

MathematicS
MathS in A.
In Action

MIKE LUDKOVSKI

Regression Monte Carlo for Impulse Control

Volume 11 (2022), p. 73-90.

<https://doi.org/10.5802/msia.18>

© Les auteurs, 2022.



Cet article est mis à disposition selon les termes de la licence CREATIVE COMMONS ATTRIBUTION 4.0.

<http://creativecommons.org/licenses/by/4.0/>



MathematicS In Action est membre du
Centre Mersenne pour l'édition scientifique ouverte

<http://www.centre-mersenne.org/>

e-ISSN : 2102-5754

Regression Monte Carlo for Impulse Control

MIKE LUDKOVSKI *

* Department of Statistics and Applied Probability, University of California, Santa Barbara
CA 93106-3110, USA
E-mail address: ludkovski@pstat.ucsb.edu.

Abstract

I develop a numerical algorithm for stochastic impulse control in the spirit of Regression Monte Carlo for optimal stopping. The approach consists in generating statistical surrogates (aka functional approximators) for the continuation function. The surrogates are recursively trained by empirical regression over simulated state trajectories. In parallel, the same surrogates are used to learn the intervention function characterizing the optimal impulse amounts. I discuss appropriate surrogate types for this task, as well as the choice of training sets. Case studies from forest rotation and irreversible investment illustrate the numerical scheme and highlight its flexibility and extensibility. Implementation in R is provided as a publicly available package posted on GitHub.

1. Introduction

Stochastic impulse control is concerned with systems where the state process (X_t) is subject to stochastic dynamics as well as repeated lumpy interventions or shocks by the controller. Such impulses make an instantaneous, rather than sustained, impact on (X_t) and carry an instantaneous cost/reward. The goal of the controller is to maximize total expected (discounted) profit that is driven by the impulses and the continuous revenue function $\pi(X_t)$. Impulse control problems have a long history with manifold applications over the past 40+ years, see below. In particular, impulsive controls are common in commodities applications to describe management of natural resources or industrial capacity planning. Nevertheless, numerical methods for stochastic impulse control are surprisingly thinly studied, especially in comparison to the vast literature on numerical optimal stopping, and the emergent literature on numerical continuous control via Neural Networks.

In this article I propose to leverage Monte Carlo based methods that have been developed for optimal stopping in order to create a related approach to stochastic impulse control. To do so, I rely on the interpretation of impulse control as a two-stage decision making; at each time instant, the controller must first decide whether to act or to wait; conditional on acting, in the second stage the controller picks the optimal action. This perspective reduces impulse control to repeated optimal stopping with an implicit payoff function specified via the so-called intervention operator \mathcal{M} . In turn, it permits to import algorithms for multiple stopping problems after incorporating the computation of the intervention operator. Through this lens, solvers for impulse control can be built on top of related code for optimal stopping.

The proposed algorithm extends the realm of Regression Monte Carlo (RMC) methods to the setting of impulse control. It employs the main features of RMC (statistical surrogates for a functional approximation of the continuation value and simulation for empirical training of these surrogates) in the context of multiple impulsing actions. Additionally, I propose direct optimization of the surrogate over the action set to obtain optimal impulses. Methodologically, this strategy highlights the advantageous modularity of RMC which makes the paradigm applicable

Work partially supported by NSF DMS-1736439 and DMS-1821240.

Keywords: Impulse Control, Statistical Surrogates, Irreversible Investment.

2020 Mathematics Subject Classification: 93E25, 65C05, 49N25.

beyond classical stop/continue decisions. On the implementation side, the algorithm is coded in R and is integrated into the existing `m1OSP` “Machine Learning for Optimal Stopping Problems” package developed by the author over the past few years [29]. Thus, `m1OSP` effectively subsumes numerical resolution of impulse control into the existing framework of RMC. The package is publicly available via GitHub at github.com/mludkov/m1OSP and offers reproducible vignettes. Thus, all the underlying code and case studies can be fully examined by the readers or other researchers, facilitating reproducibility and future extensions.

1.1. Literature Review

Relative to other types of stochastic control impulse control problems are rarely solvable explicitly, with only a few exceptions, see [2, 17, 19]. In part, this is because there are many problem ingredients to solve for: impulse thresholds, impulse targets, intervention function, value function, etc. Thus, typically only very special cases, such as time-stationary models with linear intervention costs and linear dynamics, have been studied in detail.

The most well known sub-case is when the state (X_t) is one dimensional and is expected to be a renewal process when optimally controlled. The so-called (s, S) strategies focus on determining an impulse threshold s and a target level S and reduce computation of optimal strategy to a two-dimensional optimization over the two constants s, S . (s, S) strategies have been studied in Operations Research for over 20 years, formulations similar to the one I discuss have appeared in optimal inventory [11, 14, 26] and dividend payout problems [7, 10, 18].

Another major application of stochastic impulse control has been in *real options*, in the context of (ir)reversible investment [1, 3, 23]. Alternatively called the capacity expansion problem, this class of models considers gradual addition of capacity, e.g. power generation capacity in the context of owning a fleet of electricity power plants. More sophisticated models [13, 24] treat separately the commodity price and the current capacity, leading to a two-dimensional impulse control formulation, with one exogenous and one endogenous component, see Section 4.2. A kind of a conceptual opposite to investment are *harvesting* problems, especially the Faustmann problem of forest management [4, 5, 12]. The state variable represents the current forest stand value; actions correspond to cutting trees down for sale, known as a “rotation”. Thus impulses are down and are viewed as revenue rather than cost. I discuss the Faustmann problem further in Section 4.1. Other applied domains include control of foreign exchange rates by a central bank [15] and management of energy retail prices [9].

The standard approach to numerical solution of impulse control is via quasi-variational inequalities, that reduce to a HJB-type partial differential equation. However, the non-local term in the equation that corresponds to the impulses makes numerical schemes quite challenging. So far there is limited literature available to handle it, [8]. Indeed, eliminating this limitation of the HJB approach is one of the gaps I aim to address in the present publication that takes a completely probabilistic/statistical perspective and does not depend on the smoothness of the value function. Similarly, due to the associated analytic difficulties, the vast majority of works consider time-stationary one-dimensional impulse control on infinite horizon, see [12] and [21, 22] for recent results on finite-horizon SIC. Multivariate SIC is treated in [7, 16] among others.

The rest of the paper is organized as follows. Section 2 introduces the problem and the new RMC-based algorithm. Section 3 discusses implementation, including how to evaluate optimal impulses. That Section concludes with a concrete example, illustrated with R code snippets. Section 4 contain two case studies, about forest rotation (Section 4.1) and two-dimensional capacity expansion (Section 4.2).

2. Problem Formulation

To set the stage, we focus on diffusion models where the system state (X_t) in the absence of any impulses is assumed to satisfy a Stochastic Differential Equation of Itô type,

$$dX_t = \mu(X_t) dt + \sigma(X_t) dW_t, \quad (2.1)$$

where (W_t) is a (multi-dimensional) Brownian motion and the drift $\mu(\cdot)$ and volatility $\sigma(\cdot)$ are smooth enough to yield a unique strong solution to (2.1). The state space of (X_t) is $\mathcal{X} \subset \mathbb{R}$ and we assume the standard probabilistic structure of $(\Omega, \mathcal{F}, (\mathcal{F}_t), \mathbb{P})$, where X_t is adapted to the filtration $\mathbb{F} = (\mathcal{F}_t)$. Generalization to multiple dimensions is straightforward.

The above is the uncontrolled dynamics, which are subject to control shocks. To describe the latter, let \mathfrak{A} be the set of all admissible controls. An admissible control $A = \{\tau_n, z_n\}$ is a double sequence such that

- τ_n is an increasing sequence of \mathbb{F} -stopping times, such that $\tau_n < \tau_{n+1}$ \mathbb{P} -a.s. and $\lim_{n \rightarrow \infty} \tau_n = T$ \mathbb{P} -a.s.
- z_n is a sequence of random variables taking values in $\mathcal{Z} \subset \mathbb{R}$ such that z_n is $\mathcal{F}_{\tau_n^-}$ -measurable for every $n \geq 1$
- Integrability condition is satisfied $\sum_n \mathbb{E}[e^{-r\tau_n}(1 + z_n)] < \infty$ which ensures finiteness of the discounted interventions.

A controlled process $X^{t,x,A}$ is indexed by its initial condition $X_t = x$ and the associated admissible impulse strategy and satisfies for $s > t$

$$X_s^{t,x,A} = x + \int_t^s \mu(X_r^{t,x,A}) dr + \int_t^s \sigma(X_r^{t,x,A}) dW_r + \sum_{n:t < \tau_n \leq r} z_n. \quad (2.2)$$

The corresponding expectation conditional on $X_t = x$ is denoted by $\mathbb{E}_{t,x}[\cdot]$.

The dynamics in (2.1) then correspond to the case of no control: $A = \emptyset$, and one may view (2.2) as the concatenation of the uncontrolled (2.1) on each $[\tau_n, \tau_{n+1})$ plus the instantaneous jumps $X_{\tau_n^+}^{t,x,A} = X_{\tau_n^-}^{t,x,A} + z_n$.

Let $\pi : x \mapsto \mathbb{R}$ be the running reward function, and $\kappa : (x, z) \mapsto \mathbb{R}$ be the impulse cost function, representing the net revenue of applying impulse of size $z \in \mathbb{R}$ at time t and state x . Typically $\kappa(x, z) < 0$, capturing the cost of moving X_t from $x \in \mathcal{X}$ to $x + z \in \mathcal{X}$. We assume a finite horizon T with a respective terminal condition $\phi(X_T)$. The controller aims to maximize discounted expected reward

$$\mathbb{E} \left[\int_0^T e^{-rt} \pi(X_t^{0,x,A}) dt + \sum_{n:\tau_n < T} e^{-r\tau_n} \kappa(X_{\tau_n^-}^{0,x,A}, z_n) + e^{-r(T-t)} \phi(X_T^{0,x,A}) \right] \rightarrow \max! \quad (2.3)$$

on the horizon $T < \infty$, where $r \geq 0$ is the discount factor. Above we assume that π, ϕ are such that $\mathbb{E}_{0,x} \left[\int_0^T e^{-rt} |\pi(X_t^\emptyset)| dt + |\phi(X_T^\emptyset)| \right] < \infty$.

For an admissible strategy A , denote by

$$J_{t,T}(x; A) := \int_t^T e^{-r(s-t)} \pi(X_s^{t,x,A}) ds + \sum_{n:\tau_n < T} e^{-r(\tau_n-t)} \kappa(X_{\tau_n^-}^{t,x,A}, z_n) + e^{-rT} \phi(X_T^{t,x,A}). \quad (2.4)$$

the reward from applying the strategy A on the interval $[t, T]$ and starting with $X_t = x$. Then our goal is to evaluate the **value function** $V : [0, T] \times \mathcal{X} \rightarrow \mathbb{R}$,

$$\begin{aligned} V(t, x) &:= \sup_{A \in \mathfrak{A}_t} \mathbb{E}_{t,x} \left[\int_t^T e^{-rs} \pi(X_s) ds + \sum_n e^{-r(\tau_n-t)} \kappa(X_{\tau_n^-}, z_n) + e^{-r(T-t)} \phi(X_T) \right] \\ &= \sup_{A \in \mathfrak{A}_t} \mathbb{E} [J_{t,T}(x; A)]. \end{aligned} \quad (2.5)$$

The infinitesimal generator of the uncontrolled X^\emptyset is

$$\mathcal{L}u(t, x) := ru(t, x) - \mu(x) \frac{\partial u(t, x)}{\partial x} - \frac{1}{2} \sigma^2(x) \frac{\partial^2 u}{\partial x^2}(t, x) - \frac{\partial u(t, x)}{\partial t}.$$

Also define the intervention operator

$$\mathcal{M}u(t, x) := \sup_{z \in \Xi} \{u(t, x + z) - \kappa(x, z)\}.$$

Optimality for the controller's actions implies that $V(t, x) \geq \mathcal{M}V(t, x)$ for all (t, x) . At the same time, Ito's lemma implies that $\mathcal{L}V(t, x) \geq \pi(x)$. Putting the two together yields the quasi-variational inequality (QVI)

$$\min(\mathcal{L}V(t, x) - \pi(x), V(t, x) - \mathcal{M}V(t, x)) = 0 \quad (2.6)$$

with the boundary condition $V(T, x) = \phi(x)$. The analytic approach then characterizes V as the viscosity solution of the QVI (2.6), see e.g. Chapter 6 in Oksendal and Sulem [30]. HJB-driven methods either attempt to find a classical smooth solution to the QVI, or consider finite-difference schemes for (2.6); the challenge being the non-local operator \mathcal{M} .

2.1. Dynamic Programming Equation

For the remainder of the article I adopt the discrete-time paradigm, where decisions are made at K pre-specified instances $t_0 = 0 < t_1 < \dots < t_k < t_{k+1} < \dots < t_K = T$, where typically we have $t_k = k\Delta t$ for a given discretization step Δt . Henceforth, with a slight abuse of notation I index everything by k and work with $\mathcal{T} = (t_k)_{k=0}^K$. In particular this implies that we restrict $\tau_n \in \mathcal{T}$ and rule out multiple instantaneous actions, so that A consists of at most K impulses.

The dynamic programming Bellman equation for impulse control on $[t, t + \Delta t]$ is:

$$V(t, x) = \max \left(\mathbb{E}_{t,x} \left[e^{-r\Delta t} V(t + \Delta t, X_{t+\Delta t}) + \int_t^{t+\Delta t} e^{-r(s-t)} \pi(X_s) ds \right], \sup_{z \in \Xi} \mathbb{E}_{t,x+z} \left[e^{-r\Delta t} V(t + \Delta t, X_{t+\Delta t}) + \int_t^{t+\Delta t} e^{-r(s-t)} \pi(X_s) ds \right] + \kappa(x, z) \right). \quad (2.7)$$

Discretizing in time and writing $V(k, x) \equiv V(t_k, x)$, etc. we substitute

$$\mathbb{E}_{t,x} \left[\int_t^{t+\Delta t} e^{-r(s-t)} \pi(X_s) ds \right] \simeq \pi(x) \Delta t$$

and end up with

$$V(k, x) = \pi(x) \Delta t + \max(Q(k, x), M(k, x)),$$

where $Q(k, x) = \mathbb{E}_{t,x} \left[e^{-r\Delta t} V(k + 1, X_{k+1}) \right]$ is the so-called Q-value and

$$M(k, x) = \sup_{z \in \Xi} \{Q(k, x + z) + \kappa(x, z)\}.$$

The latter *intervention operator* captures the value of making the best possible impulse. In line with above, we view optimal impulse control as a two-stage sequential decision making. At each time period k , the controller must decide whether to continue (0) or act ($\neq 0$). In the latter case, she must further select the best action z^* . This matches the appearance of \max in $V(t, x)$. One should continue if the Q-value dominates the intervention value, and one should impulse otherwise. Within the Markovian structure of (2.2) the impulse strategy can be encoded as mapping each input x according to the respective feedback *action map* $\mathcal{Z}_k(x) \in \{0\} \cup \Xi$:

$$\mathcal{Z}_k(x) = \arg \max_z \{Q(k, x + z) + \kappa(x, z)\} \cdot 1_{\{M(k,x) > Q(k,x)\}}. \quad (2.8)$$

The action map \mathcal{Z}_k gives rise to the action region

$$\mathcal{S}_k := \{x : \mathcal{Z}_k(x) \neq 0\} \subseteq \mathcal{X},$$

where the optimal choice is to act.

Regression Monte Carlo proceeds by recursively constructing surrogates $\widehat{Q}(k, \cdot)$ that are used to induce the respective $\widehat{\mathcal{Z}}_k$ according to (2.8). The inductive logical loop is achieved by employing $\widehat{\mathcal{Z}}_k$ to define the forward $J_{k,K}(x; \widehat{\mathcal{Z}})$. To do so, given any set of (admissible) action maps $\mathcal{Z}_{k:K}(\cdot)$ we define the corresponding discrete-time controlled state process $X^{x,\mathcal{Z}}$ according to the Euler scheme: $X_k = x$ and

$$X_{k+1}^{x,\mathcal{Z}} = X_k^{x,\mathcal{Z}} + \mu(X_k^{x,\mathcal{Z}})\Delta t + \sigma(X_k^{x,\mathcal{Z}})\Delta W_k + \mathcal{Z}_k \quad \text{where} \quad \mathcal{Z}_k \equiv \mathcal{Z}_k(X^{x,\mathcal{Z}}). \quad (2.9)$$

Note that the action is only applied at the end of the period. The respective total revenue along the path $X^{x,\mathcal{Z}}$ is then (setting $\kappa(x, 0) \equiv 0$)

$$J_{k,K}(x; \mathcal{Z}_{k:K}) = \sum_{\ell=k}^{K-1} e^{-r(t_\ell - t_k)} \left\{ \pi(X_\ell^{x,\mathcal{Z}}) \cdot (t_{\ell+1} - t_\ell) + \kappa(X_\ell^{x,\mathcal{Z}}, \mathcal{Z}_\ell) \right\} + e^{-r(T-t_k)} \phi(X_K^{x,\mathcal{Z}}). \quad (2.10)$$

On a given path, we can also record the pathwise realized impulse times τ_n and respective impulses z_n :

$$\tau_n = \min\{k > \tau_{n-1} : \widehat{\mathcal{Z}}_k(X_k^{x,\widehat{\mathcal{Z}}}) > 0\} \quad (2.11)$$

$$z_n = \widehat{\mathcal{Z}}_k(X_{\tau_n}^{x,\widehat{\mathcal{Z}}}). \quad (2.12)$$

Denoting by \mathcal{Z}^* the optimal action map, we have that the true Q-value satisfies

$$Q(k, x) = \mathbb{E}_{k,x} [J_{k,K}(x; \mathcal{Z}_{k:K}^*)]. \quad (2.13)$$

In RMC, $Q(k, \cdot)$'s are replaced with $\widehat{Q}(k, \cdot)$; the latter induce $\widehat{\mathcal{Z}}_k$. Finally, $\widehat{\mathcal{Z}}_{k:K}$ is used via (2.13) to characterize and fit $\widehat{Q}(k-1, \cdot)$. The resulting loop is initialized with $\widehat{V}(K, x) = \phi(x)$ and proceeds as follows:

For $k = K-1, \dots, 1$ repeat:

(1) Learn the Q-value $\widehat{Q}(k, \cdot) \simeq \mathbb{E} \left[e^{-r\Delta t} \widehat{V}(k+1, X_{k+1}) \mid X_k = \cdot \right]$;

(2) Evaluate the intervention function $\widehat{M}(k, \cdot) = \sup_{z \in \Xi} \{ \widehat{Q}(k, \cdot + z) + \kappa(\cdot, z) \}$

(3) Set

$$\widehat{\mathcal{Z}}_k(x) := \begin{cases} 0 & \text{if } \widehat{Q}(k, x) > \widehat{M}(k, x) \\ \arg \max_z \{ \widehat{Q}(k, x+z) + \kappa(x, z) \} & \text{otherwise.} \end{cases} \quad (2.14)$$

(4) Record $\widehat{V}(k, x) := \max(\widehat{Q}(k, x), \widehat{M}(k, x)) + \pi(x)\Delta t$.

Note that in principle the entire \widehat{V} is superfluous. Indeed, we have generically that for any w

$$Q(k, x) = \mathbb{E} \left[J_{k,k+w}(x; \mathcal{Z}^*) + e^{-r\Delta t w} V(k+w, X_{k+w}^{k,\mathcal{Z}^*}) \right].$$

The look-ahead horizon $w \in \{1, \dots, K-k\}$ allows to combine pathwise rewards based on $\widehat{\mathcal{Z}}$ and the approximate value function \widehat{V} w -steps into the future [20]. We focus on the cases $w=1$, $w=K-k$ and w fixed that correspond to learning

$$w=1: \quad \widetilde{Q}^{(1)}(k, x) = \mathbb{E}_{k,x} \left[\pi(x)\Delta t + e^{-r\Delta t} \widehat{V}(k+1, X_{k+1}^{k,\emptyset}) \right]$$

$$w \text{ fixed:} \quad \widetilde{Q}^{(w)}(k, x) = \mathbb{E}_{k,x} \left[J_{k,k+w}(x; \widehat{\mathcal{Z}}_{k:k+w}) + e^{-r w \Delta t} \widehat{V}(k+w, X_{k+w}^{k,\widehat{\mathcal{Z}}}) \right]$$

$$w=K-k: \quad \widetilde{Q}^{(LS)}(k, x) = \mathbb{E}_{k,x} \left[J_{k,K}(x; \widehat{\mathcal{Z}}_{k:K}) + e^{-r(K-k)\Delta t} \phi(X_K^{k,\widehat{\mathcal{Z}}}) \right]$$

The choice $w = 1$ is analogous to the Tsitsiklis-van Roy [31] scheme for optimal stopping: simulate one-step-ahead paths and regress $\pi(X_k)\Delta t + \widehat{V}(k+1, X_{k+1})$ against X_k . The choice $w = K - k$ is analogous to the Longstaff–Schwartz [28] scheme, where we regress the full future rewards to go on the interval $\{k, k+1, \dots, K\}$ against X_k . These choices are not numerically identical, because $\widehat{Q}(k+1, x) \neq \mathbb{E}[J_{k+1, \ell}(x; \widehat{\mathcal{Z}}_{k:\ell}) | X_{k+1} = x]$ for different ℓ 's due to the approximation error. In all examples below, we utilize the Longstaff–Schwartz version with $w = K - k$, so that \widehat{V} is never computed until the very end.

2.2. Algorithm

The proposed RMC approach reduces optimal impulse control to a double sequence of probabilistic function approximation tasks. The primary task entails fitting a functional approximator \widehat{Q} based on empirical simulations and then utilizing a statistical model to capture the observed input-output relationship. To do so, we define a regression model and the training set used as input to the regression model. The three ingredients are the inputs $x^{1:N} \in \mathcal{X}$, the outputs $y^{1:N}$ and the approximation class \mathcal{H}_k . The inputs are the sampled states at step k . The outputs are viewed as a random realization $Y(x)$ of the pathwise reward starting at (k, x) such that $\mathbb{E}_{k,x}[Y(x)] = Q(k, x)$. Specifically, they are the realizations of $J_{k,K}(x; \widehat{\mathcal{Z}}_{k:K})$ along a set of independent paths $x^{(k),n}$ that start with $x_k^{(k),n} = x^n$. Statistically these $y^{1:N}$ are linked to the inputs by the observation model

$$Y(x) = \mathbb{E}[Y(x)] + \epsilon(x), \quad \mathbb{E}[\epsilon(x)] = 0, \quad \text{Var}(\epsilon(x)) = \sigma_\epsilon^2(x). \quad (2.15)$$

Given a training collection $\mathcal{D} = x^{1:N}$ henceforth called the simulation design, we collect the simulation outputs $y^{1:N} = Y(x^{1:N})$ and then obtain the approximate continuation value $\widehat{Q}(k, \cdot)$ (viewed as a statistical object, rather than say a vector of numbers) as the empirical L^2 minimizer in the given function space \mathcal{H} . Namely we minimize the penalized mean squared error from the observations,

$$\widehat{Q}(k, \cdot) = \arg \min_{f \in \mathcal{H}} \sum_{n=1}^N (f(x^n) - y^n)^2 + \lambda \|f\|_{\mathcal{H}}. \quad (2.16)$$

The summation in (2.16) is a measure of closeness of f to data, while the right-most term penalizes the fluctuations of f to avoid over-fitting.

Indexing everything by the time steps $k = 1, \dots, K$ and allowing for time dependence we summarize the following notation:

- N_k : number of training inputs at step k ;
- \mathcal{D}_k : simulation design, i.e. the collection of training inputs $x^{1:N_k}$, $|\mathcal{D}_k| = N_k$
- \mathcal{H}_k : functional approximation space where $\widehat{Q}(k, \cdot)$ is searched within;
- $y_k^{1:N_k}$ pathwise samples of reward-to-go used as the responses in the regression model.

Equipped with above, Algorithm 1 presents the overall scheme that abstracts from the regression module for fitting $\widehat{Q}(k, \cdot)$ and subsequently $\widehat{M}(k, \cdot)$. The algorithm matches the **mIOSP** template from [29] as implemented in the eponymous **R** package.

The output of Algorithm 1 is the approximate action maps $\widehat{\mathcal{Z}}_k(\cdot)$. Once computed, they induce the expected reward $\mathbb{E}[J_{0,K}(x; \widehat{\mathcal{Z}}_{0:K}) | X_0 = x]$, which can be evaluated over an out-of-sample set of test scenarios. Thus, we compute the sample average reward across a fresh set of

Algorithm 1 Regression Monte Carlo for Impulse Control based on **mIOSP** template.

Require: $K = T/\Delta t$ (time steps), (N_k) (simulation budget per step), w (path lookahead)

- 1: **for** $k = K - 1, \dots, 0$ **do**
 - 2: Generate training design $\mathcal{D}_k := (x_k^{(k),1:N_k})$ of size N_k
 - 3: Set $y_{k+1}^{1:N_k} \leftarrow 0$ // pathwise rewards
 - 4: **for** $\ell = k + 1, \dots, k + w \wedge K$ **do**
 - 5: Sample $x_{\ell-1}^{(k),n} \mapsto x_\ell^{(k),n}$ // pathwise controlled trajectories
 - 6: Set $y_{k+1}^n \leftarrow y_{k+1}^n + \pi(x_{\ell-1}^{(k),n})\Delta t$
 - 7: Evaluate $m_\ell^{(k),n} = \widehat{M}(\ell, x_\ell^{(k),n})$ and $q_\ell^{(k),n} = \widehat{Q}(\ell, x_\ell^{(k),n})$
 - 8: Set $x_\ell^{(k),n} \leftarrow x_\ell^{(k),n} + \widehat{\mathcal{Z}}_\ell(x_\ell^{(k),n})$ for n where $m_\ell^{(k),n} > q_\ell^{(k),n}$ // impulse
 - 9: Set $y_{k+1}^n \leftarrow y_{k+1}^n + \kappa(x_\ell^{(k),n}, \widehat{\mathcal{Z}}_\ell(x_\ell^{(k),n}))$ for n where $m_\ell^{(k),n} > q_\ell^{(k),n}$
 - 10: **end for**
 - 11: Set $y_{k+1}^n \leftarrow y_{k+1}^n + e^{-rw\Delta t} \max(\widehat{Q}(k+w, x_{k+w}^{(k),n}), \widehat{M}(k+w, x_{k+w}^{(k),n}))$.
 - 12: Fit $\widehat{Q}(k, \cdot)$ by regressing $\{y_{k+1}^{1:N_k}\}$ on $\{x_k^{(k),N_k}\}$
 - 13: **end for**
 - 14: Return fitted objects $\{\widehat{Q}(k, \cdot)\}_{k=0}^{K-1}$
-

$(x_k^{1:N'}, \widehat{\mathcal{Z}}, k = 1, \dots, K, x_0^{n'} = x),$

$$\check{V}(0, x) = \frac{1}{N'} \sum_{n'=1}^{N'} \left\{ \sum_{k=0}^{K-1} e^{-rt_k} \pi(x_k^{n'}, \widehat{\mathcal{Z}})(t_{k+1} - t_k) - \sum_{m: \tau_m^{n'} < T} e^{-r\tau_m^{n'}} \kappa(x_{\tau_m^{n'}}^{n'}, \widehat{\mathcal{Z}}, z_m^{n'}) \right\} \quad (2.17)$$

where (τ_m, z_m) are the pathwise impulse times and impulse amounts, see (2.11)–(2.12). Note that $\check{V}(0, x)$ is an unbiased estimator of $\mathbb{E}[J_{0,K}(x; \widehat{\mathcal{Z}}_{0:K}) | X_0 = x]$ and the latter is a lower bound on the true optimal expected reward, so that

$$\mathbb{E}[\check{V}(0, x)] < V(0, x).$$

2.3. Relation to Stationary Impulse Control

The cited analytical works consider the infinite horizon case of solving for

$$v(x) = \mathbb{E}_x \left[\int_0^\infty e^{-rs} \pi(X_s) ds + \sum_n e^{-r\tau_n} \kappa(X_{\tau_n}, z_n) \right].$$

Assuming time-stationary dynamics for X^\emptyset , the optimal strategy is also time-stationary, meaning that there is a feedback action map $\mathcal{Z}^*(x)$ that is independent of t and fully characterizes the impulses.

In contrast, the solution constructed above is explicitly time-dependent, as it is specified by \widehat{Q}_k that are intrinsically distinct for different k . Nevertheless, when far from the horizon K , the time-dependence should be intuitively weak, and we expect to recover the time-stationary $\mathcal{Z}^*(x)$. Indeed, informally if we parameterize in terms of time-to-maturity $\check{V}_{K-k}(\cdot) := V(k, \cdot; K)$ we observe that by induction, $\check{V}(k, \cdot; K) = \check{V}(k+1, \cdot; K+1)$ since they both correspond to running the backward RMC algorithm for $K+1 - (k+1) = K-k$ rounds. Thus, $\check{V}_\ell(\cdot)$ is well defined and the regime $\ell \rightarrow \infty$ corresponds to being far from the terminal condition so that we expect $\check{V}_\ell \rightarrow v(\cdot)$ as $\ell \rightarrow \infty$.

The above perspective suggests that for k small, we can validate our $\widehat{Q}(k, \cdot)$ by comparing with $v(\cdot)$. Conversely, we may approximate $v(\cdot)$ by $\widehat{Q}(k, \cdot; K)$ for k small and K large. To speed up that convergence, we recall *model predictive control* where during forward scenario generation

one uses \mathcal{Z}_k (rather than $\mathcal{Z}_\ell, \ell = k + 1, \dots, k + w$ for all time-steps. In other words, we compute expected reward based on a time-stationary control that is derived from the latest (in the sense of backward induction) surrogate $\widehat{Q}(k, \cdot)$. The resulting $J_{k,K}(x; \widehat{\mathcal{Z}}_k)$ captures the reward over $K - k$ steps and its expectation would be a good approximation of $v(x)$ for K large. Using model predictive control in Algorithm 1 is analogous to a policy iteration search for infinite-horizon problems; it reinterprets K as the receding horizon depth.

3. Implementation

In general, there is a huge range of potential statistical models for empirically fitting a $\widehat{Q}(k, \cdot)$. Thus, any statistical learning framework could be applied; see the `mLOSP` package that allows the use of more than a dozen different regression modules, linking to the vast library of `R` regression packages, from random forests to support vector machines. However, the fact that \widehat{Q} is necessary to evaluate \widehat{M} imposes requirements on what would be *good* surrogates. For example, piecewise models (like a random forest, multivariate adaptive regression splines (MARS), or a hierarchical linear model) would tend to be inappropriate, as they would lead to discontinuities in defining $\widehat{\mathcal{Z}}_k(x)$ and hence unstable schemes due to error backpropagation. Similarly, polynomial bases might be problematic since their gradient tends to be highly oscillatory and therefore lead to unstable behavior in $\widehat{\mathcal{Z}}_k(x)$. Overall, we seek *smooth* regression models with an interpretable gradient.

Our two main proposals are *smoothing splines* (SS) and Gaussian processes (GP). Splines intrinsically target C^2 fits, and tend to be highly robust to noisy data. Their main limitation is poor scalability, but otherwise they are a great default choice in 1 or 2 dimensions. Gaussian Processes yield smooth functional interpolators that work well with non-uniform training designs. Moreover, despite being non-parametric, GPs yield analytic gradients.

Both SS and GPs consider (2.16) for a certain smoothing parameter $\lambda \geq 0$ and a Reproducing Kernel Hilbert Space (RKHS) \mathcal{H} . The representer theorem implies that the minimizer of (2.16) therefore has an expansion in terms of the RKHS eigen-functions

$$\widehat{Q}(k, \cdot) = \sum_{n=1}^N \alpha_n \mathcal{C}(\cdot, x^n). \quad (3.1)$$

Note that this is a non-parametric fit since it involves the sum over the data-driven $\mathcal{C}(\cdot, x^n), n = 1, \dots, N$.

Smoothing splines. Thin-plate splines take the RKHS \mathcal{H}_{TPS} as the set of all twice continuously-differentiable functions with $\|f\|_{\mathcal{H}_{TPS}}^2 = \int_{\mathbb{R}} \{f''(x)\}^2 dx$. As $\lambda \rightarrow \infty$, the optimization in (2.16) penalizes any convexity and ultimately reduces to the linear fit $\widehat{Q}(x) = \beta_0 + \beta_1 x$. A common parametrization for the smoothing parameter λ is through the effective degrees of freedom statistic df_λ ; one may also select λ adaptively via cross-validation or MLE [25, Chapter 5]. The respective eigenfunctions are $\mathcal{C}_{TPS}(x, x') = |x - x'|^2 \log |x - x'|$, and optimization of (2.16) gives a smooth C^2 surrogate has the explicit form

$$\widehat{Q}_{TPS}(x) = \beta_0 + \beta_1 x + \sum_{n=1}^N \alpha_n |x - x^n|^2 \log |x - x^n|. \quad (3.2)$$

See [27] for implementation of RMC via splines.

Gaussian Processes. GPs start with a positive definite kernel $c(x, x')$ which defines the function space \mathcal{H}_c and take $\lambda = 1/2$. The corresponding norm $\|\cdot\|_{\mathcal{H}}$ has a spectral decomposition in terms of differential operators [32, Chapter 6.2]. An intuitive interpretation is that GPs find

$\widehat{Q}(k, \cdot)$ through applying Gaussian conditioning equations to the training data $(x^{1:N}, y^{1:N})$. To do so, a GPR model specifies the covariance function $c(x, x')$ and a mean function $m(x)$, assumed for simplicity to be constant $m(x) \equiv \beta_0$. The GP estimate is then

$$\widehat{Q}_{GP}(x) = \beta_0 + \mathcal{C}^T (\mathbf{C} + \sigma_\epsilon^2 \mathbf{I})^{-1} (\mathbf{y} - \beta_0 \mathbf{1}) \quad (3.3)$$

where \mathbf{I} is the $N \times N$ identity matrix, $\mathbf{1}$ is the N vector of 1's,

$$\text{where } \mathbf{y} = [y^1, \dots, y^N]^T, \quad \mathcal{C}^T = [c(x_*, x^1; \vartheta), \dots, c(x_*, x^N; \vartheta)], \quad (3.4)$$

and \mathbf{C} is $N \times N$ covariance matrix described through the kernel function $\mathbf{C}_{i,j} = c(x^i, x^j; \vartheta)$. Henceforth we think of \widehat{f} as a (smooth) function, even though it is only defined pointwise via (3.3). The role of σ_ϵ^2 is to smooth out observation noise in (2.15), interpreted as being i.i.d. Gaussian with the respective variance.

A GPR is implementing by fitting the hyper-parameters ϑ governing the covariance kernel, the mean function and the observation noise. The user first specifies a parametric family and then optimizes, typically through the nonlinear MLE procedure. The GP kernel $c(x, x')$ controls the smoothness (in the sense of differentiability) of \widehat{P} and hence the roughness of its gradient. A popular choice for $c(\cdot, \cdot)$ is the (anisotropic) squared exponential (SE) family, parametrized by the lengthscale ℓ_{len} and the process variance σ_p^2 :

$$c_{SE}(x, x') := \sigma_p^2 \exp\left(-\frac{(x - x')^2}{2\ell_{\text{len}}^2}\right). \quad (3.5)$$

The SE kernel (3.5) yields infinitely differentiable fits $\widehat{Q}(k, \cdot)$ and has hyperparameters $\vartheta := (\ell_{\text{len}}, \sigma_p^2, \sigma_\epsilon^2)$. Other popular kernels include those from the Matérn family.

Remark. Artificial Neural Networks with smooth activation functions could be another appropriate framework, facilitating training via back-propagation.

3.1. Approximating the Intervention Function

The computation of $\widehat{Z}_k(x)$ is embedded deep in Algorithm 1 and drives the outputs $y_{k+1}^{1:N}$ used to fit \widehat{Q} . In this section I discuss how that piece of the solver should be implemented. The base implementation is to directly solve (2.14) by calling an optimization sub-routine. The objective function is given *implicitly* in terms of the object $\widehat{Q}(k, \cdot)$ so ostensibly a general-purpose, gradient-free optimizer may be needed. Given that $\widehat{M}(k, \cdot)$ has to be evaluated repeatedly on each forward path emanating from each training input $x^{(k),n}$ this is the major computational bottleneck. To overcome it, several efficiencies could be exploited.

First, one may speed up the computation by using a gradient-based optimizer. This requires to specify not just $\widehat{Q}(k, \cdot)$ but also $\partial_x \widehat{Q}(k, \cdot)$ in an explicit functional way. The latter is available for several types of surrogates, including splines and GPs. For the latter, we recall that given a fitted GP model $\widehat{Q}(k, \cdot)$, its gradient forms another GP with the respective mean at input x_* being

$$g_*(x_*) := \frac{\partial \widehat{Q}}{\partial x}(x_*) = \frac{\partial c}{\partial x}(x_*, \mathbf{x})(\mathbf{C} + \sigma_\epsilon^2 \mathbf{I})^{-1} (\mathbf{y} - \beta_0 \mathbf{1}), \quad (3.6)$$

Thus, the gradient of the surrogate is $g_*(x_*)$ in (3.6) which can be interpreted as formally differentiating the expression in (3.3) with respect to x . For example, for the SE kernel (3.5) we have:

$$\frac{\partial c_{SE}}{\partial x}(x, x') = \frac{x' - x}{\ell_{\text{len}}^2} c_{SE}(x, x').$$

Second, one may exploit specific features of the problem setting. As a foremost example, I now discuss the common case where $\kappa(x, z)$ is linear in z , namely $\kappa(x, z) = c_0 z + c_1$ for some constants

c_0, c_1 . In this situation, the optimization defining $\widehat{M}(k, x)$ simplifies considerably. Indeed, the first order conditions reduce to searching for the “global” impulse target S_k^* :

$$S_k^* := \arg \sup_z \{\widehat{Q}(k, x + z) + \kappa(x, z)\} = \arg \sup_y \{\widehat{Q}(k, y) + c_0(y - x) + c_1\} \\ \iff \partial_x \widehat{Q}(k, S_k^*) = -c_0. \quad (3.7)$$

In particular, the target level S_k^* is independent of the current state x , and moreover can be determined by a single root search on the gradient of the value function. This drastically simplifies and stabilizes the numerics, since we just need to determine S_k^* once, and can then immediately compute $\widehat{M}(k, x) = \widehat{Q}(k, S_k^*) + c_0 S_k^* - c_0 x + c_1$ for any x . Consequently, the action region is $\widehat{\mathcal{S}}_k = \{x : \widehat{Q}(k, S_k^*) - \widehat{Q}(k, x) > c_0(x - S_k^*) - c_1\}$.

Third, when computing $\widehat{M}(k, x^{1:N_k})$ for each training input $x^{1:N_k}$, one can record and save the resulting optimal impulse amount $z_k^{1:N_k}$. Then in subsequent calls, instead of again solving for $\widehat{M}(k, x')$ at some new x' by re-running the optimizer, one may instead train a separate independent functional representation $\widehat{Z}(k, \cdot)$ based on the dataset $(x_k^{1:N_k}, z_k^{1:N_k})$. Thus, we could build an auxiliary surrogate $\widehat{Z}(k, \cdot)$ (e.g. through another GP surrogate) and then use $\widehat{Z}(k, x')$ instead of $\arg \max_z \{\widehat{Q}(k, x' + z) + \kappa(x', z)\}$. This substitutes the prediction $\widehat{Z}(k, \cdot)$, which is typically much faster to compute, instead of calling the optimizer.

3.2. Training Designs

To train the regression surrogate, the user must supply the simulation design(s) \mathcal{D}_k . See [29] for a detailed description of various options for training optimal stopping emulators within `mLOSP`. With impulse control, the major difference is that (X_k) is no longer autonomous. Thus, it no longer makes sense to construct $\mathcal{D}_k \sim p(X_k)$ as a sample from the uncontrolled dynamics. Instead, I propose to directly specify \mathcal{D}_k , building on the idea that the quality of \widehat{Q} reflects the geometry of \mathcal{D}_k . One learns best in regions where the training samples lie. Since the optimally controlled (X_k^*) typically has a stationary distribution (modulo time-dependence imposed by the finite horizon), one may select a training *region* based on a prior guess of the latter. For example, one may choose a hyper-rectangle $\overline{\mathcal{D}}$ and then set \mathcal{D}_k as a finite, space-filling sample from $\overline{\mathcal{D}}$, yielding \mathcal{D}_k that looks like a sample from a uniform density on $\overline{\mathcal{D}}$. Some of the ways to achieve this are This can be achieved either through a deterministic lattice, or i.i.d. Uniform (stratified) sampling, or a low-discrepancy (Quasi Monte Carlo) sequence. All these choices will

- Direct specification of \mathcal{D}_k , e.g. as a fixed lattice `seq(a, b, by=Δx)`;
- Probabilistic sampling of \mathcal{D}_k , either using i.i.d. Uniforms, or a variance reduced variant of the former (e.g. stratified sampling) or a Latin Hypercube sampling method (package `lhs` in R);
- Generation of \mathcal{D}_k from a (scrambled) low-discrepancy sequence (LDS), such as Sobol (package `randtoolbox` in R). Note that in this case \mathcal{D}_k is deterministic. This is specified by the `qmc.method` field that supports LHS (default) and various LDS.

Beyond targeting a uniform density of training samples on a given region, one may also take non-uniform \mathcal{D}_k that preferentially place more training inputs in some parts of \mathcal{X} . For example, we may put more x^k 's in the region where we expect the impulse target to be, in order to improve the quality of \widehat{Q} there, and hence the quality of \widehat{M} . The underlying intuition is that learning is achieved through exploration (sampling a diverse collection of x^k 's) and exploitation (sampling x^k 's that are likely to be encountered on forward controlled paths).

A further training option that I highlight is *replication*. A replicated design is akin to a Monte Carlo forest, in the sense that some training inputs appear multiple times. In a most common

batched design, we have N_{unique} distinct sites, the so-called macro-design, and each unique x^n is then repeated N_{rep} times, so that

$$\mathcal{D} = \underbrace{\{x^1, x^1, \dots, x^1\}}_{N_{rep} \text{ times}}, \underbrace{\{x^2, \dots, x^2\}}_{N_{rep} \text{ times}}, \dots, x^{N_{unique}}, \quad (3.8)$$

where the superscripts now index *unique* inputs and the total training budget for $\widehat{Q}(k, \cdot)$ is $|\mathcal{D}| = N_{unique} \times N_{rep}$. The corresponding simulator outputs are denoted as $y^{1,1}, y^{1,2}, \dots, y^{n,i}, \dots, y^{N_{unique}, N_{rep}}$.

A replicated design allows to pre-average the corresponding y -values, $\bar{y}^n := \frac{1}{N_{rep}} \sum_{i=1}^{N_{rep}} y^{n,i}$, and then calling the regression module on the reduced dataset $(x^{1:N_{unique}}, \bar{y}^{1:N_{unique}})$. Replication with pre-averaging offers a simple way of reducing the variance of observations. This is often desirable because the pathwise rewards tend to be highly volatile especially over longer periods of time, and many functional approximators struggle under low signal-to-noise settings. With high degree of replication, one can view \bar{y}^n as almost deterministic, so that regression effectively reduces to interpolation.

3.3. Illustration

To illustrate the overall workflow of solving an optimal impulse problem, I present a few brief code snippets. These utilize the `m1OSP` constructs and can be directly reproduced by any reader who installs the package. Consider impulsing a 1-D Geometric Brownian Motion (GBM) process with uncontrolled dynamics

$$dX_t^\emptyset = \mu X_t^\emptyset dt + \sigma X_t^\emptyset dW_t, \quad X_0^\emptyset = x_0,$$

with scalar parameters μ, σ, x_0 . Thus, (X_t^\emptyset) can be simulated exactly by sampling from the respective log-normal distribution. The running payoff is of concave power-type $\pi(x) = x^\gamma/\gamma$, $0 < \gamma < 1$, and the intervention costs are linear $\kappa(x, z) = c_0 \cdot z + c_1$. This setup is motivated by Federico et al. [23] who considered irreversible investment with fixed adjustment costs. The state process (X_t) represents an economic indicator, such as the production capacity of a firm which drives the revenue rate $\pi(X_t)$.

As discussed above, linear impulse costs yield an optimal strategy of (s, S) type: intervene as soon as (X_t) goes below s and bring it back up to $S > s$. Thanks to the linearity of GBM and $\kappa(x, \cdot)$, and the power-form of $\pi(x)$ the infinite-horizon problem is known to have an explicit solution

$$\tilde{v}(x) = Bx^m + Cx^\gamma/\gamma, \quad s = \left(\frac{c_0(m-1)}{C(m-\gamma)} \right)^{\frac{1}{\gamma-1}}, \quad (3.9)$$

$$m = \left(\frac{1}{2} - \frac{\mu}{\sigma^2} \right) - \sqrt{\left(\frac{1}{2} - \frac{\mu}{\sigma^2} \right)^2 + \frac{2r}{\sigma^2}},$$

$$C = \frac{1}{r - \mu\gamma + 0.5\gamma(1-\gamma)\sigma^2}, \quad B = \frac{C(1-\gamma)}{m(m-1)} s^{\gamma-m}. \quad (3.10)$$

Thus, with infinite horizon the controlled (X_t) will be a time-stationary renewal process and undergo a cyclical behavior with renewal times $\tau_{n+1} = \tau_n + \inf\{t > 0 : X_t^{\tau_n, S} \leq s\}$ and $z_{n+1} = S - s$.

To implement the above instance, one starts by defining the `model`, which is a list of (a) parameters that determine the dynamics of (X_t^\emptyset) in (2.1); (b) the running payoff function $\pi(x)$; (c) the impulse function $\kappa(x, z)$ and (d) the tuning parameters determining the regression surrogate specification. In the example I take $r = 0.08$, $\mu = -0.07$, $\sigma = 0.25$, $c_0 = -1$, $c_1 = -10$ and square-root running reward $\gamma = 0.5$ which yields the time-stationary solution thresholds $s = 8.749$, $S = 56.99$. I use a finite horizon of $T = 10$ with $\Delta t = 0.1$, i.e. $K = 100$ time-steps.

```

modelFRT <- list(dim=1,
sim.func=sim.gbm,
r=0.08, # discount factor
div=0.15, # drift is mu=-0.07
sigma=0.25, # volatility
x0=50, # initial state
impulse.fixed.cost = 10, # fixed
impulse.cost.linear = 1, # linear
impulse.func = lin.impulse,
imp.type = "exchrates",
gamma = 0.5,
running.func = function(x)(2*sqrt(x)), # cont profit rate pi
T=10, # horizon
dt=0.1, # time step; 100 steps total
pilot.nsim=0,
batch.nrep = 10, # replicates for each unique input
nk = 20, # number of spline knots
N = 541,
)
input.dom = c(seq(1,10,by=0.05), seq(10.25,100,by=0.25))

```

Next, we must choose a solver scheme, namely specifying the surrogate type and the training sets. In `mLOSP`, the solver implementing Algorithm 1 is named `osp.impulse.control` and comes with a `method` field that controls the surrogate, and `input.domain` that controls the simulation designs. The code below utilizes a spline surrogate that requires specifying the number of knots, already set via `nk` field in `modelFRT` above. For the `input.domain` I utilize a non-uniform lattice that is dense for $x \in [1, 10]$ (region of the action set) and less so for $x \in [10, 100]$. I also employ replication, with each input replicated `batch.nrep=10` times.

For the terminal condition I take the expected value of future running rewards given the current state and no more impulses, i.e. $\widehat{Q}(K, x) = \phi(x) = \mathbb{E}_x[\int_0^\infty e^{-rt}\pi(X_t^\emptyset) dt] = Cx^\gamma/\gamma$ where C is from (3.10). This is interpreted as T being the horizon for actions, thereafter X evolves endogenously without any further controls.

```

spl.solver <- osp.impulse.control(modelFRT,
input.domain = input.dom,method="spline")

```

Note that no output is printed: the produced object `spl.solver` contains an array of 99 (one for each time step, except at maturity) fitted smoothing spline surrogates for $\widehat{Q}(k, \cdot)$. Technically, `spl.solver` is a list that has a few other diagnostics beyond the collection of the `smooth.spline` objects.

Figure 3.1 visualizes the fitted \widehat{Q} from three time-steps. To do so, we simply predict the Q-value object over a collection of test locations. In the Figure, this is done for $k = 1, 30, 60$. The middle panel shows the gradient $\partial_x \widehat{Q}(k, \cdot)$, obtained by finite-differencing, at the same time steps k .

To better understand the resulting strategy $\widehat{Z}_{0:K}$ we build an independent database of forward controlled paths. This is done via the `forward.impulse.policy` command that evaluates (2.17) and also records all the associated actions (impulse amounts and times). The right panel of Figure 3.1 shows two different controlled forward paths based on the computed $\widehat{Z}_{0:K}$. On the blue path there are 3 impulse times τ_n ; on the purple one only two. One can clearly see the (s, S) policy where $(X_t^{\widehat{Z}})$ is impulsed whenever it gets too low and is then brought up to about $S_k^* \simeq 60$ which is the target level.

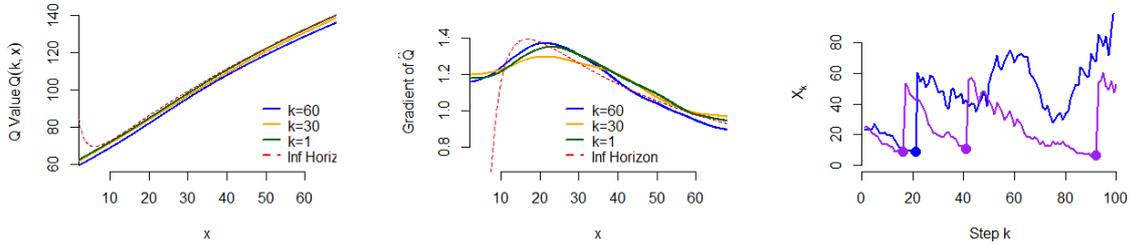


FIGURE 3.1. A 1D capacity expansion instance. Left: Q-value based on a Smoothing Spline emulator at $k \in \{1, 30, 60\}$. Middle: corresponding gradient $\partial_x \hat{Q}(k, x)$. The target level S_k^* is the threshold where $\partial_x \hat{Q}(k, S_k^*) = 1$. Right: two resulting controlled paths of $X^{\hat{Z}}$. Dots indicate the times $\tau_k^{n'}$ of impulses.

Below we also show code to plot the impulse strategy, namely the impulse boundary s_k^* and the impulse target levels S_k^* , displayed in Figure 3.2. Note that the shown s_k^* is based on the forward paths, so at some times k there is no recorded s_k^* since none of the forward paths were impused at that specific k (impulses are not so frequent). The Figure shows the time-stationary s, S values that confirm the good approximation by the present solver. One can also observe the time-dependence which manifests itself through the agent being impatient as problem horizon is approached. As a result, impulses are applied sooner (lower timing value, i.e. lower opportunity cost of acting) and we see a “boundary layer” as $k \rightarrow K$. The slight fluctuations observed in s_k^*, S_k^* are due to Monte Carlo-driven approximation errors and can be decreased with larger training sets.

```
S.target <- rep(0,100)

for (j in 1:99)
  S.target[j] <- lin.impulse(seq(1,10,by=1),modelFRT,
spl.solver$fit[[j]],ext=TRUE)$imp.target[1]

# forward paths
fi <- forward.impulse.policy(array(10,dim=c(10000,1)), 100, spl.solver$fit,modelFRT)
# these are the s-values
plot(fi$bnd, xlab='Time Step k', ylab='State x', pch=19,cex=1.2, col="red", ylim=c(0,65))
points(S.target[1:99], cex=1.2, col="blue",pch=19)
abline(h=c(56.99,8.749),lty=2,col=c("blue","red")) # solution of the inf-horizon
```

4. Case Studies

In this section I present two more case studies that showcase other problem settings and further features available in m1OSP.

4.1. Faustmann Problem of Forest Rotation

For the next example of an impulse control problem amenable to Algorithm 1, I take up the problem of forest management as nicely summarized and analyzed by Alvarez and co-authors [4, 6]. Let X_t represent the value of forest stand, i.e. the economic value of existing timber. Forest growth is modeled as a stochastic process: timber increase is uncertain and fluctuates due to weather, precipitation and other environmental effects. Droughts or insect infestations might reduce forest stand, justifying the use of stochastic differential equations for modeling (X_t).

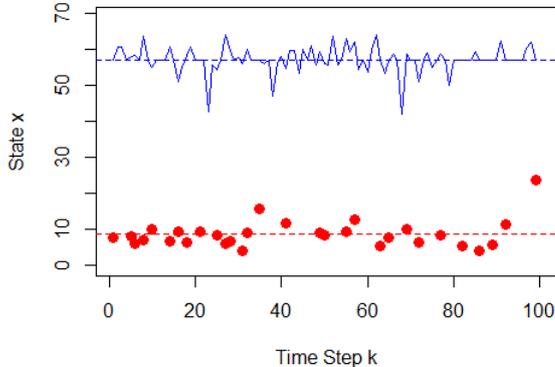


FIGURE 3.2. Threshold boundary s_k^* (dots towards the bottom) and threshold target levels S_k^* (line towards the top) for the irreversible investment case study.

The controller aims to maximize total profit from cutting down and selling timber on a given time horizon T . This is achieved through carrying out a sequence of so-called forest rotations. At each rotation, the forest stand is cut down and sold. The number of rotations is stochastic and up to the forest manager. The horizon T represents the lease term for the timberland, measured in years. The objective functional is then

$$\mathbb{E} \left[\sum_{n:\tau_n < T} e^{-r\tau_n} \kappa(X_{\tau_n-}, z_n) \right], \quad (4.1)$$

where the payoff function κ captures the value of selling z_n timber, subject to the cutting costs, and r is the intertemporal discount rate. Above we assume zero salvage value at T , $\phi(X_T) \equiv 0$. As might be expected, the timber will be cut at some time-dependent threshold S^*_k .

In the classic formulation, the problem is stated on infinite horizon, the dynamics of X_t are linear and time-homogeneous, and the post-rotation level $X_{\tau_k} \equiv \underline{x}$ is pre-specified. This offers an explicit solution, see [4], who showed that the action region is $\mathfrak{S}_k = \{x > S^*\}$ where S^* is the maximizer of a certain nonlinear equation. The finite-horizon version is analyzed in [12] and I reproduce their example where (X_t^θ) is a standard arithmetic Brownian Motion, $r = 0.1$, and there is a fixed cost for each rotation, $\kappa(x, z) = (z - 1)_+$. Thus, the forest is always cut down to nominal level zero $\underline{x} = 0$. This means that $z_n = X_{\tau_n-}$. The reference impulse boundary is $S^* = 1.84$.

I take a horizon of $T = 5$ years and time steps of $\Delta t = 0.1$, yielding $K = 50$ periods. For the regression, I utilize a Gaussian Process emulator with the squared-exponential kernel (3.5) and hyperparameters fitted via Maximum Likelihood Estimation, as done in the `DiceKriging` package. I then use the exact surrogate gradient based on (3.6) to efficiently find S_k^* in (2.8). For the training simulation design, the particular structure of this formulation implies that it is important to obtain an accurate estimate of $\hat{Q}(k, 0)$. To this end, I consider training in a Monte Carlo forest like fashion, employing a high degree of replication so that each unique input will have multiple forward paths emanating from it. Namely, I take 100 unique inputs on a lattice between $[-0.25, 2.5]$, each replicated 100 times, for a total of 10,000 forward training paths.

The left panel of Figure 4.1 effectively reproduces Figure 1 in Belak et al. [12]; in contrast to that paper where the impulse boundary is obtained from a solution of an integral equation and requires specific assumptions on the impulse cost function and dynamics of (X_t) , my method is completely generic and can be trivially re-solved if any of the ingredients were to change. In

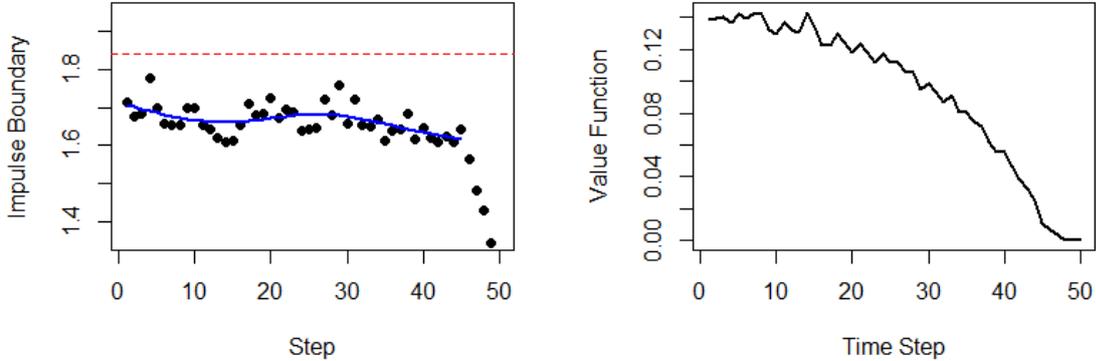


FIGURE 4.1. Left: estimated impulse boundary \hat{S}_k^* as a function of time step k , infinite horizon threshold S^* is shown as a dashed line. To smooth out minor numerical artifacts we also display a smoothed estimate of S^* as a blue curve. Right: value function $\hat{V}(k, 0)$ at zero.

Figure 4.1 we can clearly observe the effect of the terminal condition, where the manager will cut down even a bit of forest ahead of the deadline T that would give him no profit whatsoever. One also notes that the estimated impulse boundary is below that of the infinite-horizon problem. This arises due to (i) restricting actions to take place in \mathcal{T}_k , here with separation $\Delta t = 0.2$, this is known to induce the manager to act sooner; (ii) the finite horizon that remains non-negligible with $T = 5$, manifested by the impulse boundary slowly moving up as k decreases. The right panel of Figure 4.1 shows the estimated value $\hat{V}(k, 0)$ which also displays time-dependence and indicates convergence (i.e. approaching time-stationary) with about 5 years until maturity.

4.2. Two Dimensional Capacity Expansion

In the 2-dimensional version of capacity expansion, the state processes are the production price (P_t) and the current capacity C_t . The price is exogenous and stochastic, while the capacity is fully endogenous and deterministic. We refer to [13] and [24] who provided explicit solutions (up to solving an integral equation) in some special cases.

The price follows Geometric Brownian motion

$$dP_t = \mu P_t dt + \sigma P_t dW_t$$

and the capacity undergoes deterministic exponential decay/aging with rate δ :

$$C_t = e^{-\delta t} C_0 + \sum_k z_k e^{-\delta(t-\tau_k)}$$

with profit function $\pi(P_t, C_t) = P_t C_t^\alpha$ for $\alpha < 1$, representing decreasing efficiency of adding more capacity. We identify above with a two dimensional state $X_t \in \mathbb{R}_+^2$ where the impulses $z \in \mathbb{R}_+$ only affect the second coordinate: an impulse of size z leads to $P_{\tau_n} = P_{\tau_n-}$ and $C_{\tau_n} = C_{\tau_n-} + z$. Obvious generalizations to handle the two coordinates are straightforward to handle in code and are effectively abstracted away by the package as far as the user is concerned.

Following [24] I consider concave investment costs $\kappa((p, c), z) = z^\beta$ with $\beta < 1$ and $\beta > \alpha$. The concavity of κ encourages making large investments. The quantity $Y_t := P_t C_t^{\alpha-\beta}$ can be interpreted as the firm's return on assets (ROA) and affords dimension reduction in the case of log-linear dynamics as above. Indeed [24] shows that with these choices and infinite horizon,

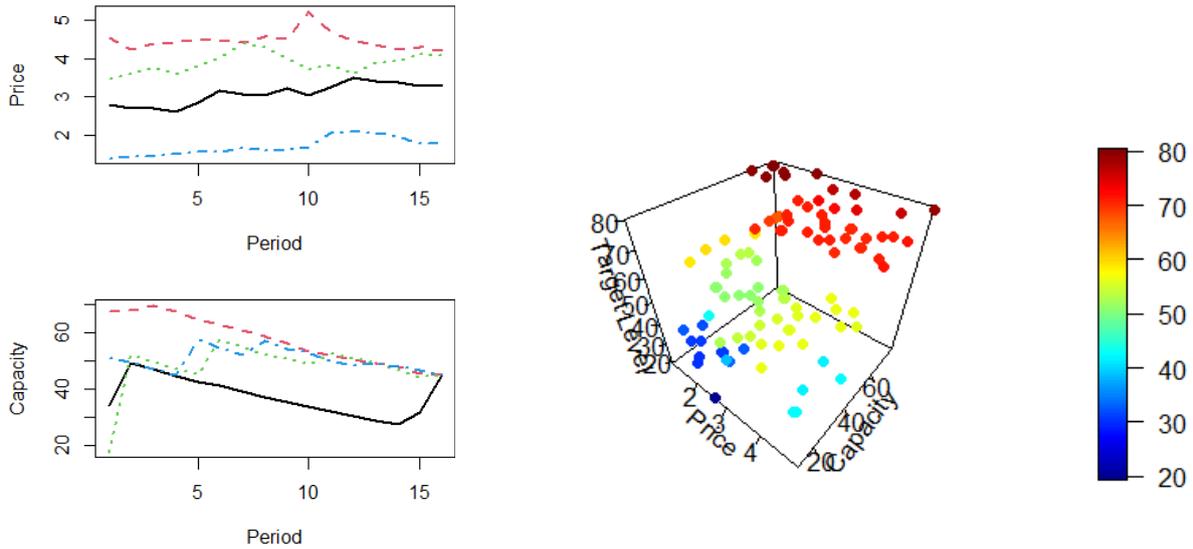


FIGURE 4.2. Two dimensional finite horizon impulse control problem inspired by [24]. Left: 4 controlled trajectories of $X_t = (P_t, C_t)$. Capacity C_t decays exponentially without impulses. Right: impulse target $(p, c) \mapsto c + \widehat{Z}_k(p, c)$ for a representative time step k .

$V(p, c) = c^\beta v(c^{\alpha-\beta} p)$ where $v(y) = B_0 y + B_1 y^\gamma$ for some explicit constants B_0, B_1, γ and the optimal impulses are of (s, S) -type in Y_t , rather than P_t, C_t separately.

Guthrie [24] considers the parameter values $r = 0.04, \mu = 0, \sigma = 0.08, \delta = 0.1, \beta = 0.95, \alpha = 0.5$ which gives ROA threshold $y_0 = 0.224$. Moreover, the respective optimal impulse is to increase capacity by 178; this happens on average once every 11 years. In this setup, the problem is strongly non-stationary: (P_t) is autonomous and can grow without bound (since $\mu > 0$) while C_t is impused upwards. Consequently, one must select time-dependent simulation designs \mathcal{D}_k lest the forward paths end up extrapolating, rather than interpolating \widehat{Q} .

For the terminal condition we set $\phi(x) = \mathbb{E}[\int_0^\infty e^{-rs} \pi(X_s) ds \mid X_0 = (p, c)] = pc^\alpha \frac{1}{r-\mu}$. There is no simple way to summarize the resulting solution; Figure 4.2 displays a few optimally controlled paths, as well as the impulse target map (namely the arg max of $\widehat{M}(k, \cdot)$) on the training set \mathcal{D}_k , here chosen to be a Sobol low-discrepancy sequence. We note that with the nonlinear impulse costs, the impulse target depends nontrivially on both coordinates, increasing both in price p and in current capacity c .

Acknowledgements

I thank the anonymous referees for useful feedback on earlier versions of this manuscript, as well as Zhuoli Jin (UCSB) for research assistance.

References

- [1] René Aid, Salvatore Federico, Huy en Pham, and Bertrand Villeneuve. Explicit investment rules with time-to-build and uncertainty. *J. Econ. Dyn. Control*, 51:240–256, 2015.
- [2] Luis H. R. Alvarez. A class of solvable impulse control problems. *Appl. Math. Optim.*, 49(3):265–295, 2004.

- [3] Luis H. R. Alvarez. Optimal capital accumulation under price uncertainty and costly reversibility. *J. Econ. Dyn. Control*, 35(10):1769–1788, 2011.
- [4] Luis H. R. Alvarez and Erkki Koskela. Optimal harvesting under resource stock and price uncertainty. *J. Econ. Dyn. Control*, 31(7):2461–2485, 2007.
- [5] Luis H. R. Alvarez and Erkki Koskela. Taxation and rotation age under stochastic forest stand value. *J. Environ. Econ. Manage.*, 54(1):113–127, 2007.
- [6] Luis H. R. Alvarez and Jukka Lempa. On the optimal stochastic impulse control of linear diffusions. *SIAM J. Control Optim.*, 47(2):703–732, 2008.
- [7] Pablo Azcue, Nora Muler, and Zbigniew Palmowski. Optimal dividend payments for a two-dimensional insurance risk process. *Eur. Actuar. J.*, 9(1):241–272, 2019.
- [8] Parsiad Azimzadeh, Erhan Bayraktar, and George Labahn. Convergence of implicit schemes for Hamilton–Jacobi–Bellman quasi-Variational inequalities. *SIAM J. Control Optim.*, 56(6):3994–4016, 2018.
- [9] Matteo Basei. Optimal price management in retail energy markets: an impulse control problem with asymptotic estimates. *Math. Methods Oper. Res.*, 89(3):355–383, 2019.
- [10] Erhan Bayraktar, Andreas E. Kyprianou, and Kazutoshi Yamazaki. Optimal dividends in the dual model under transaction costs. *Insur. Math. Econ.*, 54:133–143, 2014.
- [11] Erhan Bayraktar and Michael Ludkovski. Inventory management with partially observed nonstationary demand. *Ann. Oper. Res.*, 176(1):7–39, 2010.
- [12] Christoph Belak, Sören Christensen, and Frank Thomas Seifried. A general verification result for stochastic impulse control problems. *SIAM J. Control Optim.*, 55(2):627–649, 2017.
- [13] Alain Bensoussan and Benoît Chevalier-Roignant. Sequential capacity expansion options. *Oper. Res.*, 67(1):33–57, 2019.
- [14] Alain Bensoussan, R. H. Liu, and Suresh P. Sethi. Optimality of an (s, S) policy with compound Poisson and diffusion demands: A quasi-variational inequalities approach. *SIAM J. Control Optim.*, 44(5):1650–1676, 2005.
- [15] Abel Cadenillas and Fernando Zapatero. Classical and impulse stochastic control of the exchange rate using interest rates and reserves. *Math. Finance*, 10(2):141–156, 2000.
- [16] Yann-Shin Aaron Chen and Xin Guo. Impulse control of multidimensional jump diffusions in finite time horizon. *SIAM J. Control Optim.*, 51(3):2638–2663, 2013.
- [17] Sören Christensen. On the solution of general impulse control problems using superharmonic functions. *Stochastic Processes Appl.*, 124(1):709–729, 2014.
- [18] Irmina Czarna and Zbigniew Palmowski. De Finetti’s dividend problem and impulse control for a two-dimensional insurance risk process. *Stoch. Models*, 27(2):220–250, 2011.
- [19] Masahiko Egami. A direct solution method for stochastic impulse control problems of one-dimensional diffusions. *SIAM J. Control Optim.*, 47(3):1191–1218, 2008.
- [20] Daniel Egloff, Michael Kohler, and Nebojsa Todorovic. A dynamic look-ahead Monte Carlo algorithm for pricing Bermudan options. *Ann. Appl. Probab.*, 17(4):1138–1171, 2007.
- [21] Brahim El Asri and Sehail Mazid. Stochastic impulse control Problem with state and time dependent cost functions. *Math. Control Relat. Fields*, 10(4):855, 2020.
- [22] Brahim El Asri and Sehail Mazid. Zero-sum stochastic differential game in finite horizon involving impulse controls. *Appl. Math. Optim.*, 81(3):1055–1087, 2020.
- [23] Salvatore Federico, Mauro Rosetolato, and Elisa Tacconi. Irreversible investment with fixed adjustment costs: a stochastic impulse control approach. *Math. Financ. Econ.*, 13(4):579–616, 2019.
- [24] Graeme Guthrie. Uncertainty and the trade-off between scale and flexibility in investment. *J. Econ. Dyn. Control*, 36(11):1718–1728, 2012.

- [25] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer Series in Statistics. Springer, 2009.
- [26] Jianqiang Hu, Cheng Zhang, and Chenbo Zhu. (s, S) inventory systems with correlated demands. *INFORMS J. Comput.*, 28(4):603–611, 2016.
- [27] Michael Kohler. A regression-based smoothing spline Monte Carlo algorithm for pricing American options in discrete time. *AStA, Adv. Stat. Anal.*, 92(2):153–178, 2008.
- [28] Francis A. Longstaff and Eduardo S. Schwartz. Valuing American options by simulations: a simple least squares approach. *Rev. Financ. Stud.*, 14:113–148, 2001.
- [29] Mike Ludkovski. mlOSP: Towards a Unified Implementation of Regression Monte Carlo Algorithms. <https://arxiv.org/abs/2012.00729>, 2020.
- [30] Bernt Karsten Øksendal and Agnes Sulem. *Applied stochastic control of jump diffusions*, volume 498. Springer, 2007.
- [31] John Tsitsiklis and Benjamin Van Roy. Regression Methods for Pricing Complex American-Style Options. *IEEE Trans. Neural Netw.*, 12(4):694–703, 2001.
- [32] Christopher K. I. Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2006.