond uniform batching that uses a fixed N_b number of replicates at each site, one could also employ adaptive batching with site-specific N_b with more replications around the switching boundaries to gain better precision [6].

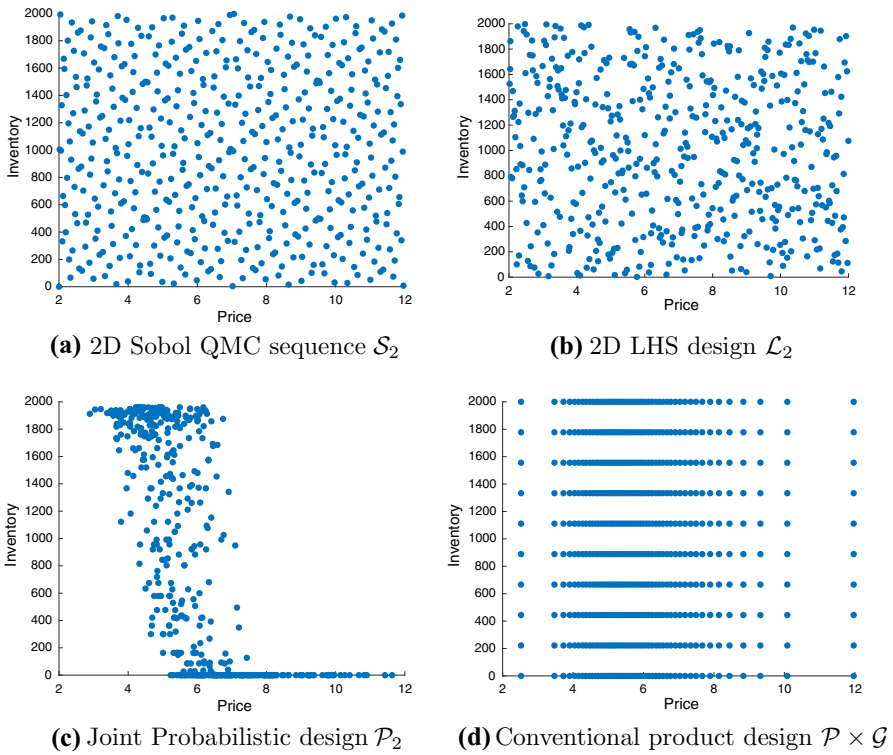
5.4 Dynamic designs

DEA lets us easily combine different designs at various time steps. For example, one may vary the step-wise simulation budget, employing larger N near maturity to effectively capture the effect of the terminal conditions in the continuation function, and lesser budget thereafter.

We conclude this section with two illustrations. In Fig. 3 we display four representative designs: Sobol space-filling sequence in 2D $\mathcal{S}_2$, LHS in 2D $\mathcal{L}_2$, joint probabilistic $\mathcal{P}_2$, and conventional design $\mathcal{P} \times \mathcal{G}$. These will also be used in the numerical examples below. While the Sobol sequence fills the input space with a symmetric pattern, LHS is randomized. The joint probabilistic design mimics the distribution of the state variables $(P_t, \hat{I}_t)$, putting most sites at the boundaries of the inventory I_{min}, I_{max} and around the mean of the price. Note that this design is very aggressive and only explores a small subset of the input space; therefore, in the following numerical examples we blend (via a statistical mixture) probabilistic designs with other types. Such blending is a natural way to resolve the underlying exploration-exploitation trade-off. Finally, conventional design discretizes the inventory while maintaining the adaptive distribution in the price dimension.

Next, in Fig. 4 we display the effect of regression and design on the continuation value function and the corresponding control maps. The left panel compares the continuation value function of GP-2D and PR-2D at the last step before maturity $t = T - \Delta t$. We observe that PR-2D has a poor fit compared to GP-2D, which almost perfectly matches the "hockey-stick" penalty. Furthermore, -1D regressions have "non-smooth" switching boundaries due to the piecewise regressions in I. This is evident in the center panel of the figure, where the control map shows jumps at $I \in \{500, 1000, 1500\}$, the intermediate discretization levels of inventory (we used $N^I = 5$ with $\Delta I = 500$). This behaviour becomes less prominent when N^I is increased. The right panel of the figure visualizes two control maps for PR-1D with LHS and conventional design. The effect of an oversized input domain (we used $[P_{min}, P_{max}] = [2, 20]$ which explores *too much*) for LHS is evident, as we notice that the Store region is much too wide in the latter variant. Thus, the controller does not benefit from withdrawing when prices are $P \in [7, 7.5]$, ultimately resulting in lower valuation. Overall, Fig. 4 highlights the need to properly pick both $\mathcal{H}$ and $\mathcal{D}$ to obtain good performance.

The effect of design shape can be conveniently visualized with Gaussian process emulators which provide a proxy for local estimation standard error through the posterior GP standard deviation $s(x)$. Figure 5 compares $s(x)$ that results from Mixture (we used a 60/40 mix of $\mathcal{S}_2$ and $\mathcal{P}_2$) and Space-filling (Sobol QMC $\mathcal{S}_2$) 2D designs



(a) 2D Sobol QMC sequence $\mathcal{S}_2$ **(b)** 2D LHS design $\mathcal{L}_2$

(c) Joint Probabilistic design $\mathcal{P}_2$ **(d)** Conventional product design $\mathcal{P} \times \mathcal{G}$

Fig. 3 Illustration of different simulation designs $\mathcal{D}$. In all cases we take $N = 500$ design sites

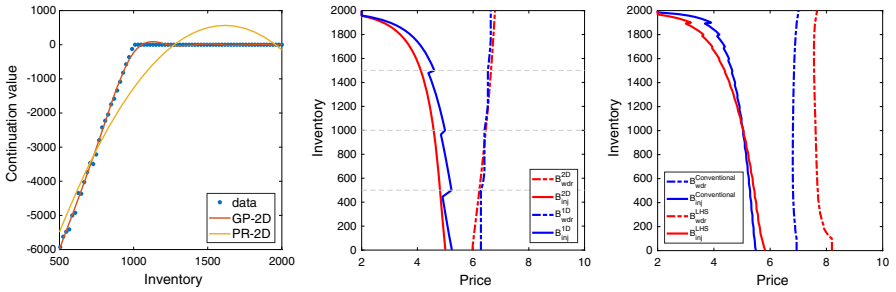


Fig. 4 *Left panel:* Comparison of the continuation function $q(t, P, \cdot)$ for different regressions at one step to maturity $t = T - \Delta t$ and $P = 6$. *Center panel:* the control maps $\widehat{m}(t, P, I)$ at $t = 2.7$ years corresponding to GP-1D and GP-2D regressions. For GP-1D we used $N_s = 100$, $N_b = 20$ and $N_I = 5$. For GP-2D we used $N_s = 500$ and $N_b = 20$. The horizontal lines represent the inventory discretization levels for the GP-1D design. *Right panel:* the control maps corresponding to conventional and LHS design with PR-1D regressions. We used $N^P = 2000$, $N^I = 21$ and price domain for LHS $[P_{min}, P_{max}] = [2, 20]$

of same size $N = 500$. For a space-filling design we observe a constant posterior variance, i.e. GPR learns $\hat{q}(t_k, \cdot)$ equally well across the interior of the regression domain. At the same time, the posterior standard deviation is very high around the

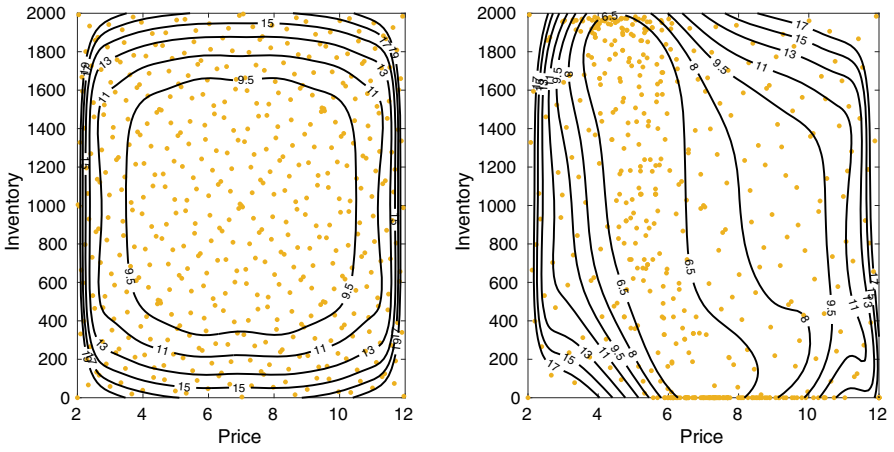


Fig. 5 Impact of design on the posterior standard error $s(x)$ of GPR at $t = 1.5$ years. We show the contours of $x \mapsto s(x)$ with dots indicating the underlying respective designs with $N = 500$ sites. *Left panel:* Space-filling design (Sobol QMC $\mathcal{S}_2$) of Fig. 3a leads to rectangular level sets of $s(\cdot)$. *Right panel:* under a Mixture design, $s(\cdot)$ resembles the shape of $\mathcal{P}_2$ in Fig. 3c

edges of $[P_{\min}, P_{\max}] \times [I_{\min}, I_{\max}]$, which is problematic for correctly identifying the strategy when inventory is almost full or almost empty. For the Mixture design, the posterior variance reflects the concentration of the input sites along the diagonal, compare to the joint probabilistic design $\mathcal{P}_2$ in Fig. 3c. In turn, this is beneficial for learning the control map, as the GPR prediction is most accurate along the switching boundaries, cf. Fig. 2a. Higher precision around the switching boundaries allows the Mixture design to allocate the simulation budget to the regions where accuracy is needed most. We emphasize that such local inference quality is only available with non-parametric tools; global schemes like PR cannot directly benefit from focused designs.

6 Natural gas storage facility

To illustrate DEA and its various ingredients, we present several numerical studies. In this section, we consider the gas storage problem which has been already explored in existing literature and hence affords a good testbed for comparison. Our main aims are to: (i) illustrate multiple implementations of DEA in terms of picking the regression spaces $\mathcal{H}$ and designs $\mathcal{D}$; (ii) explain the effect of the DEA modules on the ultimate problem solution, e.g. on the control maps; (iii) document the relative performance of the different schemes to draw some conclusions on their merit in this context. While we present quite a lot of numeric experiments, we stress that we do not seek to provide a rigorous benchmarking, but rather view these as case studies. To this end, we do not concentrate on one single setup, instead exploring several formulations, including with and without switching costs, and with varying dimensionality. Our main message is that there is a lot of gains to be unlocked from fine-tuning DEA through carefully

selecting its ingredients, in particular by mixing-and-matching existing proposals. Thus, the advertised flexibility of the template indeed leads to superior performance through optimizing the regression and design aspects, and exploiting the synergies between the two.

6.1 Market and storage description

In this section we revisit the gas storage problem in the setting of Chen and Forsyth [11]. The exogenous state process P follows logarithmic mean-reverting dynamics

$$P_{t_{k+1}} - P_{t_k} = \alpha(\underline{P} - P_{t_k})\Delta t + \sigma P_{t_k} \Delta W_{t_k}, \tag{27}$$

where P_t is the spot price per unit of natural gas and is quoted in "dollars per million of British thermal unit ($/MMBtu)". The inventory I_t is quoted in million cubic feet (MMCf). Since roughly 1000 MMBtu = 1 MMcf, we multiply P_t by 10^3 when calculating revenue or profit.

The penalty function $W(P_T, I_T)$ at maturity is $W(P_T, I_T) = -2P_T \max(1000 - I_T, 0)$. Thus, the target inventory is $I_T = 1000$ or 50% capacity. There is no compensation for excess inventory and a strong penalty (at 200% of the market price) for being short. As a result, the value function at maturity has a non-smooth hockey-stick shape in I, with zero slope for $I_T > 1000$ and a slope of $-2P_T$ otherwise.

The withdrawal and injection rates are inventory-dependent and given by

$$c_{wdr}(I) = -k_1\sqrt{I} \text{ and } c_{inj}(I) = k_2\sqrt{\frac{1}{I + k_3} - \frac{1}{k_4}}.$$

These functional specifications are derived from the physical hydrodynamics of the gas storage facility, see, [36]. The resulting dynamics of the inventory is:

$$I_{t_{k+1}} = I_{t_k} + \begin{cases} c_{wdr}(I_{t_k})\Delta t, & \text{if } m_{t_k} = -1 \quad \text{(withdrawal);} \\ 0 & \text{if } m_{t_k} = 0; \\ (c_{inj}(I_{t_k}) - k_5)\Delta t, & \text{if } m_{t_k} = +1 \quad \text{(injection).} \end{cases} \tag{28}$$

The constant k_5 measures the cost of injection which is represented as "gas lost". It leads to a profit gap between production and injection whereby no-action ($m^* = 0$) will be the optimal action if the price is "close" to the mean level, $P_t \approx \underline{P}$.

In the numerical experiments below we discretize T into $K = 1000$ steps, so that $\Delta t = 0.001T$, the rest of the parameters are listed in Table 1. The switching costs are taken to be zero $K(i, j) \equiv 0 \forall i, j$. Absence of switching cost reduces the state variables in the continuation function $q(t_k, \cdot)$ in (16) to $(P_{t_k}, I_{t_{k+1}})$. As a result, in this example every time step requires one projection of the 2D value function.

Table 1 Parameters for the gas storage facility in Sect. 6

$\alpha = 2.38,\ \sigma = 0.59,\ \underline{P} = 6$
$k_1 = 2040.41,\ k_2 = 7.3 \cdot 10^5,\ k_3 = 500,\ k_4 = 2500,\ k_5 = 620.5$
$I_{\max} = 2000$ MMcf, $T = 3$, $\Delta t = 0.003$, $r = 10\%$

6.2 Benchmarking setup

To benchmark the performance, we compare to two schemes utilizing conventional product design $\mathcal{P} \times \mathcal{G}$ (with inventory uniformly discretized): degree-3 global polynomial approximation over (P, I) (PR-2D) and piecewise continuous approximation (with degree-3 polynomial regressions in P at each inventory level I_k, PR-1D). These can be viewed as a "classical" TvR scheme with a joint regression [10,14], and the discretized-I version as used by [3,4,7,31]. Additionally, we implement five regression approaches (PR-1D, GP-1D, PR-2D, LOESS-2D and GP-2D) on several different designs, with simulation budgets: $N \simeq 10\,K, 40\,K, 100\,K$:

- Randomized space filling design implemented via LHS ($\mathcal{L}_1 \times \mathcal{G}$ for piecewise-continuous regression and $\mathcal{L}_2$ otherwise). We use a large conservative input domain $P \in [2, 10]$.
- Mixture-2D design with 40% of sites from space-filling and the remaining 60% from the joint empirical distribution $\mathcal{P}_2, \mathcal{D}_M := \mathcal{P}_2(0.6N) \cup \mathcal{L}_2(0.4N)$. To implement $\mathcal{P}_2$, we need to estimate $\mathfrak{p}^P(t_k, \cdot)$ and $\mathfrak{p}^I(t_k, \cdot)$. This is done offline by first running the algorithm with a small budget and conventional product design. We then generate forward paths $(P_{t_k}^{n'}, \hat{I}_{t_k}^{n'})$ to estimate the joint $(\mathfrak{p}^P(t_k, \cdot), \hat{\mathfrak{p}}^I(t_k, \cdot))$ at t_k. Since the marginal distribution $\hat{\mathfrak{p}}^I(t_k, \cdot)$ starts to concentrate around $I = 1000$ as we get close to the maturity of the contract, the resolution at other parts of the domain is reduced and $\mathcal{L}_2$ is to used to compensate for this effect.
- Adaptive-1D: For 1D regressions, we estimate $\mathfrak{p}^P(t_k, \cdot)$ as above, and then non-uniformly discretize the inventory to incorporate the fact that the optimally controlled inventory process $\hat{I}_{t_k}$ concentrates around $I_{\min}, I_{\max}$ and $I = 1000$. Discretization levels for each simulation budget are detailed in Table 6 in the Appendix.
- Dynamic time-dependent design that varies the simulation budget N_{t_k} across t_k. We used the specification $N_{t_k}(N_{(1)}, N_{(2)}) := N_{(1)}\mathbf{1}_{\{k<900\}} + N_{(2)}\mathbf{1}_{\{k\geq 900\}}$ such that (approximately) $0.9N_{(1)} + 0.1N_{(2)} \in \{10K, 40K, 100K\}$. Exact specification is given in Table 5 in the Appendix. We use two variants of it:
  - fixed projection space $\mathcal{H}$ (namely GP-1D) and conventional product design $\mathcal{D} = \mathcal{P} \times \mathcal{G}$.
  - time-dependent projection and designs, namely GP-2D and Mixture design for $k < 900$ and PR-1D and Conventional design for $k \geq 900$.

  The motivation for the above Dynamic scheme is to better handle the non-smooth terminal condition by devoting to it larger simulation budget, as well as using the -1D regression.

For GP we use `Matlab`'s in-built implementation `fitrgp`. For LOESS, we use the `curvefitting` toolbox again from `Matlab` that constructs local quadratic approximations based on the tri-cube weight function $\kappa(x_*, x^n) = \left(1 - (\frac{|x_* - x^n|}{\lambda(d, x_*)})^3\right)^3$. Above $\lambda(d, x_*)$ is the Euclidean distance from x_* to the most distant x^k within the span d. We use the default span of $d = 25\%$, keeping P on its original scale and re-scaling I to be in the range $[0, 2]$. To reduce the regression overhead of both GPR and LOESS we utilize batched designs with N_b replicates (see Appendix) on top of the underlying design type.

In order to compare the performance of different designs/regressions, we use the estimate of the value function $\hat{V}(0, P_0, I_0)$ at $P_0 = 6$, $I_0 = 1000$ using a fixed set of $N' = 10,000$ out-of-sample paths (i.e. fixed $P^{n'}_{0:T}$) as a performance measure.

6.3 Results

Table 2 presents the performance of different designs and regression methods. We proceed to discuss the results focusing on three different aspects: (i) impact of different regression schemes, in particular parametric PR vs. non-parametric GP and LOESS approaches; (ii) impact of simulation design; (iii) joint -2D vs. interpolated -1D methods.

First, our results confirm that the interpolated -1D method performs extremely well in this classical example, perfectly exploiting the 2-dimensional setup with a 1-dimensional inventory variable. In that sense the existing state-of-the-art is already excellent. There are two important reasons for this. First PR-1D is highly flexible with lots degrees of freedom, allowing a good fit (with overfitting danger minimized due to a 1D setting). Second, PR-1D perfectly exploits the fact that the value function is almost (piecewise) linear in inventory. We obtain a slight improvement by replacing PR-1D with GP-1D; another slight gain is picked up by replacing the equi-spaced inventory discretization with an adaptive approach that puts more levels close to the inventory boundaries. As a further enhancement, the Dynamic design utilizes a step-dependent simulation budget (to capture the boundary layer effect due to the non-smooth "hockey-stick" terminal condition that requires more effort to learn statistically), leading to significant improvement, highlighting the potential benefit of mixing-and-matching approximation strategies across time-steps. By taking N time-dependent one may effectively save simulation budget (e.g. the valuation for Dynamic GP-1D with $N = 10^4$ is comparable to $N = 2 \cdot 10^4$ for Adaptive GP-1D). Nevertheless, as we repeatedly emphasize, the -1D methods do not scale well if more factors/inventory variables are added.

The much more generic bivariate -2D regressions give a statistically equitable treatment to all state variables and hence permit arbitrary simulation designs. The resulting huge scope for potential implementations is both a blessing and a curse. Thus, we document both some good and some bad choices in terms of picking a regression scheme, and picking a simulation design. We find that PR-2D tends to significantly underperform which is not surprising given that it enforces a strict parametric shape for

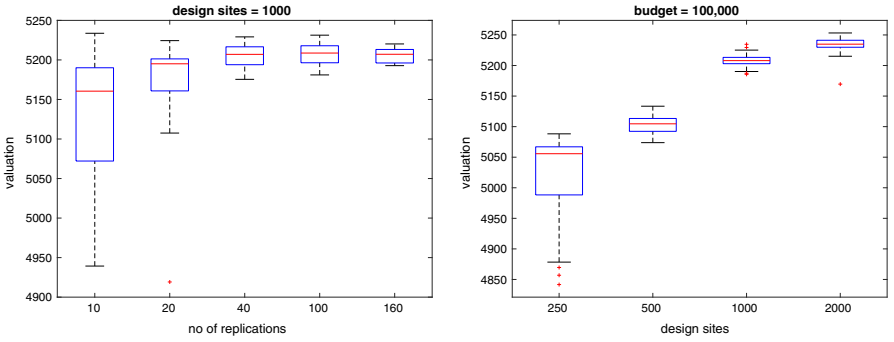
Table 2 Valuation $\hat{V}(0, 6, 1000)$ (in thousands) using different design-regression pairs and three simulation budgets: Low $N \simeq 10\,K$, Medium $N \simeq 40\,K$, Large $N \simeq 100\,K$, cf. Appendix. The valuations are averages across ten runs of each scheme except for LOESS-2D with large budget: due to the excessive overhead of LOESS only a single run was carried out

Design	Regression Scheme	Simulation budget Low	Medium	Large
Conventional	PR-1D	4,965	5,097	5,231
	GP-1D	4,968	5,107	5,247
	PR-2D	4,869	4,888	4,891
	LOESS-2D	4,910	4,969	5,011
	GP-2D	4,652	5,161	5,243
Space-filling	PR-1D	4,768	4,889	5,028
	GP-1D	4,854	5,064	5,224
	PR-2D	4,762	4,789	4,792
	LOESS-2D	4,747	4,912	4,934
	GP-2D	4,976	5,080	5,133
Adaptive 1D	PR-1D	5,061	5,187	5,246
	GP-1D	5,079	5,195	5,245
Dynamic	GP-1D	5,132	5,225	5,266
	Mixed	5,137	5,205	5,228
Mixture 2D	PR-2D	4,820	4,835	4,834
	LOESS-2D	4,960	4,987	5,003
	GP-2D	5,137	5,210	5,233

the continuation value with insufficient room for flexibility. Similarly, we observe middling performance by LOESS; on the other hand GPR generally works very well.

Turning our attention to different 2D simulation designs, we compare the conventional $\mathcal{P} \times \mathcal{G}$ choice against three alternatives: conservative space-filling $\mathcal{L}_2$ on a large input domain; joint probabilistic design $\mathcal{P}_2$; a mixture design that blends the former two. We find that both plain space-filling and joint probabilistic do not work well; the first one is not targeted enough, spending too much budget on regions that make little contribution to $\hat{V}$; the second is too aggressive and often requires extrapolation produces inaccurate predictions when computing $\widehat{m}$. In contrast, the mixture design is a winner, significantly improving upon the conventional one. In particular, GP-2D with Mixture design is the only bivariate regression scheme which performs neck-to-neck with GP-1D. This is significant because unlike GP-1D, GP-2D can be extended to higher dimensions in a straightforward manner and does not require a product design (or any interpolation which is generally slow). These findings highlight the importance of proposal density in design choice. To highlight the flexibility of our algorithm, we also present another dynamic design combining Adaptive PR-1D with Mixture GP-2D. This combination maintains the same accuracy but runs about 10-15% faster thanks to lower regression overhead of PR-1D.

Simulation designs for -1D methods. In the context of -1D regression methods that regress on P only and discretize + interpolate in I, the design needs to be of product type. We find that a probabilistic $\mathcal{P} \times \mathcal{G}$ consistently outperforms a space-filling design, such as $\mathcal{L} \times \mathcal{G}$. We can also compare the performance of Conventional PR-1D and Adaptive PR-1D (Table 2) designs that share the same $\mathcal{P}$-design in P based on the



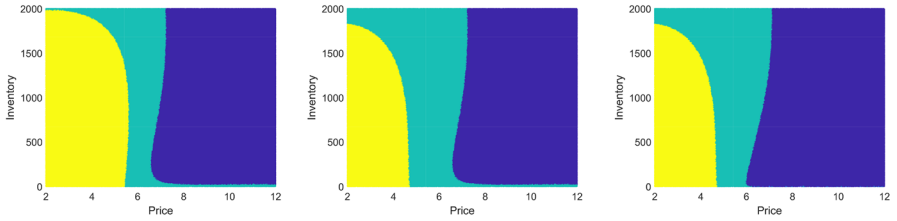
(a) Impact of replication N_b **(b)** Impact of distinct design sites N_s

Fig. 6 *Left panel:* performance of Mixture GP-2D as a function of N. We fix the number of unique design sites $N_s = 1000$ and progressively increase the number of replicates N_b, reporting the resulting $\hat{V}(0, P, I)$ at $P = 6, I = 1000$. *Right:* effect of increasing N_s for fixed $N = 10^5$. Results are for 20 runs of each algorithm. In each boxplot, the central mark represents the median, the box indicates the 25th and 75th percentiles, and the whiskers represent the most extreme runs

marginal distribution of P_t, but utilize different approaches for inventory discretization. The non-uniform discretization in the Adaptive version improves precision close to $I_{\min}$, $I_{\max}$ and $I = 1000$ and leads to a higher valuation relative to Conventional, Our take-away is that for the inventory discretization approach, one should concentrate on fine-tuning the I-mesh.

Replicated designs with GPR. Implementation of GPR also requires to manage the tradeoff between the number of design sites N_s and the replication amount N_b. The use of replication is necessitated since having more than $N_s > 2000$ distinct sites is significantly time consuming, at least for the off-shelf-implementation of GPR we used. In the left pane of Fig. 6a we consider fixing N_s and varying N_b (hence N). While larger simulation budgets obviously improve results, we note that eventually increasing N_b with N_s fixed does not improve the regression quality (although it still reduces standard error). In the right panel of Fig. 6b we present the impact of N_s for fixed total budget $N = N_b \cdot N_s$. We find that replication in general sub-optimal and better results are possible when N_s is larger (i.e. N_b is smaller). To manage the resulting speed/precision trade-off, we recommend taking $N_b \in [20, 50]$; for instance when $N = 10^5$ we use $N_s = 2000$, $N_b = 50$ and when $N = 10^4$ we use $N_s = 500$, $N_b = 20$.

Take-aways: Our experiments suggest the following key observations: (i) Among the inventory-discretized -1D methods, Gaussian process regression outperforms the standard polynomial regression in all cases, and is more robust to "poor" design or low simulation budget. (ii) Within joint -2D schemes, we continue to observe superior performance of GPR compared to the alternate bivariate regression schemes (LOESS and PR). Moreover, GP-2D is neck-in-neck with the best-performing GP-1D. We emphasize that efficient implementation of GPR and LOESS relies on batched designs which is another innovation in our DEA implementations. (iii) We also find strong dependence between choice of design and performance. We confirm that best results come from designs, such as the Mixture and Dynamic versions we implemented, that balance



(a) $m = +1$ (Inject) **(b)** $m = 0$ (Store) **(c)** $m = -1$ (Withdraw)

Fig. 7 The control maps $\widehat{m}(t, P, I, m)$ at $t = 2.7$ years for the model with switching costs in (29), for $m \in \{+1, 0, -1\}$. The colors are $\widehat{m}_{t+\Delta t} = +1$ (inject, light yellow), $\widehat{m}_{t+\Delta t} = 0$ (store, medium cyan), $\widehat{m}_{t+\Delta t} = -1$ (withdraw, dark blue). The solution used GP-2D regression with Mixture design (color figure online)

filling the input space and targeting the domain where most of $(P_t, \hat{I}_t)$ trajectories lie. Otherwise, plain space-filling or conversely aggressive boundary-following degrade performance. (iv) Between the two parametric methods PR-1D and PR-2D, we find PR-1D to significantly outperform PR-2D irrespective of the design and simulation budget, indicating the advantage of piecewise continuous regression.

6.4 Gas storage modelization with switching costs

We generalize the previous example by incorporating switching costs. Switching costs make the control map depend on the current regime m_{t_k} and induce inertia, i.e. preference to continue with the same regime so as to reduce overall costs. To handle the discrete m-dimension, we treat it as $|\mathcal{J}|$ distinct continuation functions, estimated through distinct regressions. Otherwise, the algorithm proceeds exactly the same way as before. This illustrates the flexibility of DEA to handle a range of problem formulations.

Besides the injection loss through k_5, we also add switching cost $K(i, j)$ with the following specification:

$$\begin{aligned} K(-1, 1) = K(0, 1) = 15000; \quad K(1, -1) = K(0, -1) = 5000; \\ K(1, 0) = K(-1, 0) = 0, \end{aligned} \tag{29}$$

i.e. switching cost depends only on the regime the controller decides to switch to, with switching to injection the costliest and switching to no-action free. In Fig. 7 we present the policy of the controller for different regimes. The inertia of being in regime $m_{t_k} = 0$ is evident as the corresponding control map has the widest Store region. Effect of $K(i, j)$ is also evident when comparing the Store region of left and center panels. If the controller moves from Inject to Store regime, she finds more resistance while trying to move back to injection due to the switching cost.

In Table 3 we present the performance of different design-regression pairs with a $N = 40K$ simulation budget. We dropped LOESS from this and the following case study to simplify the exposition and also because Matlab's implementation of LOESS does not support $d > 2$. As expected, introduction of the switching costs leads to lower valuation relative to Table 2. Moreover, the relative behaviour remains similar

Table 3 Valuation of a gas storage facility with switching costs $\hat{V}(0, 6, 1000)$ of different design-regression pairs with simulation budget of $N = 40,000$. Results are averages (standard deviations in brackets) across ten runs of each algorithm

Design	Regression	$\hat{V}(0, P_0, I_0)$ ('000s)
Conventional	PR-1D	4,901 (12)
	PR-2D	4,654 (11)
Space-filling	PR-1D	4,663 (16)
	GP-1D	4,757 (10)
	PR-2D	4,594 (13)
	GP-2D	4,879 (22)
Adaptive-1D	PR-1D	4,978 (14)
	GP-1D	5,058 (12)
Dynamic	PR-1D	4,997 (17)
	GP-1D	5,102 (20)
Mixture 2D	PR-2D	4,602 (30)
	GP-2D	4,978 (8)

to the previous section i.e. space-filling design has the worst performance, Mixture design observes significant improvement, but Dynamic design finally wins the race.

By comparing the valuation in Table 3 with Table 2 we may infer the impact and number of the switching costs. For example, previously Dynamic GP-1D produced valuation of $\hat{V}(0, 6, 1000) = \$5,266$ (in 1000), however, with switching cost it is now $\$5,102K$. The respective loss of $\$166K$ can be interpreted as approximately 16 regime switches on a typical trajectory (assuming $\$10K$ as an average switching cost).

Table 3 confirms the superior performance of GPR relative to PR, and the gains from using a Mixture (or even better a Dynamic) simulation design compared to the Conventional one. We remind the reader that implementing DEA in this setup is identical to the case where $K(i, j) \equiv 0$, except that a separate approximation $\hat{h}(\cdot, m)$ is constructed for each of the three levels of $m \in \mathcal{J}$. Thus, DEA immediately incorporates this extension and it is not surprising that the findings in Tables 2 and 3 are consistent with each other.

6.5 3D test case with two facilities

An important motivation for our work has been algorithm scalability in terms of the input dimension. In the classical storage problem the dimensionality is two: price P and inventory I. However, in many contexts there might be multiple stochastic factors (e.g. the power demand and supply processes in the microgrid example below) or multiple inventories. The respective problem would then be conceptually identical to those considered, except that $\mathbf{X}$ has $d \geq 3$ dimensions.

Taking up such problems requires the numerical approach to be agnostic to the dimensionality. In terms of existing methods, the piecewise continuous strategy (such as PR-1D) has been the most successful, but it relies critically on interpolating in the single inventory variable. In contrast, joint polynomial regression is trivially scalable in d but typically performs poorly. Thus, there is a strong need for other joint-d methods that can improve upon PR.

In this section we illustrate the performance of DEA with a three-dimensional state variable. To do so, we consider a joint model of two gas storage facilities, which leads to three state space variables: price P_t, inventory of first storage I_t^1 and inventory of second storage I_t^2. Each storage facility is independently controlled and operated (so that there are nine possible regimes $m_t \in \{-1, 0, 1\} \times \{-1, 0, 1\}$. Furthermore, the two facilities have identical operating characteristics, each matching those of Sect. 6.1; it follows that the total value of two such caverns is simply twice the value of a single cavern.

To implement DEA in this setup we employ a direct analogue of the previous 2D problem. Namely, we use PR and GP regressions, together with Mixture and space-filling designs, highlighting the scalability of these choices. Two secondary changes are made: (i) For the space-filling design we switch to a 3D Sobol sequence; while we do not observe any significant difference in performance between LHS and Sobol sequences for 2D problems, in 3D LHS is less stable (higher variability of $\hat{V}(0, 6, 1000, 1000)$ across runs) and yields estimates that are about \$80–100 K worse than from Sobol designs; (ii) for the mixing weights we take this time 50%/50% of sites from space-filling and from empirical distribution i.e. $\mathcal{D} = \mathcal{P}_3(0.5N) \cup \mathcal{S}_3(0.5N)$. The reason for both modifications is due to lower density of design sites per unit volume compared to previous examples , i.e. effectively lower budget. Adequate space-filling, best achieved via a QMC design, is needed to explore the relevant input space and fully learn the shape of the 3D continuation function.

In Fig. 8, we present the performance of PR-3D and GP-3D for Mixture and space-filling designs. In addition, we also implement PR-1D with conventional design, meaning that we generate a probabilistic design in P and do a two-dimensional gridded discretization (plus linear interpolation) in I^1, I^2. This approach reduces to doing N_I^2 one-dimensional polynomial regressions in P and forces to take quite low N_I, N_P values to fulfill an overall simulation budget of $N = N_P \times N_I \times N_I$, see Table 7. In contrast, -3D methods can borrow information from all N paths, drastically improving statistical efficiency. To ease the comparison, we report half of total value of the two facilities, which should ideally match the original values in Table 2. Not surprisingly, the 3D problem is harder, so for the same simulation budget the reported valuations are lower. For example, at 10^4 budget, Adaptive GP-3D obtains a valuation \$251K below that of Mixture GP-2D; this gap declines to \$109 K as we increase the simulation budget to $N = 10^5$. Moreover, difference between the performance of Mixture and space-filling design is evident even with polynomial regression (Sobol PR vs. Mixture PR). Mixture design with GP-3D further improves the valuation; we observe difference of over \$300 K comparing Mixture PR and Mixture GP at $N = 10^5$ budget. Consistent with previous examples, we again find that the valuation from conventional PR-1D is between the valuation obtained via PR-3D and Mixture GP-3D. However, the significantly larger standard errors of PR-1D is a sign of deteriorating stability of inventory discretization in the increased dimension, and highlight the limited scope of that technique. The take-away is that the gains from fine-tuning the DEA components grow as problem complexity increases. The emulation paradigm further suggests that non-parametric/adaptive approaches will be best able to maximize performance in "hard" contexts.

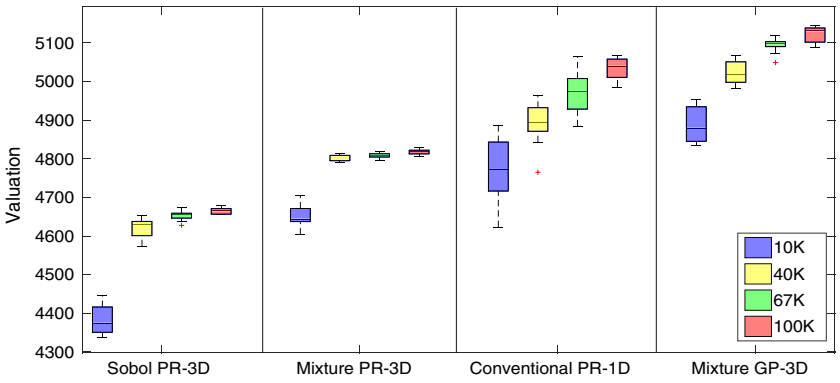


Fig. 8 Half of estimated value $\hat{V}(0, 6, 1000, 1000)/2$ for the 3D example with two storage caverns from Sect. 6.5. Results are for ten runs of each algorithm across four different simulation budgets $N \in \{10\,K, 40\,K, 67\,K, 100\,K\}$. Description of the boxplots is same as in Fig. 6

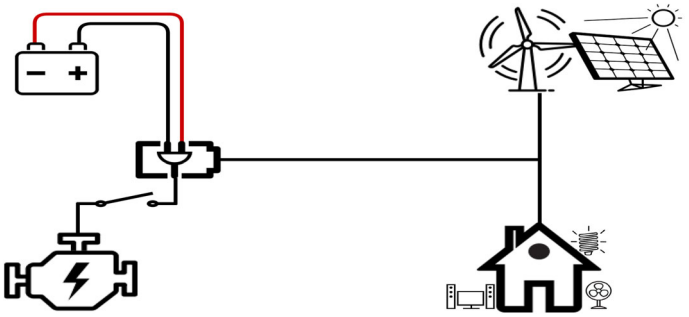


Fig. 9 Microgrid topology

7 Microgrid balancing under stochastic net demand

In this example, we use the framework of Sect. 2 in the context of a Microgrid, which is a scaled-down version of a power grid comprising of renewable energy sources, a diesel generator, and a battery for energy storage. The microgrid could be isolated or connected to a national grid and the objective is to supply electricity at the lowest cost by efficiently utilizing the diesel generator and the battery to match demand given the intermittent power production from the renewable sources. The topology of the microgrid considered here is similar to that in [2] and presented in Fig. 9.

The exogenous factor corresponds to residual demand $X_{t_k} = L_{t_k} - R_{t_k}$, where L_{t_k}, R_{t_k} are the demand and output from the renewable source, respectively. We assume that the microgrid controller bases his policy only on X, modeled as a discrete Ornstein–Uhlenbeck process:

$$X_{t_{k+1}} - X_{t_k} = \alpha(\underline{X} - X_{t_k})\Delta t + \sigma \Delta W_{t_k}, \qquad \Delta W_{t_k} \sim \mathcal{N}(0, \Delta t). \tag{30}$$

On the supply side, the controller has two resources: a battery (energy storage) and a diesel generator. The state of the battery is denoted by $I_t \in [0, I_{\max}]$ with dynamics given by

$$I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\Delta t = I_{t_k} + B_{t_k}\Delta t, \tag{31}$$

where $a(c_{t_k})$ is interpreted as battery output, driven by c_{t_k} the diesel output. The latter has two regimes $m_{t_k} \in \{0, 1\}$. In the OFF regime $m_{t_{k+1}} = 0$, $c_{t_k}(0) = 0$. When the diesel is ON, $m_{t_{k+1}} = 1$, its power output is state dependent and given by:

$$c_{t_k}(1) = X_{t_k}\mathbf{1}_{\{X_{t_k}>0\}} + B_{\max} \wedge \frac{I_{\max} - I_{t_k}}{\Delta t}, \tag{32}$$

where $B_{\max} > 0$ is the maximum power input to the battery. The power output from the battery is the difference between the residual demand and the diesel output, provided it remains within the physical capacity constraints $[0, I_{\max}]$ of the battery:

$$B_{t_k} := a(c_{t_k}) = -\frac{I_{t_k}}{\Delta t} \vee \left(B_{\min} \vee (c_{t_k} - X_{t_k}) \wedge B_{\max}\right) \wedge \frac{I_{\max} - I_{t_k}}{\Delta t}. \tag{33}$$

To describe the cost structure, define an imbalance process $S_t = S(c_t, X_t)$:

$$S_{t_k} = c_{t_k} - X_{t_k} - B_{t_k}. \tag{34}$$

Normally the imbalance is zero, i.e. the battery absorbs the difference between production and demand. $S_{t_k} < 0$ implies insufficient supply of power resulting in a *blackout*; $S_{t_k} > 0$ leads to curtailment or waste of energy. We penalize both scenarios asymmetrically using costs $C_{1,2}$ for curtailment and blackout, taking $C_2 \gg C_1$ in order to target zero blackouts:

$$\pi(c, X) := -c^\gamma - |S|\Big[C_2\mathbf{1}_{\{S<0\}} + C_1\mathbf{1}_{\{S>0\}}\Big]. \tag{35}$$

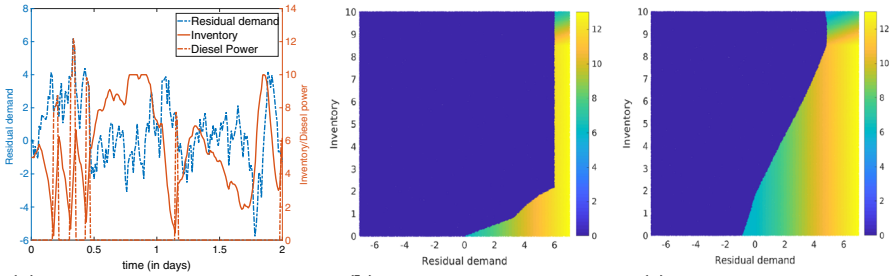
More discussion on the choice of this functional form can be found in [2,20]. Furthermore, starting the diesel generator when it is OFF incurs a switching cost $K(0, 1) = 10$, but no cost is incurred to switch off the diesel generator, $K(1, 0) = 0$. The final optimization problem is starting from state $(X_{t_k}, I_{t_k}, m_{t_k})$ and observing the residual demand process $\mathbf{X}_{t_k}$, to maximize the pathwise value following the policy $\mathbf{m}_{t_k}$, exactly like in (6).

7.1 Optimal microgrid control

The parameters we use are given in Tables 4 and 5. For the terminal condition we again force the controller to return the microgrid with at least the initial battery charge: $W(X_T, I_T) = -200\max(I_0 - I_T, 0)$. The effect of this penalty is different compared to the gas storage problem in Sect. 6. Because the controller can only partially control the inventory, we end up with $\hat{I}_T \in [I_0, I_{\max}]$. We use simulation budget of $N = 10,000$ and out-of-sample budget of $N' = 200,000$. For simplicity of implementation,

Table 4 Parameters for the Microgrid in Sect. 7

$\alpha = 0.5,\ \underline{X} = 0,\ \sigma = 2$
$I_{\max} = 10\ (\text{kWh}),\ B_{\min} = -6, B_{\max} = 6\ (\text{kW}),\ K(0,1) = 10,\ K(1,0) = 0$
$C_1 = 5, C_2 = 10^6,\ \gamma = 0.9,\ T = 48\ (\text{hours}),\ \Delta t = 0.25\ (\text{hours})$



(a) Pathwise trajectory **(b)** Generator OFF $(m = 0)$ **(c)** Generator ON $(m = 1)$

Fig. 10 *Left panel:* trajectory of the residual-demand (X_t), corresponding to policy (c_t) and the resultant inventory trajectory $(\hat{I}_t)$. *Middle and right panels:* the control policy $\hat{c}(t, X, I, m)$ at $t = 24$ h. Recall that $c(0) = 0$ whenever the diesel is OFF. All panels are based on GP-2D regression and Mixture design $\mathcal{D} = \mathcal{P}_2(0.5\,N) \cup \mathcal{L}_2(0.5\,N)$

we use same simulation designs across both m-regimes (i.e. $\mathcal{D}_k$ is independent of m, cf. Algorithm 1).

Figure 10a illustrates the computed policy $(\widehat{m}_t)$ of the microgrid controller for a given path of residual demand (X_t). The left panel plots the joint trajectory of demand $X_{0:T}$ (left y-axis), inventory $\hat{I}_{0:T}$ and diesel output $c_{0:T}$ (both right y-axis). The diesel is generally off; the controller starts the diesel generator whenever the residual demand X_{t_k} is large, or the inventory I_{t_k} is close to empty. When the generator is on, the battery gets quickly re-charged according to (32); otherwise $\hat{I}$ tends to be decreasing, unless $X_{t_k} < 0$. The center and right panels of the Figure visualize the resulting policy $c(t, X, I, m) = c(\widehat{m}(t, X, I, m))$. Due to the effect of the switching cost, when the generator is ON (Fig. 10c), it continues to remain ON within a much larger region of the state space compared to when it is OFF.

7.2 Numerical results

Figure 11 shows the estimated value $\hat{V}(0, 0, 5, 1)$ of the microgrid across different designs and regression methods at $N = 10,000$. Recall that in this setup, the controller only incurs costs so that $\hat{V} < 0$ and smaller (costs) is better. The relative performance of the schemes remains similar to Sect. 6. We continue to observe lower performance of space filling designs across regression methods. However, GP is more robust to this design change compared to traditional regression methods. Moreover, GPR dramatically improves upon PR-2D (whose performance is so bad it was left off Fig. 11). Adaptive design with GP-2D once again produces the highest valuation (lowest cost), and substantially improves upon PR-1D.

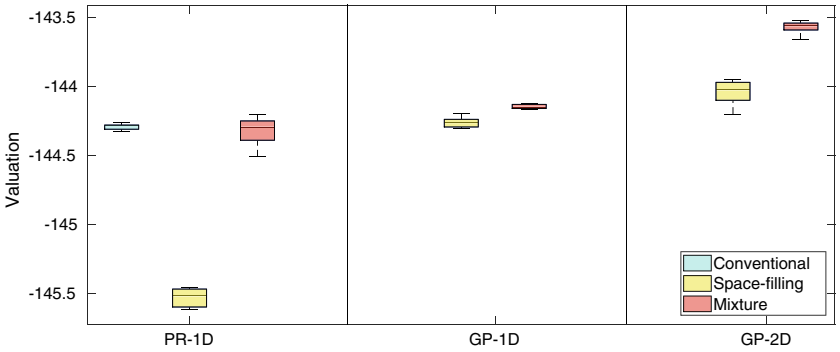


Fig. 11 Estimated value $\hat{V}(0, 0, 5, 1)$ for the microgrid example and different design-regression pairs. Results are for 10 runs of each algorithm. For comparison, PR-2D estimated mean valuation was significantly lower at -153.6, -156.4, -152.3 for Conventional, Space-filling and Mixture designs, respectively. Description of the boxplots is the same as in Fig. 6

7.3 Additional applications

The DEA is applicable in numerous other contexts; in this section we mention some additional examples.

Generalized microgrid. The microgrid example discussed above is highly simplified. More realistic models would consider separate stochastic factors for the different components (e.g. renewable output (R_t) and demand load (L_t)). Moreover, the controller can typically control both the diesel generator, as well as the battery, adding further states to $\mathcal{M}$. In a long-term planning context, battery degradation due to repeated charge/discharge becomes important and needs to be captured by an additional "age" variable. Thus, an industrial-grade implementation of microgrid management would call for a high-dimensional formulation $d \gg 4$, where DEA aspects such as choice of design become critical for performance.

Commodity extraction. The problem of managing a mine that contains a finite amount of extractable commodity is well studied and closely matches the stochastic storage framework. The typical regimes include { Open, Close, Abandon }, where Open corresponds to extraction and sale at the stochastic spot price P_t, Close corresponds to temporary mothballing that still carries fixed (maintenance) costs, and Abandon is a permanent shut-down. See for example the recent work [23] where the authors apply DPP with deterministic piecewise-linear approximation.

Portfolio optimization. Another application of DEA could be for portfolio optimization, with inventory corresponding to the current wealth that is driven by the investment strategy c_t. The objective is to maximize expected utility of terminal wealth, typically subject to various market and investment constraints. The typical state is then the asset price P_t and the current wealth I_t. Zhang et al. [40] recently used the control randomization approach for this problem.

Robot/drone motion control. Dynamic path optimization is a classical engineering problem that is frequently formulated as a stochastic switching-type control. The robot motion is subject to random shocks (e.g. wind disturbance for an unmanned aerial

vehicle) and the control is specified in terms of simple instructions such as { Left, Right, Straight}. Objectives might include target-tracking, obstacle avoidance, fuel use optimization, etc. The robot location/speed are the endogenous state components, while the external shocks (wind speed, obstacle motion) are the stochastic factors. See, e.g. [3,34] for approaches where a modification of DEA could be beneficial.

8 Conclusion and future work

The developed DEA template generalizes the existing methodologies used in the sphere of RMC methods for stochastic storage problems. The modularity of DEA allows a wide range of modifications that can enhance current state-of-the-art and improve scalability. In particular, we show several combinations of approximation spaces and simulation designs that are as good or better than any benchmarks (using both -1D and -2D approaches) reported in the literature. Emphasizing the experimental design aspect we show that there is wide latitude in removing memory requirements of traditional RMC by eliminating the need to simulate global paths. Similarly, non-parametric regression approaches like GPR, minimize the concern of picking "correct" basis functions. Furthermore, we stress the possibility to mix and match different methods. As an example, we illustrated DEA-based valuation of a gas storage facility using different designs, regressions, and budgets across the time-steps.

A natural extension is to consider the setting where the injection/withdrawal rates are continuous. In that case, one must optimize over c_t, replacing the arg max operator with a bonified arg sup over the admissible control set $c \in \mathcal{A}$. Depending on how switching costs are assessed, one might eliminate the regime m_t entirely, or have a double optimization

$$V(t, P_t, I_t, m_t) = \max_{m \in \mathcal{J}} \Big\{ \sup_{c \in \mathcal{A}(P_t, I_t, m)} \pi(P_t, c)\Delta t + q(t, P_t, I_{t+\Delta t}(c), m) - K(m_t, m) \Big\}.$$

This could be interpreted as solving a no-bang-bang switching model, which would arise when there is some nonlinear link between profit π and c, or a nonlinear effect of c on $I_{t+\Delta t}$. Numerically carrying out the inner optimization over c calls for joint regression schemes in order that $\hat{q}(t, \cdot)$ is smooth in $I_{t+\Delta t}$.

A further direction afforded by our template is to move to look-ahead strategies using Remark 2 for generation of pathwise continuation values. By taking $w > 1$ one may interpolate between the Tsitsiklis–van Roy approach and the Longstaff-Schwartz one, ideally via a data-driven scheme that adaptively selects the look-ahead at each time step. To this end, GP regressions results can be used to quantify the single-step projection errors in $\hat{q}(t_k, \cdot)$.

In a different vein, one may consider modifications where the autonomous/endogenous dichotomy between (P_t) and (I_t) is blurred. For example, a variant of the storage problem arises in the context of hydropower operations, i.e. controlling a double dammed reservoir that receives inflows from upstream and can release water downstream [1,10]. Moreover, the reservoir experiences evaporation and/or natural drawdown, irrespective of the operations. In this setup, the inventory I_t experiences stochastic shocks, either due to random inflows (due to precipitation) or random outflows (due to temperature-based evaporation, etc). Therefore, I_{t+1} is a function of both I_t, c_t, and some outside noise (or factor) O_t. Moreover, if the dam is large, hydropower management has endogenous stochastic risk, i.e. the control c_{t_k} also affects the distribution of the price process $P_{t_{k+1}}$ by modifying the regional supply of energy and hence affecting the supply-demand equilibrium that drives changes in (P_t).

Appendix: Design specifications

Table 5 Design construction for different methods in Table 2 of Sect. 6.1

Design	Regression	Low	Medium	High
Conventional	PR-1D/-2D ($N_P \times N_I$)	1050×10	2100×20	3400×30
	GP-1D/-2D, LOESS ($N_s \times N_b \times N_I$)	$105 \times 10 \times 10$	$210 \times 10 \times 20$	$340 \times 10 \times 30$
Space-filling	PR-1D ($N_P \times N_I$)	1050×10	2100×20	3400×30
	GP-1D ($N_s \times N_b \times N_I$)	$105 \times 10 \times 10$	$210 \times 10 \times 20$	$340 \times 10 \times 30$
	PR-2D (N)	10500	42000	102000
	LOESS/GP-2D ($N_s \times N_b$)	500×21	1000×42	2000×51
Adaptive-1D	PR-1D ($N_P \times N_I$)	950×11	2000×21	3300×31
	GP-1D ($N_s \times N_b \times N_I$)	$95 \times 10 \times 11$	$200 \times 10 \times 21$	$330 \times 10 \times 31$
Mixture-2D	PR-2D (N)	10500	42000	102000
	LOESS/GP-2D ($N_s \times N_b$)	500×21	1000×42	2000×51
Dynamic	GP-1D ($N_{(2)}$)	$150 \times 10 \times 21$	$340 \times 10 \times 31$	$440 \times 10 \times 41$
	($N_{(1)}$)	$74 \times 10 \times 11$	$168 \times 10 \times 21$	$300 \times 10 \times 31$
	PR-1D + GP-2D ($N_{(2)}$)	2000×21	3400×31	4400×41
	($N_{(1)}$)	500×21	1000×42	2000×51

Table 6 Discretized inventory levels used for Adaptive design for -1D methods in Table 2

N_I	Specification
11	[0:100:200, 500:250:1500, 1800:100:2000]
21	[0:50:200, 400:200:800, 900:50:1100, 1200:200:1600, 1800:50:2000]
31	[0:25:100, 150, 200:100:900, 950:50:1050, 1100:100:1800, 1850, 1900:25:2000]
else	Uniformly spaced

Table 7 Design specifications for different DEA implementations of Sect. 6.5 in Fig. 8

Design	Regression	10 k	40 k	67 k	100 k
Sobol/adaptive	PR-3D (N)	10500	42000	67500	102000
	GP-3D ($N_s \times N_b$)	500×21	1000×42	1500×45	2000×51
Conventional	PR-1D ($N_P \times N_I^2$)	215×7^2	347×11^2	400×13^2	450×15^2

References

1. Alais, J.C., Carpentier, P., De Lara, M.: Multi-usage hydropower single dam management: chance-constrained optimization and stochastic viability. Energy Syst. **8**(1), 7–30 (2017)
2. Alasseur, C., Balata, A., Ben Aziza, S., Maheshwari, A., Tankov, P., Warin, X.: Regression Monte Carlo for Microgrid Management. Tech. Rep., (2018). arXiv:1802.10352
3. Balata, A., Palczewski, J.: Regress-Later Monte Carlo for optimal control of Markov processes. Tech. Rep., (2017). arXiv:1712.09705
4. Balata, A., Palczewski, J.: Regress-Later Monte Carlo for Optimal Inventory Control with applications in energy. Tech. Rep., (2017). arXiv:1703.06461
5. Bäuerle, N., Riess, V.: Gas storage valuation with regime switching. Energy Syst. **7**(3), 499–528 (2016)
6. Binois, M., Gramacy, R.B., Ludkovski, M.: Practical heteroscedastic gaussian process modeling for large simulation experiments. J. Comput. Gr. Stat. **27**(4), 808–821 (2018)
7. Boogert, A., de Jong, C.: Gas storage valuation using a Monte Carlo method. J. Deriv. **15**(3), 81–98 (2008)
8. Boogert, A., de Jong, C.: Gas storage valuation using a multi-factor price process. J. Energy Markets **4**, 29–52 (2011)
9. Bouchard, B., Warin, X.: Monte Carlo valuation of American options: Facts and new algorithms to improve existing methods. In: Carmona, R.A., Del Moral, P., Hu, P., Oudjane, N. (eds.) Numerical Methods in Finance: Bordeaux, June 2010, pp. 215–255. Springer, Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Carmona, R., Ludkovski, M.: Valuation of energy storage: an optimal switching approach. Quant. Financ. **10**(4), 359–374 (2010)
11. Chen, Z., Forsyth, P.A.: A semi-Lagrangian approach for natural gas storage valuation and optimal operation. SIAM J. Sci. Comput. **30**(1), 339–368 (2008)
12. Cong, F., Oosterlee, C.: Multi-period meanvariance portfolio optimization based on Monte Carlo simulation. J. Econo. Dyn. Control **64**, 23–38 (2016)
13. Davison, M., Zhao, G.: Optimal control of two-dam hydro facility. Syst. Eng. Procedia **3**, 1–12 (2012)
14. Denault, M., Simonato, J.G., Stentoft, L.: A simulation-and-regression approach for stochastic dynamic programs with endogenous state variables. Comput. Oper. Res. **40**(11), 2760–2769 (2013)
15. Egloff, D.: Monte Carlo algorithms for optimal stopping and statistical learning. Ann. Appl. Probab. **15**(2), 1396–1432 (2005)
16. Egloff, D., Kohler, M., Todorovic, N.: A dynamic look-ahead monte carlo algorithm for pricing bermudan options. Ann. Appl. Probab. **17**(4), 1138–1171 (2007)

17. Felix, B.J., Weber, C.: Gas storage valuation applying numerically constructed recombining trees. Eur. J. Oper. Res. **216**(1), 178–187 (2012)
18. Gramacy, R.B., Ludkovski, M.: Sequential design for optimal stopping problems. SIAM J. Financ. Math. **6**(1), 748–775 (2015)
19. Han, J., E, W.: Deep Learning Approximation for Stochastic Control Problems. Tech. Rep., (2016). arXiv:1611.07422
20. Heymann, B., Bonnans, J.F., Martinon, P., Silva, F.J., Lanas, F., Jiménez-Estévez, G.: Continuous optimal control approaches to microgrid energy management. Energy Syst. **9**(1), 59–77 (2018)
21. Heymann, B., Bonnans, J.F., Silva, F., Jimenez, G.: A stochastic continuous time model for microgrid energy management. In: 2016 European Control Conference (ECC), pp. 2084–2089 (2016)
22. Heymann, B., Martinon, P., Bonnans, F.: Long term aging : an adaptative weights dynamic programming algorithm. Tech. Rep., hal 01349932 (2016)
23. Hinz, J., Tarnopolskaya, T., Yee, J.: Efficient algorithms of pathwise dynamic programming for decision optimization in mining operations. Ann. Oper. Res. (2018). https://doi.org/10.1007/s10479-018-2910-3
24. Jain, S., Oosterlee, C.W.: Pricing high-dimensional Bermudan options using the stochastic grid method. Int. J. Comput. Math. **89**(9), 1186–1211 (2012)
25. Jain, S., Oosterlee, C.W.: The stochastic grid bundling method: efficient pricing of Bermudan options and their Greeks. Appl. Math. Comput. **269**, 412–431 (2015)
26. Kharroubi, I., Langrené, N., Pham, H.: A numerical algorithm for fully nonlinear HJB equations: an approach by control randomization. Monte Carlo Meth. Appl. **20**, 145–165 (2014)
27. Langrené, N., Tarnopolskaya, T., Chen, W., Zhu, Z., Cooksey, M.: New regression Monte Carlo methods for high-dimensional real options problems in minerals industry. In: 21st International Congress on Modelling and Simulation, pp. 1077–1083 (2015)
28. Longstaff, F.A., Schwartz, E.S.: Valuing American options by simulation: a simple least-squares approach. Rev. Financ. Stud. **14**(1), 113–147 (2001)
29. Ludkovski, M.: Kriging metamodels and experimental design for Bermudan option pricing. J. Comput. Financ. **22**(1), 37–77 (2018)
30. Malyscheff, A.M., Trafalis, T.B.: Natural gas storage valuation via least squares Monte Carlo and support vector regression. Energy Syst. **8**(4), 815–855 (2017)
31. Mazieres, D., Boogert, A.: A radial basis function approach to gas storage valuation. J. Energy Mark. **6**(2), 19–50 (2013)
32. Nadarajah, S., Margot, F., Secomandi, N.: Comparison of least squares Monte Carlo methods with applications to energy real options. Eur. J. Oper. Res. **256**(1), 196–204 (2017)
33. Pham, H.: Optimal switching and free boundary problems. In: Continuous-time stochastic control and optimization with financial applications, pp. 95–137. Springer, Berlin, Heidelberg (2009)
34. Quintero, S.A.P., Ludkovski, M., Hespanha, J.P.: Stochastic optimal coordination of small uavs for target tracking using regression-based dynamic programming. J Intell. Robot. Syst. **82**(1), 135–162 (2016)
35. Thompson, M.: Natural gas storage valuation, optimization, market and credit risk management. J. Commod. Mark. **2**(1), 26–44 (2016)
36. Thompson, M., Davison, M., Rasmussen, H.: Natural gas storage valuation and optimization: a real options application. Naval Res. Logist. (NRL) **56**(3), 226–238 (2009)
37. Tsitsiklis, J.N., Van Roy, B.: Regression methods for pricing complex American-style options. Trans. Neural Netw. **12**(4), 694–703 (2001)
38. Van-Ackooij, W., Warin, X.: On conditional cuts for Stochastic Dual Dynamic Programming. Tech. Rep., (2017). arXiv:1704.06205
39. Warin, X.: Gas storage hedging. In: Carmona, R.A., Del Moral, P., Hu, P., Oudjane, N. (eds.) Numerical methods in finance: Bordeaux, June 2010, pp. 421–445. Springer, Berlin, Heidelberg (2012)
40. Zhang, R., Langrené, N., Tian, Y., Zhu, Z., Klebaner, F., Hamza, K.: Dynamic Portfolio Optimization with Liquidity Cost and Market Impact: A Simulation-and-Regression Approach. Tech. Rep., (2017). arXiv:1610.07694
41. Zhao, G., Davison, M.: Optimal control of hydroelectric facility incorporating pump storage. Renew. Energy **34**(4), 1064–1077 (2009)