# On parametric optimal execution and machine learning surrogates

Tao Chen, Mike Ludkovski & Moritz Voß

# On parametric optimal execution and machine learning surrogates

TAO CHEN†, MIKE LUDKOVSKI ‡* and MORITZ VOSS §

†Department of Mathematics, University of Michigan, 530 Church Street, Ann Arbor, MI 48109-1043, USA
‡Department of Statistics & Applied Probability, University of California Santa Barbara, Santa Barbara, CA 93106-3110, USA
§Department of Mathematics, University of California Los Angeles, Los Angeles, CA 90095, USA

We investigate optimal order execution problems in discrete time with instantaneous price impact and stochastic resilience. First, in the setting of linear transient price impact we derive a closed-form recursion for the optimal strategy, extending the deterministic results from Obizhaeva and Wang [Optimal trading strategy and supply/demand dynamics. *J. Financial Mark.*, 2013, **16**(1), 1–32]. Second, we develop a numerical algorithm based on dynamic programming and deep learning for the case of nonlinear transient price impact as proposed by Bouchaud et al. [Fluctuations and response in financial markets: The subtle nature of 'random' price changes. *Quant. Finance*, 2004, **4**(2), 176–190]. Specifically, we utilize an actor-critic framework that constructs two neural-network (NN) surrogates for the value function and the feedback control. The flexible scalability of NN functional approximators enables parametric learning, i.e. incorporating several model or market parameters as part of the input space. Precise calibration of price impact, resilience, etc., is known to be extremely challenging and hence it is critical to understand the sensitivity of the execution policy to these parameters. Our NN learner organically scales across multiple input dimensions and is shown to accurately approximate optimal strategies across a wide range of parameter configurations. We provide a fully reproducible Jupyter Notebook with our NN implementation, which is of independent pedagogical interest, demonstrating the ease of use of NN surrogates in (parametric) stochastic control problems.

*Keywords*: Optimal execution; Parametric control; Neural network surrogates; Stochastic resilience

## 1. Introduction

In the past decade, an extensive literature has analyzed optimal execution of trades within a micro-structural framework that accounts for the interaction between trades, prices and limit order book liquidity. For example, one notable strand follows the approach of Obizhaeva and Wang (2013) which is able to generate closed-form formulas for optimal trading strategy with linear transient price impact. However, any practical use of these elegant mathematical derivations must immediately confront the strong dependence of the solution on the given model parameters. Concepts such as order book resilience or book depth are mathematical abstractions and are not directly available in the real world. Similarly, parameters such as inventory penalty, are model-specific and have to be entered by the user. Consequently, model calibration becomes highly nontrivial and in turn requires understanding the interaction between model parameters and the resulting strategy. In parallel, model risk, i.e. mis-specification of the dynamics, is also a major concern. Model risk can be partially mitigated by considering more realistic nonlinear models, but this comes at the cost of losing closed-form solutions.

Motivated by these issues, in this article we approach optimal execution using the lens of *parametric* stochastic control. To this end, we investigate numerical algorithms that determine the optimal execution strategy jointly in terms of the state variables (inventory and limit order book spread/markup), as well as model parameters (instantaneous price impact, resilience factor, inventory penalty, etc.). We consider augmenting 1–4 model parameters to the training space, yielding multi-dimensional control problems.

Our contribution is twofold. In terms of the numerical methods, we propose a direct approach to parametric control that focuses on generating functional approximators to the value function. A concrete choice of such *statistical surrogates* that we present below are Neural Networks (NNs). Our

*Corresponding author. Email: ludkovski@pstat.ucsb.edu

approach employs NNs to approximate the Bellman equation and can be contrasted with other ways of using NNs, such as Deep Galerkin Methods (Al-Aradi *et al.* 2018, 2019, Germain *et al.* 2021) for PDEs; or parameterization of the feedback control (Huré *et al.* 2018, Bachouch *et al.* 2018) in tandem with stochastic gradient descent. One advantage of our method is its conceptual simplicity, as such it is an excellent *pedagogical* testbed for machine learning methods. Indeed, given that optimal execution is now a core problem that is familiar to anyone working in mathematical finance, we believe that this example offers a great entryway to students or researchers who wish to understand and 'play' with modern numerical tools for stochastic control. To this end, we provide a detailed Python Jupyter notebook that allows a fully reproducible checking of our results. Our notebook is made as simple as possible, to distill where the statistics tools come in, and aiming to remove much of the 'ML mystique' that can be sometimes present. Our approach also emphasizes the questions of how to train the statistical surrogate, and how to approximate the optimal control, both important implementation aspects whose discussion is often skipped.

In terms of the financial application, we contribute to the optimal execution literature in several ways that would be of independent interest to experts in that domain. Specifically, our optimal execution modeling framework builds on the discrete-time, linear transient price impact model with exponential decay from Obizhaeva and Wang (2013) and additionally allows for (i) nonlinear, power-type price impact à la Bouchaud *et al.* (2004, 2009), Gatheral (2010), Alfonsi *et al.* (2010), Dang (2017) and Curato *et al.* (2017); (ii) a stochastic transient price impact driven by its own noise, akin to the continuous-time model in Becherer *et al.* (2018) and (iii) a risk-aversion-type running quadratic penalty on the inventory as arising, e.g. in Schied *et al.* (2010). In other words, our model nests various existing proposals in the literature, offering a unified discrete-time framework that we investigate in detail. Other related works on optimal order execution with (exponentially) decaying (linear) transient price impact in discrete and continuous time include, e.g. Alfonsi *et al.* (2008), Alfonsi and Schied (2010), Predoiu *et al.* (2011), Alfonsi *et al.* (2012), Gatheral *et al.* (2012), Lorenz and Schied (2013), Alfonsi and Schied (2013), Alfonsi and Acevedo (2014), Bank and Fruth (2014), Alfonsi and Blanc (2016), Fruth *et al.* (2014, 2019), Graewe and Horst (2017), Lehalle and Neuman (2019), Horst and Xia (2019), Chen *et al.* (2019), Ackermann *et al.* (2021a, 2021b), Forde *et al.* (2021) and Neuman and Voß (2022); we also refer to the recent monograph by Webster (2023) for an excellent overview and discussion of price impact modeling. Moreover, in the linear case, we also establish a new explicit formula, see proposition 3.1, for the optimal execution strategy with stochastic transient price impact and inventory penalty, which extends the explicit deterministic solution from Obizhaeva and Wang (2013) and allows us to also accurately benchmark our machine learning approach. Therefore, our numerical experiments provide new reliable insights on the interaction between different model parameters and the optimal strategy. Obtaining these insights, in other words building a better intuition on how the model behaves in different regimes, was the original motivation for our work,

and is valuable for practitioners who must develop gut feelings on how the model reacts as the real-world (i.e. calibrated parameters) changes. In particular, our numerical analysis complements the studies on deterministic optimal execution strategies with non-linear transient price impact carried out in Dang (2017) and Curato *et al.* (2017).

In the broader context of machine learning methods for stochastic optimal control, our work is related to other applications of neural networks to financial problems, see Huré *et al.* (2018), Bachouch *et al.* (2018) and Ismail and Pham (2019). Perhaps the closest is Leal *et al.* (2021) who also study execution problems, but in a model with only temporary and permanent price impact à la Bertsimas and Lo (1998), Almgren and Chriss (2001), Cartea and Jaimungal (2016); see also Papanicolaou *et al.* (2023) for a similar study. For other approaches in optimal execution in the presence of temporary price impact more in the flavor of reinforcement learning, see, e.g. the recent survey articles by Hambly *et al.* (2021) and Jaimungal (2022) and the references therein.

This article is organized as follows. Section 2 formulates our generalized optimal execution setting. Section 3 presents an explicit reference solution for unconstrained trading strategies. Section 4 describes our methodology for parametric stochastic control via statistical surrogates. Section 5 presents the numerical experiments and resulting insights. Following a brief conclusion in Section 6, Section 7 contains the proofs.

## 2. Problem formulation

Let $(\Omega, \mathscr{F}, \mathbb{F} = (\mathscr{F}_n)_{n=0,\ldots,N}, \mathbb{P})$ be a discrete-time filtered probability space with trivial $\sigma$-field $\mathscr{F}_0$ and a terminal time horizon $N \in \mathbb{N}$. We consider a financial market with one risky asset whose $\mathbb{F}$-adapted real-valued unaffected fundamental price process is denoted by $P = (P_n)_{n=0,\ldots,N}$. We set $P_0 := p_0 \in \mathbb{R}_+$.

Suppose a large trader dynamically trades in the risky security and incurs price impact in an adverse manner. Specifically, for $n \in \{1, \ldots, N\}$, by choosing her number of shares in the risky asset at time $n-1$, she trades $u_n \in \mathscr{F}_{n-1}$ shares in the $n$th trading period and confronts the $n$th fundamental random shock $\Delta P_n := P_n - P_{n-1} \in \mathscr{F}_n$. Her action permanently affects the future evolution of the mid-price process $M = (M_n)_{n=0,\ldots,N}$ which becomes

$$M_n := P_n + \gamma \sum_{j=1}^{n} u_j, \quad n = 1, \ldots, N, \qquad (1)$$

after the $n$th order $u_n$ is executed (we set $M_0 := P_0$). The parameter $\gamma \geq 0$ represents linear permanent price impact.

In addition, the trader's market orders are filled at a deviation $D_n$ from the mid-price $M_n$ in (1). The *post*-execution dynamics of these deviations $D = (D_n)_{n=1,\ldots,N}$ from the mid-price after trading $u_n$ shares at time $n \in \{1, \ldots, N\}$ are modeled as

$$D_0 := d_0,$$
$$D_n := (1 - \kappa)D_{n-1} + \eta|u_n|^{\alpha}\,\mathrm{sgn}(u_n) + \epsilon_n,$$
$$n = 1, \ldots, N, \qquad (2)$$

with $d_0 \in \mathbb{R}$ denoting the given initial deviation. Thus, in the absence of the large trader's actions, the deviation tends to revert exponentially to zero at rate $\kappa \in (0, 1]$, with the latter parameter known as the book *resilience*. The resilience captures the transience of the instantaneous price impact, with $\kappa$ representing the fraction by which the deviation from the mid-price $M$ incurred by past trades diminishes over a trading period. Empirically, the deviation process $D$ can be calibrated by considering the limit order book (LOB) spread and depth.

Due to finite market depth, which is measured by $1/\eta > 0$, the trader's turnover of $u_n$ shares pushes the deviation in the trade's direction by a constant factor $\eta$ times the instantaneous price impact, which is assumed to be $|u_n|^\alpha \operatorname{sgn}(u_n)$, $\alpha > 0$. Following, e.g. Bouchaud *et al.* (2004, 2009), the latter power-type term generalizes the common linear situation $\alpha = 1$ from Obizhaeva and Wang (2013) where the instantaneous price impact is $\eta u_n$. However, empirically the instantaneous price impact is observed to be concave (at least for relatively small $u_n$), so that $\alpha < 1$ seems more realistic; see Lillo *et al.* (2003), Bouchaud *et al.* (2009) and Bacry *et al.* (2015).

The $\mathbb{R}$-valued, $\mathbb{F}$-adapted sequence of zero-mean random variables $(\epsilon_n)_{n=1,\ldots,N}$ represents additional small perturbations in the deviation stemming, e.g. from market and limit orders, which are placed at time $n$ by other small market participants, making the transient price impact captured by the deviation process $D$ stochastic; cf., e.g. Becherer *et al.* (2018).

For the rest of the paper, we assume that $(\Delta P_n)_{n=1,\ldots,N}$ are independent and square integrable random variables with mean zero. The perturbations $(\epsilon_n)_{n=1,\ldots,N}$ in (2) are i.i.d. normally distributed random variables with mean zero and variance $\sigma^2 > 0$, and independent of $(\Delta P_n)_{n=1,\ldots,N}$ as well. We also refer to the case $\sigma = 0$ where $\epsilon_n \equiv 0$ for every $n \in \{1, \ldots, N\}$. Finally, we let the filtration $\mathbb{F}$ be given by $\mathscr{F}_n = \sigma(\{\Delta P_1, \epsilon_1, \ldots, \Delta P_n, \epsilon_n\})$ for all $n \in \{1, \ldots, N\}$.

### 2.1. Optimal trade execution

From now on, we suppose that the large trader wants to carry out a *buying program* to buy $X_0 > 0$ shares by executing $N$ market *buy orders* $u_n \geq 0$, $n = 1, \ldots, N$. She starts with zero inventory and we use $X_n$ to denote the remaining number of shares she needs to buy after step $n$ to reach her target. That is, we set

$$X_n := X_0 - \sum_{j=1}^n u_j, \quad n = 1, \ldots, N, \tag{3}$$

where $X_n$ represents her remaining order to be filled *after* she executed her $n$th trade $u_n$.

As common in the literature, to describe the evolution of the large trader's cash balance, we assume that the $n$th transaction $u_n$ affects the mid-price $M$ and deviation $D$ gradually. More precisely, half of the $n$th order is filled at the *pre*-transaction's mid-price $M_{n-1}$, as well as the refreshed *pre*-transaction's deviation $(1 - \kappa)D_{n-1}$, whereas the other half is executed at the less favorable *post*-transaction quantities $M_n - \Delta P_n$ (before the $n$th fundamental random shock hits the stock price) and $D_n - \epsilon_n$ (before the $n$th random perturbation

of the deviation). Hence, assuming zero interest rates, the self-financing condition dictates that changes in the trader's cash balance $(C_n)_{n=1,\ldots,N}$ with initial value $c_0 \in \mathbb{R}$ are only due to her buying activity of the risky asset which is executed at the previously described *average* execution prices:

$$C_0 := c_0,$$
$$C_n := C_{n-1} - \left[ \frac{1}{2}(M_{n-1} + (1 - \kappa)D_{n-1}) \right.$$
$$\left. + \frac{1}{2}(M_n - \Delta P_n + D_n - \epsilon_n) \right] u_n,$$
$$= C_{n-1} - \left( P_{n-1} - \frac{\gamma}{2}(X_n + X_{n-1}) + \gamma X_0 \right) u_n$$
$$- \left( (1 - \kappa)D_{n-1} + \frac{\eta}{2} u_n^\alpha \right) u_n, \quad n = 1, \ldots, N. \tag{4}$$

LEMMA 2.1 *The terminal cash position $C_N$ at time $N$ of a buying schedule $(u_n)_{n=1,\ldots,N}$ with $u_n \geq 0$ for all $n \in \{1, \ldots, N\}$ and terminal state constraint $X_N = 0$ is given by*

$$C_N = c_0 - \left( p_0 + \frac{\gamma}{2} X_0 \right) X_0 - \sum_{n=1}^{N-1} X_n \Delta P_n$$
$$- \sum_{n=1}^N \left( (1 - \kappa)D_{n-1} + \frac{\eta}{2} u_n^\alpha \right) u_n. \tag{5}$$

*Proof* Using $u_n = X_{n-1} - X_n$, we obtain from (4) together with (2) and (1)

$$C_N = C_0 - \sum_{n=1}^N P_{n-1} u_n + \frac{\gamma}{2} \sum_{n=1}^N (X_{n-1}^2 - X_n^2) - \gamma X_0 \sum_{n=1}^N u_n$$
$$- \sum_{n=1}^N \left( (1 - \kappa)D_{n-1} + \frac{\eta}{2} u_n^\alpha \right) u_n$$
$$= c_0 - \sum_{n=1}^{N-1} X_n \Delta P_n - p_0 X_0 - \gamma X_0 (X_0 - X_N)$$
$$+ \frac{\gamma}{2} (X_0^2 - X_N^2) - \sum_{n=1}^N \left( (1 - \kappa)D_{n-1} + \frac{\eta}{2} u_n^\alpha \right) u_n. \tag{6}$$

Under the terminal state constraint $X_N = 0$, the representation in (6) simplifies to (5) ∎

Next, to introduce the trader's optimization problem let us denote for all time steps $n = 1, \ldots, N$ the collection of admissible buying strategies by

$$\mathscr{A}_n := \left\{ (u_j)_{j=n,\ldots,N} : u_j \in L^{(\alpha+1)\vee 2}(\mathscr{F}_{j-1}, \mathbb{P}), u_j \geq 0 \text{ a.s.} \right.$$
$$\left. \text{for all } j = n, \ldots, N, \sum_{j=n}^N u_j = X_{n-1} \right\}. \tag{7}$$

The trader aims to maximize her expected terminal cash position given in (5) while also controlling for inventory

risk. The latter is modeled through a quadratic urgency penalty $\nu X_n^2$ for an urgency parameter $\nu \geq 0$ on her outstanding order. Combining the two terms, the objective is to minimize

$$\inf_{(u_n)_{n=1,\ldots,N} \in \mathscr{A}_1} \mathbb{E}\left[ \sum_{n=1}^{N} \left\{ \left( (1-\kappa)D_{n-1} + \frac{\eta}{2}u_n^\alpha \right) u_n \right. \right.$$
$$\left. \left. + \nu(X_{n-1} - u_n)^2 \right\} \right]. \tag{8}$$

Note that $X_{n-1} - u_n = X_n$ is the remaining order to be filled; our notation emphasizes the role of $X_{n-1}$ and $D_{n-1}$ as the state variables, and $u_n$ as the control. As it is well known in the literature, we remark that the permanent impact $\gamma$ disappears in (8) and from our further discussion since it only adds a fixed offset $-0.5\gamma X_0^2$ to the terminal cash position, irrespective of the trading strategy. Similarly, the martingale term $\sum_n X_n \Delta P_n$ in (5) disappears as well after taking expectations thanks to the independence of price increments.

We introduce for all $n \in \{1, \ldots, N\}$ the value function as

$$V_n(x,d) := \inf_{(u_j)_{j=n,\ldots,N} \in \mathscr{A}_n} \mathbb{E}\left[ \sum_{j=n}^{N} \left\{ \left( (1-\kappa)D_{j-1} + \frac{\eta}{2}u_j^\alpha \right) u_j \right. \right.$$
$$\left. \left. + \nu(X_{j-1} - u_j)^2 \right\} \,\middle|\, X_{n-1} = x, D_{n-1} = d \right]. \tag{9}$$

To characterize $V_n$, we use the corresponding dynamic programming (DP) equation. Since at the last period $N$, the admissible set $\mathscr{A}_N$ is a singleton $u_N \equiv X_{N-1}$, we have the terminal condition

$$V_N(x,d) = (1-\kappa) \cdot d \cdot x + \frac{\eta}{2}x^{\alpha+1}, \tag{10}$$

and then for $n = N-1, \ldots, 1$

$$V_n(x,d) = \inf_{u \in [0,x]} \mathbb{E}\left[ (1-\kappa) \cdot d \cdot u + \frac{\eta}{2}u^{\alpha+1} + \nu(x-u)^2 \right.$$
$$\left. + V_{n+1}(x-u, D_n) \,\middle|\, X_{n-1} = x, D_{n-1} = d \right] \tag{11}$$

with $D_n = (1-\kappa)d + \eta u^\alpha + \epsilon_n$ as postulated in (2) and the expectation being with respect to the Gaussian noise $\epsilon_n \sim \mathscr{N}(0, \sigma^2)$.

The DP equation (11) has two primary state variables $(X_{n-1}, D_{n-1})$. Below, we will also consider its dependence on the static parameters $\kappa, \eta, \alpha, \nu, \sigma$. Recall that $\kappa \in (0,1]$ is the book resilience (smaller $\kappa$ increases the transient price impact); $\eta > 0$ is the instantaneous price impact (larger $\eta$ makes trades affect $D_n$ more); $\nu \geq 0$ is the urgency parameter (larger $\nu$ encourages larger buys to mitigate inventory risk); $\alpha \approx 1$ is the exponent of the instantaneous price impact function ($\alpha \gtrless 1$ leads to convex (resp. concave) price impact) and $\sigma > 0$ is the standard deviation of the one-step-ahead deviation $D_n$. Note that while $\kappa$ and $\alpha$ are dimensionless, the value of $\nu$ should be thought of relative to the initial inventory $X_0$, and the values of $\eta, \sigma$ should be picked relative to fluctuations

in $D_n$. For example, if $X_0 = 10^5$ (buying program of a 100,000 shares), then $\nu$ should be on the order of $10^{-4}$, $\eta$ should be on the order of $10^{-3}$ (so that $D_n$ is on the order of 10–100), and $\sigma$ should be on the order of 1.

REMARK 2.1 (Unconstrained Problem) In the above formulated optimal trade execution problem, it is tempting to a priori allow for trading in both directions (buy orders $u_n > 0$ and sell orders $u_n < 0$) as in the discrete-time linear transient price impact model in Ackermann *et al.* (2021b); i.e. to compute

$$V_n^\circ(x,d) :=$$
$$\inf_{(u_j)_{j=n,\ldots,N} \in \mathscr{A}_n^\circ} \mathbb{E}\left[ \sum_{j=n}^{N} \left\{ \left( (1-\kappa)D_{j-1} + \frac{\eta}{2}|u_j|^\alpha \,\mathrm{sgn}(u_j) \right) u_j \right. \right.$$
$$\left. \left. + \nu(X_{j-1} - u_j)^2 \right\} \,\middle|\, X_{n-1} = x, D_{n-1} = d \right] \tag{12}$$

over the set of unconstrained order schedules

$$\mathscr{A}_n^\circ := \left\{ (u_j)_{j=n,\ldots,N} : u_j \in L^{(\alpha+1)\vee 2}(\mathscr{F}_{j-1}, \mathbb{P}), \; \mathbb{R} \right.$$
$$\left. -\text{valued for all } j = n, \ldots, N, \sum_{j=n}^{N} u_j = X_{n-1} \right\}. \tag{13}$$

However, as it will become apparent in section 3, the exogenous noise $(\epsilon_n)_{n=1,\ldots,N}$ in the deviation process $D$ in (2) would then trigger price manipulation in the sense of Huberman and Stanzl (2004). That is, there would exist profitable round-trip trades, i.e. nonzero strategies $(u_n)_{n=1,\ldots,N} \in \mathscr{A}_1^\circ$ that generate strictly negative expected costs with $X_0 = 0 = X_N$ by exploiting a nonzero deviation $D_n$ for some $n \in \{0, 1, \ldots, N\}$. This can be ruled out by either introducing a bid-ask spread or confining trading in one direction only; see also the discussion on price manipulation in Ackermann *et al.* (2021b) and Fruth *et al.* (2014, 2019) for the linear case $\alpha = 1$, as well as Alfonsi and Schied (2010), Gatheral (2010) and Curato *et al.* (2017) for the nonlinear case $\alpha \neq 1$.

## 3. Explicit solution for the unconstrained linear case

The unconstrained optimal trade execution problem in (12) and (13) can be solved explicitly in the case of linear transient price impact ($\alpha = 1$) following a similar computation as done by Obizhaeva and Wang (2013). They derived a solution in the deterministic case ($\sigma = 0$) without inventory penalty ($\nu = 0$) and with initial deviation $d_0 = 0$.

PROPOSITION 3.1 *Let $\alpha = 1$ and let $\sigma \geq 0$. Define*

$$a_N := \frac{\eta}{2}, \quad b_N := 1 - \kappa, \quad c_N := 0 \tag{14}$$

*and, recursively, for all $n = N - 1, \ldots, 1$, set*

$$
\begin{cases}
a_n := & \nu + a_{n+1} - \dfrac{(2\nu + 2a_{n+1} - \eta b_{n+1})^2}{2\eta + 4\nu + 4a_{n+1} - 4\eta b_{n+1} + 4\eta^2 c_{n+1}}, \\[3mm]
b_n := & (1-\kappa)b_{n+1} + \dfrac{2(1-\kappa)(2\nu + 2a_{n+1} - \eta b_{n+1})(1 - b_{n+1} + 2\eta c_{n+1})}{2\eta + 4\nu + 4a_{n+1} - 4\eta b_{n+1} + 4\eta^2 c_{n+1}}, \\[3mm]
c_n := & (1-\kappa)^2 c_{n+1} - \dfrac{(1-\kappa)^2(1 - b_{n+1} + 2\eta c_{n+1})^2}{2\eta + 4\nu + 4a_{n+1} - 4\eta b_{n+1} + 4\eta^2 c_{n+1}}.
\end{cases}
\tag{15}
$$

*Then the value function in* (12) *is quadratic in its arguments and given by*

$$
V_n^\circ(x, d) = a_n x^2 + b_n x d + c_n d^2 + \sigma^2 \sum_{j=n+1}^{N} c_j
$$

$$
(n = 1, \ldots, N).
\tag{16}
$$

*Moreover, the optimal order execution strategy $(u_n^\circ)_{n=1,\ldots,N} \in \mathscr{A}_1^\circ$ is given by*

$$
u_n^\circ = \frac{(2\nu + 2a_{n+1} - \eta b_{n+1})X_{n-1}^\circ - (1 - b_{n+1} + 2\eta c_{n+1})}{(1-\kappa)D_{n-1}^\circ}{\eta + 2\nu + 2a_{n+1} - 2\eta b_{n+1} + 2\eta^2 c_{n+1}}
$$

$$
(n = 1, \ldots, N-1),
$$

$$
u_N^\circ = X_{N-1}^\circ,
\tag{17}
$$

*where $X_{n-1}^\circ = X_0 - \sum_{j=1}^{n-1} u_j^\circ$ and*

$$
D_{n-1}^\circ = (1-\kappa)^{n-1} d_0 + \eta \sum_{j=1}^{n-1} (1-\kappa)^{(n-1)-j} u_j^\circ
$$

$$
+ \sum_{j=1}^{n-1} (1-\kappa)^{(n-1)-j} \epsilon_j
\tag{18}
$$

*for all $n = 1, \ldots, N+1$.*

REMARK 3.1 In the deterministic case $\sigma = 0$ and setting $\nu = 0$ as well as $d_0 = 0$, the solution in proposition 3.1 coincides with the deterministic solution presented in Obizhaeva and Wang (2013, proposition 1).

Observe that for every $n \in \{1, \ldots, N\}$ the optimal execution policy $u_n^\circ$ in (17) is given as a deterministic linear feedback function of the controlled random state variables $X_{n-1}^\circ$ and $D_{n-1}^\circ$. Also, the coefficients of this linear function do not depend on $\sigma$. Therefore, in the deterministic version of the problem where $\sigma = 0$, the optimal policy is given by the exact same feedback law in (17). Moreover, since the state variables $X_{n-1}^\circ$ and $D_{n-1}^\circ$ are themselves linear in $(u_j^\circ)_{j=1,\ldots,n-1}$ and the i.i.d. zero-mean Gaussian noise $(\epsilon_j)_{j=1,\ldots,n-1}$, it follows that the optimal trades $(u_n^\circ)_{n=1,\ldots,N}$ of the stochastic version of the problem with $\sigma > 0$ are in fact just a Gaussian-distributed perturbation from the corresponding deterministic solution with mean given by the latter.

COROLLARY 3.1 *On average, the optimal order executions $(u_n^\circ)_{n=1,\ldots,N}$ in* (17) *coincide with $u_n^\circ$ for the deterministic problem with $\sigma = 0$. Moreover, viewed as a random variable, $u_n^\circ$ has a Gaussian distribution with the above mean.*

REMARK 3.2 (Profitable Round-Trip Strategies) Let $N \geq 2$. By virtue of Lemma 7.1, we obtain that the coefficients $(c_n)_{n=1,\ldots,N-1}$ in (15) are all strictly negative for $\kappa < 1$. As a consequence, if $\sigma > 0$ and taking zero initial buying volume outstanding ($X_0 = 0$), we have $V_n^\circ(0, d) < V_n^\circ(0, 0) < 0$ in (16) for $n < N$. In other words, the resulting optimal strategy in (17) is a profitable round-trip strategy in the sense of Huberman and Stanzl (2004). This happens because the feedback policy readily exploits a nonzero deviation and its resilience as it arises; see also the detailed discussion in Ackermann *et al.* (2021b) and Fruth *et al.* (2014, 2019). Similarly, in line with the latter references, observe that there are no profitable round-trip strategies (nor transaction triggered price manipulation strategies in the sense of Alfonsi *et al.* 2012) in the deterministic resilience case $\sigma = 0$, as long as the initial deviation satisfies $d_0 = 0$ (because $V_0^\circ(0, 0) = 0$ and $u_n^\circ \equiv 0$ for all $n = 1, \ldots, N$).

### 3.1. Deterministic resilience case $\sigma = 0$

In the deterministic case $\sigma = 0$, the solution presented in proposition 3.1 can be rewritten in an explicit closed-form without backward recursion in (15) and forward feedback policy in (17). In view of corollary 3.1, this is useful for revealing the dependence of the average optimal order executions $u_1^\circ, \ldots, u_N^\circ$ on the model parameters $\kappa \in (0, 1]$, $\eta > 0$ and $\nu \geq 0$.

PROPOSITION 3.2 *Let $\sigma = 0$. Moreover, let $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \mathfrak{a}^x, \mathfrak{b}^x, \mathfrak{c}^x, \mathfrak{a}^d, \mathfrak{b}^d, \mathfrak{c}^d \in \mathbb{R}^N$ denote the vectors defined in* (50), (52), (54) *below. Set*

$$
\hat{b} := -\frac{\mathfrak{a}^\top(\eta\mathfrak{b} + (1-\kappa)\mathfrak{b}^d) + (1-\kappa)(\mathfrak{a}^d)^\top\mathfrak{b} + 2\nu(\mathfrak{a}^x)^\top\mathfrak{b}^x}{\eta\mathfrak{a}^\top\mathfrak{a} + 2(1-\kappa)(\mathfrak{a}^d)^\top\mathfrak{a} + 2\nu(\mathfrak{a}^x)^\top\mathfrak{a}^x},
$$

$$
\hat{c} := -\frac{\mathfrak{a}^\top(\eta\mathfrak{c} + (1-\kappa)\mathfrak{c}^d) + (1-\kappa)(\mathfrak{a}^d)^\top\mathfrak{c} + 2\nu(\mathfrak{a}^x)^\top\mathfrak{c}^x}{\eta\mathfrak{a}^\top\mathfrak{a} + 2(1-\kappa)(\mathfrak{a}^d)^\top\mathfrak{a} + 2\nu(\mathfrak{a}^x)^\top\mathfrak{a}^x}.
\tag{19}
$$

*Then the optimal order execution strategy from proposition 3.1 in* (17) *is given by*

$$
u_1^\circ = \hat{b} \cdot d_0 + \hat{c} \cdot X_0
\tag{20}
$$

*and*

$$
u_n^\circ = \mathfrak{a}_n \cdot u_1^\circ + \mathfrak{b}_n \cdot d_0 + \mathfrak{c}_n \cdot X_0 \quad (n = 2, \ldots, N).
\tag{21}
$$

*Moreover, the optimally controlled deviation process and remaining inventory are given by*

$$
X_n^\circ = \mathfrak{a}_{n+1}^x \cdot u_1^\circ + \mathfrak{b}_{n+1}^x \cdot d_0 + \mathfrak{c}_{n+1}^x \cdot X_0,
$$

$$D_n^\circ = \mathfrak{a}_{n+1}^d \cdot u_1^\circ + \mathfrak{b}_{n+1}^d \cdot d_0 + \mathfrak{c}_{n+1}^d \cdot X_0$$
$$(n = 0, \ldots, N-1). \tag{22}$$

Observe that in proposition 3.2 the deterministic optimal strategy $u_1^\circ, \ldots, u_N^\circ$ in (21) is fully characterized by the first trade $u_1^\circ$ in (20) and the size of the orders vary over time $n \in \{1, \ldots, N\}$. A further simplification is obtained when there is no urgency/inventory penalty, i.e. $\nu = 0$. Specifically, all intermediate trades $u_2^\circ, \ldots, u_{N-1}^\circ$ are flat and determined as a $\kappa$-fraction of the first order $u_1^\circ$, shifted by a proportion of $d_0$.

COROLLARY 3.2 *Set $\nu = 0$ in proposition* 3.2. *Then in* (19) *it holds that*

$$\hat{b} = -\frac{\eta b(N-2)(1+\kappa(N-1)) + \kappa(1-\kappa)}{\kappa\eta(N-1)(2+(N-2)\kappa)} \quad \text{and}$$

$$\hat{c} = \frac{1}{2+(N-2)\kappa}, \tag{23}$$

*and the optimal intermediate trades in* (21) *with initial trade $u_1^\circ$ in* (20) *are constant and simplify to*

$$u_n^\circ = \kappa \cdot u_1^\circ + \tilde{b} \cdot d_0 \quad (n = 2, \ldots, N-1) \tag{24}$$

*with $\tilde{b}$ defined in* (48) *below. The final trade is given by $u_N^\circ = X_0 - (1+(N-2)\kappa) \cdot u_1^\circ - (N-2)\tilde{b} \cdot d_0$. The optimally controlled deviation process and remaining inventory in* (22) *simplify to*

$$X_n^\circ = X_0 - (1+(n-1)\kappa) \cdot u_1^\circ - (n-1)\tilde{b} \cdot d_0,$$
$$D_n^\circ = \eta \cdot u_1^\circ + (1-\kappa) \cdot d_0$$
$$(n = 1, \ldots, N-1). \tag{25}$$

*In particular, if $d_0 = 0$ it holds that*

$$u_1^\circ = u_N^\circ = \frac{X_0}{2+(N-2)\kappa} \quad \text{and} \quad u_2^\circ = \ldots = u_{N-1}^\circ = \kappa u_1^\circ. \tag{26}$$

Corollary 3.2 shows that in the deterministic case without urgency penalty $\nu = 0$ the optimal execution strategy in (24) is constant for the intermediate trades from 2 to $N-1$ and keeps the deviation process in (25) flat until $N-1$. If, in addition, $d_0 = 0$ (i.e. the setup in Obizhaeva and Wang 2013) the optimal strategy simplifies to the symmetric U-shaped (26) and becomes independent of the instantaneous price impact parameter $\eta$ (as observed in Obizhaeva and Wang 2013). Initial and last trades are the same and all remaining intermediate trades are simply prescribed as a constant $\kappa$-fraction of the initial trade. In particular, for full resilience $\kappa = 1$ all trades are just equal to $X_0/N$.

REMARK 3.3 The simple formula in (26) has also been derived in Alfonsi *et al.* (2010, corollary 6.1).

### 3.2. Optimal execution profiles

Figure 1 illustrates the behavior of an optimal buying program from proposition 3.2 with linear transient price impact ($\alpha = 1$) for different values of the model parameters $\kappa, \eta$.
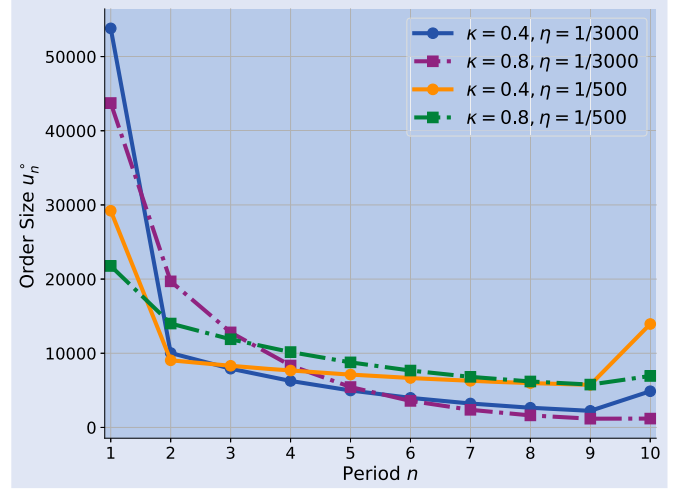


Figure 1. Benchmark unconstrained buying programs ($u_n^\circ$) based on proposition 3.2 for the case $\sigma = 0$. We consider executing $X_0 = 10^5$ shares in $N = 10$ steps for different values of the model parameters $\kappa, \eta$ and $\nu = 0.00005$, $d_0 = 0$.

We consider buying $X_0 = 100\,000$ shares in $N = 10$ trades, with initial deviation $d_0 = 0$. As discussed, when $\nu = 0$ we obtain the well-known U-shape for the buying schedule, driven only by $\kappa$. With positive $\nu > 0$ several novel qualitative effects appear: (i) the optimal strategy now depends on *both* resilience rate $\kappa$ *and* temporary price impact $\eta$; (ii) intermediate trades are in general not flat anymore; and (iii) the overall pattern may shift from a U-shaped to a monotonically decreasing sequence of trades. The presence of an inventory penalty $\nu > 0$ creates an incentive to buy faster in the beginning, which becomes more pronounced for small $\eta$. Consequently, the case of $\nu > 0$ and small $\eta$ might lead to a decreasing execution schedule, cf. the purple curve in figure 1.

Observe that relatively small changes in model parameters generate significant impact on the optimal strategy $u_n^\circ$. For example, the initial trade $u_1^\circ$ can be over 55% of total $X_0$ when $\kappa$ is small and $\eta$ is small (which causes the emphasis to be on inventory risk), and less than 25% of $X_0$ for $\kappa$ large and $\eta$ large (where strong resilience and strong temporary impact encourage to trade nearly equal amounts at each step). Also note the non-monotone behavior of the strategies, where the curves $n \mapsto u_n^\circ$ cross each other at different steps.

In the right panel of figure 2, we illustrate how the pathwise strategies $u_n^\circ$ for $\sigma > 0$ from proposition 3.1 are normally distributed and coincide on average with the deterministic solution, cf. corollary 3.1. In the plot, the blue bars correspond to the optimal deterministic solution from proposition 3.2 and corollary 3.2; and the box plots show the distribution of $(u_n^\circ)_{n=1,\ldots,10}$ from proposition 3.1. We take $\sigma = 2$ and average over $10\,000$ paths, considering both the classical case with $\nu = 0$ (left panel with U-shaped strategy) and our extension to positive inventory penalty $\nu > 0$ (right panel). In line with corollary 3.1, the empirical means match the deterministic solution. Since this feature is true for any configuration of $(\kappa, \eta, \nu)$ the above qualitative effects can be deduced for the stochastic optimal feedback controls in (17).
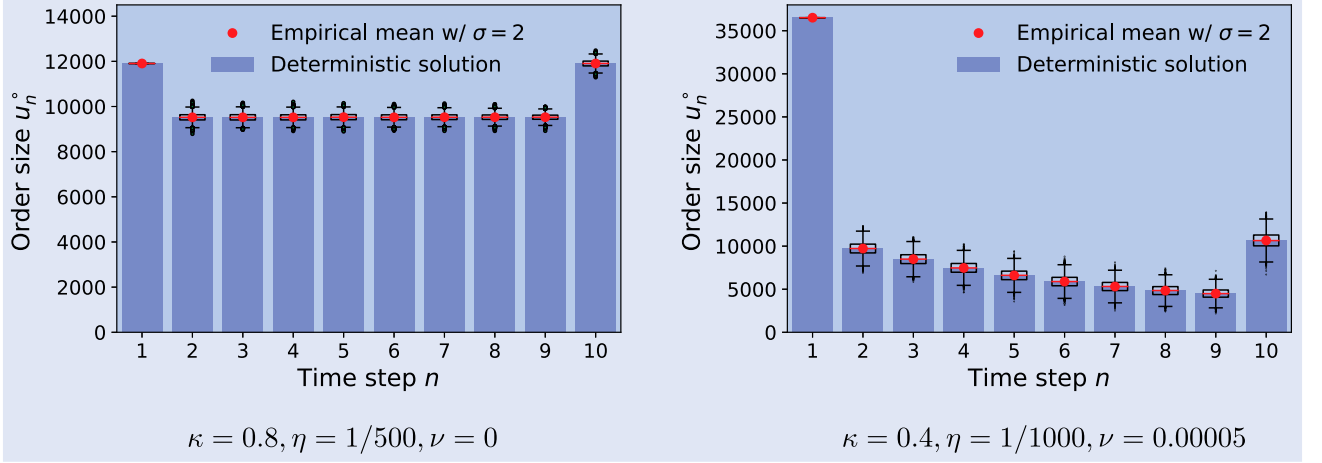
Figure 2. Distribution of $(u_n^\circ)_{n=1,\ldots,10}$ from proposition 3.1 over 10 000 paths for $\sigma = 2$ (box plots) together with the empirical mean (in red) compared to the formula from proposition 3.2 for $\sigma = 0$ (bar plot).

## 4. Numerical implementation

In the case of nonlinear transient price impact $\alpha \neq 1$, no closed-form solution is possible and numerical methods are needed to compute an optimal solution $(u_n^*)_{n=1,\ldots,N}$ for the minimization problem in (8). A numeric algorithm is also needed for $\alpha = 1$ to handle the constrained buy-only setting $u_n \geq 0$. Finally, the algorithm will be useful to study the dependence of $V_n, u_n^*$ on model parameters. Our algorithm to compute the value function and the optimal strategy relies on the Bellman equation (11) which provides a recursive characterization for $V_n$ and therefore for $u_n^*$. Given the parametric setup, we consider a generic state $y$ that fuses stochastic states $(x, d)$ and some (or none) of the aforementioned model parameters; and for the rest of this section treat $V_n$ and $u_n^*$ as a function of $y$.

To solve (11), we need to be able to evaluate its right-hand side. This entails (i) evaluating $V_{n+1}(\cdot)$; (ii) evaluating the conditional expectation; (iii) taking the *inf* over $u$. None of those steps are possible to do analytically and numerical techniques are necessary.

Below, we propose and implement a direct approach based on constructing a functional approximator, also known as a *surrogate*, $\hat{V}$, to the value function that is trained via an empirical regression. This method is also known as *Approximate Dynamic Programming (ADP)* and *Projected Value Iteration (PVI)* in the literature; see, e.g. Keshavarz and Boyd (2014) and the references therein. Specifically, we consider the use of (feed-forward) Neural Networks (NN) for the latter. Note that our method is based on the DP equation; we do not make any reference to Hamilton–Jacobi–Bellman equations that are used in continuous-time setups, and which admit their own suites of NN approaches. Neither do we consider reinforcement learning (RL) that dispenses with the divide-and-conquer paradigm underlying the Bellman equation and the value function, and aims to maximize total trading revenue on the entire horizon. Namely, RL solves for all $V_n$'s in parallel over $n = 1, 2, \ldots, N$ while we maintain the sequential backward learning of $V_{N-1}, V_{N-2}, \ldots, V_1$.

Instead, our approach is conceptually faithful to the classical DP philosophy and offers the following advantages:

- It is straightforward to understand and implement, in some sense offering the most immediate approach to employing surrogate and other machine learning techniques for DP. As such, we bypass the more subtle ideas that have been advanced, offering a pedagogic-flavored setup.
- It provides modularization, emphasizing that NN solvers are just one type of *many* potential surrogates. Thus it de-mystifies deep learning, simply treating it as a choice among many. Indeed, our implementation requires just a few lines of code to substitute a different surrogate type.

### 4.1. Building a surrogate

Denote by $\mathcal{Y} \subseteq \mathbb{R}^\ell$ the state space of the surrogate, which includes the bona fide inputs $x$, $d$, as well as all the relevant model parameters among $\kappa, \eta, \alpha, \nu$ which we wish to capture. In the examples below, we consider $\ell \in \{2, 3, 4, 5\}$, with the main illustrative example being $\ell = 4$ where we take $y = (x, d, \kappa, \eta)$, i.e. we simultaneously learn the value function and the strategy as a function of $(x, d)$, the resilience $\kappa$ and the instantaneous price impact $\eta$ for some fixed $\alpha$ and $\nu$.

We denote by $\mathbf{G}(y, u, \epsilon)$ the one-step transition function of $y$ given external noise $\epsilon$ and action $u$. For example,

$$\mathbf{G}((x, d, \kappa, \eta), u, \epsilon) = (x - u, (1 - \kappa)d + \eta u^\alpha + \epsilon, \kappa, \eta).$$

Our resolution of the DP equation (11) operates as-is with each of the underlying sub-steps. This means that considering a generic intermediate step $n$ of the backward recursion and given a surrogate $\hat{V}_{n+1}$ we first substitute it in place of $V_{n+1}$. Next, the expectation in (11) is over the stochastic shocks $\epsilon_n$ which are one-dimensional Gaussian random variables. We employ Gaussian quadrature to replace the respective integral with a finite sum. Specifically, we rely on the optimal quantization of Pagès *et al.* (2004) and Bally *et al.* (2005) to select $j = 1, \ldots, N'$ weights $w^j$ and respective knots $e^j$ to

approximate

$$\mathbb{E}\left[\hat{V}_{n+1}(\mathbf{G}(y,u,\epsilon_n)) \,\Big|\, Y_n = y\right] \simeq \sum_{j=1}^{N'} w^j \hat{V}_{n+1}\left(\mathbf{G}(y,u,\mathrm{e}^j)\right). \tag{27}$$

REMARK 4.1 One can straightforwardly consider non-Gaussian (e.g. heavy tailed) noise distributions for $\epsilon_n$ thereby providing a different nonlinear generalization. Non-Gaussian $\epsilon_n$ just reduces to taking a different set of $\mathrm{e}^j, w^j$'s.

The optimization over the number of shares to buy $u_n$ is done via a numerical optimizer, namely the standard gradient-free optimization routine (such as L-BFGS) that is available in any software package. Note that $u_n$ is scalar, allowing the use of fast one-dimensional root finding algorithms. In our implementation, we do not evaluate any gradients of $\hat{V}_{n+1}$ although that is feasible. To restrict to $u \in [0, x]$ and rule out any selling, we optimize on the above bounded interval, straightforwardly supported by such solvers.

Finally, it remains to construct $\hat{V}_n$. As a machine learning task, the goal is to learn the true input–output map $y \mapsto V_n(y)$. Such functional approximation, aka surrogate construction (Gramacy 2020), is carried out by selecting a collection of training inputs $y^{1:M} \in \mathscr{Y}$, evaluating (a noisy version of) $V_n(y^{1:M}) =: v_n^{1:M}$ and then fitting a statistical representation $\hat{V}_n(\cdot)$ that can interpolate to new, out-of-sample $y$'s. The evaluation of $v_n^m$ is achieved by direct computation via the nonlinear optimizer and the quantized integral on the right-hand-side of (11):

$$v_n^m := \inf_{u \in [0, x_{n-1}^m]} \left\{ (1 - \kappa^m) d_{n-1}^m u + \frac{\eta^m}{2} u^{\alpha^m + 1} + \nu^m (x_{n-1}^m - u)^2 \right.$$
$$\left. + \sum_{j=1}^{N'} w^j \hat{V}_{n+1}(\mathbf{G}(y_{n-1}^m, u, \mathrm{e}^j)) \right\}. \tag{28}$$

Observe that (28) uses $\hat{V}_{n+1}$ and so leads to a recursive construction backward in time, fully mimicking the dynamic programming equation. This recursion is instantiated with the exact terminal condition $\hat{V}_N(y) \equiv V_N(y) = ((1 - \kappa)d + \frac{\eta}{2}x^{\alpha})x$ and then run for $n = N - 1, N - 2, \ldots, 1$. Our notation emphasizes the pointwise nature of the optimization for $v_n^m$ by taking the static parameters $\kappa, \eta, \alpha, \nu$ as part of the training design, hence varying in $m$.

### 4.2. Policy approximation

The primary output of the numerical solver is the *execution strategy*, given in feedback form as $y \mapsto u_n^*(y)$. Indeed, the strategy is what the controller is ultimately after and yields a clear interpretation of how many shares the solver recommends to buy next. In contrast, the approximate value function $\hat{V}$ is harder to interpret (since it is not in pure monetary dollars but potentially also involves the abstract inventory costs) and moreover due to the intermediate approximations does not have any concrete probabilistic representation. Indeed, while the true $V$ is the expected cost of the strategy, $\hat{V}$ is *not*

an expectation on $[0, T]$, since it is obtained from one-step recursions.

Conventionally, $u^*$ is characterized as the $\arg\inf$ of the Bellman recursion (11), so that to obtain $u_n^*(y)$ one must re-do the optimization over $\hat{V}_n$. This is time-consuming, inefficient and non-transparent to the user. Instead we seek a direct representation for $u_n^*$ and in the spirit of the machine learning mindset (specifically actor-critic frameworks) propose to construct a second surrogate, auxiliary to the one describing $\hat{V}$. Accordingly, we construct a separate surrogate $\hat{u}_n(\cdot)$ that is trained based on the recorded $u_n^m$, the optimal execution amounts for each $y_{n-1}^m$. Note that the fitting of $\hat{u}_n$ is independent of the main loop above, so can be done in parallel with $\hat{V}_n$ or after-the-fact.

Even when the training inputs $u_n^m$ are in the range $[0, x_{n-1}^m]$, it is not generally guaranteed that this would be true for the fitted prediction $\hat{u}(y_{n-1}^m)$. To enforce this constraint, we train $\hat{u}$ based on the *fractions* $u_n^m / x_{n-1}^m \in [0, 1]$ and use a *sigmoid* transformation to intrinsically restrict all predictions (again interpreted as fractions of current inventory to be bought) into $(0, 1)$.

REMARK 4.2 In our setup, we first pointwise approximate $u_n(y_{n-1}^m)$ and then fit a statistical surrogate to those samples; in Huré *et al.* (2018) and Bachouch *et al.* (2018) the strategy is the opposite: first parametrize potential $u(\cdot; \theta_n)$ through a neural network with weights $\theta_n$; then use back-propagation to optimize the respective hyper-parameters $\theta_n$. In that sense, their resulting $y \mapsto u(y; \theta_n)$ is not a solution of any optimization problem, and there is no underlying dataset $(y_{n-1}^{1:M}, u_n^{1:M})$ like for our approach.

### 4.3. Neural network solvers

A popular class of surrogates consists of feed-forward neural networks (NN). NNs allow efficient fitting of high-dimensional parametric surrogates using back-propagation and a variety of stochastic optimization techniques, such as stochastic gradient descent.

An NN represents $\hat{V}_n$ as a composition, using linear hidden units at each layer, a user-chosen activation function across layers, and a user-selected number of layers. In our context, the precise architecture of the feed-forward NN is not conceptually important. The NN parameters are optimized via batch stochastic gradient descent. Relative to other statistical models, NN is *overparametrized* with thousands of parameters, known as the weights. Nevertheless NN is known to enjoy excellent empirical convergence (i.e. the algorithms find near-optimal weights), especially for large-scale datasets, including in other stochastic control applications (Han and E 2016, Bachouch *et al.* 2018, Huré *et al.* 2018, Ismail and Pham 2019).

Training an NN requires specifying the training inputs $y_n^{1:M} \equiv (x_n^{1:M}, d_n^{1:M}, \kappa_n^{1:M}, \eta_n^{1:M}) \in \mathscr{Y}$ and initializing the NN weights. For the former, we propose a space-filling experimental design, matching the standard approach in statistics. As default, we select a hyper-rectangular training domain $\bar{\mathscr{Y}}$ and sample each coordinate of $y$ uniformly and independently on the respective training interval, for example we sample the inventory $x_n \in [0, X_0]$. One may also implement joint

sampling in $\bar{\mathscr{Y}}$, such as Latin Hypercube Sampling (LHS). A further option is to use low-discrepancy Quasi Monte Carlo (QMC) sequences to achieve a space filling training set of arbitrary size $M$. Compared to i.i.d. Uniform sampling, LHS and QMC offer lower variance and better coverage, avoiding any clusters or gaps in the training locations, which is relevant when $M$ is relatively small compared to the dimension of $y$. Note that training sets are indexed by $n$; we sample fresh $y_n^{1:M}$'s at each step, so that the training inputs vary across $n$'s, although they have the same size and shape.

To initialize the weights, we found it very beneficial to *rescale* the inputs to the unit hypercube, which permits the use of standard NN weight priors (namely Truncated Normal with mean zero). Similarly, for $\hat{V}$ we re-scale the training outputs $v_n^m$ to be in the range $[0, 1]$ too. For training the policy surrogate $\hat{u}$ we apply a sigmoid activation function on the output layer of the NN that directly ensures that $\hat{u}(y) \in (0, 1)$. The latter fraction is multiplied by the current inventory $x$ to get the number of shares to trade.

Our implementation (see the supplementary Jupyter Notebook) employs the `TensorFlow` library in `Python`, which provides one of the most popular engines for NNs. We employ 'factory defaults' to construct our neural networks using the `tensorflow.keras.Sequential` architecture. This is a linear stack of layers; we use the same number of neurons per layer and the same activation function across layers. For the experiments below, we utilize 4 layers and 20 neurons, with the ELU activation function, $ELU(x) = x1_{\{x>0\}} + (e^x - 1)1_{\{x\le 0\}}$. Training uses the Adam algorithm with default learning rate, batch size of 64 and $E$ epochs. The latter represents a single pass through all training data, and many epochs are needed given the high noise in the underlying stochastic gradient descent optimizer of the NN weights. With the above choices, training a NN takes just a few lines in `TensorFlow`. Indeed, it takes more code to scale/re-scale the inputs and outputs than to actually build and fit the Neural Net. Other neural network libraries, such as `scikit-learn` or `PyTorch` could be straightforwardly substituted.

We end this section with a few final remarks:

- It is completely straightforward to modify the NN architecture. For instance, to replace a 4-layer 'deep' architecture with a single layer, it takes just commenting out 3 lines in our code. Similarly, one can add more layers with a single line change.
- Because we maintain the underlying Dynamic Programming paradigm, the NNs are fitted one-by-one. Therefore, one has complete flexibility in modifying any aspect of the fitting procedure to make it step-dependent. This includes the size $M$ and shape of the training set $y_{n-1}^{1:M}$; the parameters for the NN optimization, including the initial NN weights; the NN architecture, such as the number of neurons; and even the surrogate type itself. For example, one could mimic RL techniques to select time-dependent training regions $\mathscr{X}_n$ to reflect the natural time-dependency of the solution, such as the remaining order to fill $X_n$ decreasing over time.
- The convergence of the stochastic gradient descent to find a good $\hat{V}_n$ is quite slow. We find that $E \gg$

1000 epochs are necessary to achieve good results. The number of epochs is the primary determinant of fit quality, cf. section 5.3.
- Any surrogate can be re-trained/updated at any point of the overall backward loop. For example, we suggest training $\hat{V}_n$'s for all $n$'s and then training more (i.e. run the backward loop again with same training samples or newly generated ones) as one way to improve empirical convergence. This process is completely transparent and just requires loading the existing NN objects corresponding to $\hat{V}_n$'s rather than initializing new ones. Similarly, one can straightforwardly implement *warm starting*, using the fitted NN weights at step $n + 1$ as an initial guess for the weights of $\hat{V}_n$.
- A typical failure point for surrogates is unstable prediction when extrapolating beyond the range of the training region. Thus, care must be taken to select the training domain $\bar{\mathscr{Y}}$ to minimize extrapolation. Usually the modeler knows a priori the test cases of interest and so can ensure that the training range is at least as large. For example, below we wish to test for $\kappa \in \{0.4, 0.6, 0.8\}$ and therefore we train on $\kappa_n^{1:M} \in [0.38, 0.82]$ to mitigate any issues with prediction at or beyond the edge of the training domain.

REMARK 4.3 Other surrogate types could be employed in place of NNs. For example, Gaussian Processes (GP) (Rasmussen and Williams 2006) is a kernel regression method where the kernel hyperparameters are fitted using maximum likelihood. They are implemented in, e.g. `scikit-learn` in Python. GPs offer variable selection through automatic relevance determination, which allows to 'turn off' covariates/parameters that make little impact on the response. GP surrogates are known for excellent performance on limited datasets and are very popular for emulating expensive computer and stochastic experiments. Another surrogate type are LASSO linear models that explicitly project $\hat{V}_n$ onto the span of the basis functions $\{B_r(y)\}$, $\hat{V}_n(y) = \beta_0 + \sum_r \beta_r B_r(y)$. The respective coefficients are determined from the ($L_1$-penalized) least squares equations. Finally, we note that not all regression methods are appropriate; non-smooth frameworks like Random Forests would yield unstable or discontinuous estimates of $\hat{u}(\cdot)$ and therefore should not be applied.

### 4.4. Workflow

To summarize, the algorithmic workflow to solve the parametric optimal execution problem is given in Algorithm 1. We provide a `TensorFlow` implementation of the above in the supplementary fully-reproducible Jupyter notebook.

## 5. Numerical experiments

### 5.1. Comparison to reference model

With linear price impact $\alpha = 1$, the unconstrained strategy $(u_n^\circ)_{n=1,\dots,N}$ is available by parsing recursively the formulas

**Algorithm 1** Solving the Optimal Execution Problem

**Require:** $M$ (number of training locations)

1: Set $\hat{V}_N(y) \equiv V_N(y) = ((1 - \kappa)d + \frac{\eta}{2}x^\alpha)x$ (no approximation needed at terminal time)
2: **for** $n = N-1, N-2, \ldots, 1$ **do**
3:     Initialize the neural net surrogates $\hat{V}_n$ and $\hat{u}_n$
4:     Select the experimental design
    $y_{n-1}^{1:M} = (x_{n-1}^m, d_{n-1}^m, \kappa^m, \eta^m, \nu^m, \alpha^m)_{m=1,\ldots,M}$
5:     **for** $m = 1, \ldots, M$ **do**
6:         Evaluate $v_n^m$ in (28) on $y_{n-1}^m$ using a gradient-free optimizer; record the corresponding $u_n^m$
7:     **end for**
8:     Use the dataset $(y_{n-1}^{1:M}, v_n^{1:M})$ to fit the surrogate $\hat{V}_n(\cdot): \mathscr{Y} \to \mathbb{R}$
9:     Train the control surrogate $\hat{u}_n(\cdot)$ using $(y_{n-1}^{1:M}, u_n^{1:M}/x_{n-1}^{1:M})$
10: **end for**
11: Generate $M'$ out-of-sample forward paths $y^{1:M'}$
12: Record the cost $\check{v}^m, m = 1, \ldots, M'$ based on utilizing $\hat{u}_n(y_{n-1}^m)$ as the control at step $n$.
13: **return** Empirical average execution cost

$$\check{V}(0, Y_0) = \frac{1}{M'} \sum_{m=1}^{M'} \check{v}^m. \qquad (29)$$

in proposition 3.1. This yields a concrete benchmark to evaluate our numerical algorithms. In this section, we compare an NN surrogate to the above ground truth. Even though $u^\circ$ is not feasible (since it can and does turn negative in some states), this comparison is still useful. For the chosen parameter configurations, $u_n^\circ \gg 0$ (cf., figure 2) so that the non-negativity constraint is not binding on the vast proportion of the paths and hence $u^\circ$ is very nearly optimal for the constrained problem as well; that is, $u^\circ \approx u^*$ with $u^*$ denoting the minimizer in (8). This 'near'-optimality is confirmed by the closeness between the constrained NN-strategy $\hat{u}$ and the unconstrained $u^\circ$ strategy (LF), offering an additional consistency check on the NN solver.

To enable an apples-to-apples assessment of $(\hat{u}_n)_{n=0}^{N-1}$, we *fix* the set of 'noise' $\epsilon_n^{1:M'}$ that feed into the realized $D_n$'s, and employ this fixed database of $\epsilon$'s to generate the test forward trajectories both for the NN approximator and for the exact solution. Since $D_n$ depends on the past execution amounts, the trajectories of the approximating strategy will differ at each and every step $n$. Thus we are *not* making a pairwise comparison between $\hat{u}_n(\cdot)$ and $u_n^\circ(\cdot)$ from (17), but compare in terms of the final execution cost in (29). Indeed, due to stochastic fluctuations, on any given path the realized costs from the latter may be higher or lower than the costs of the benchmark strategy, however, the Law of Large Numbers guarantees that for $M'$ large, the *average* execution cost must be at least as much as the benchmark.

For illustration purposes, we train an NN that takes in the four inputs $(x, d, \kappa, \eta)$ and compute the corresponding $\hat{u}_n$. The respective four-dimensional training domain is taken to be a hyper-rectangle specified as $x \in [0, 10^5], d \in [0, 100], \kappa \in$

[0.38, 0.82], $\eta \in [1/900, 1/5000]$. The remaining parameters are fixed as $\nu = 0.00005, \sigma = 1$. We then select $M = 4000$ training points i.i.d. uniformly in the above domain, independently for each step $n$. Note that while we kept the same training domain across steps, one can vary this as $n$ changes. The ranges of the training domain are ultimately driven by the desired *test configurations*. For example, below we show the results for the test set with $\kappa = 0.4, \eta = 1/1000$ and initial condition $X_0 = 10^5, D_0 = 0$. The latter affect the range of $X_n$. Since $X_n \leq X_0$ and $X_N = 0$, we train on the range $[0, X_0]$. The range for $D_n$ depends on $D_0, X_0$ and $\eta$. With $X_0 = 10^5$, we expect to buy up to 50–60 K shares in the first step, which with $\eta \simeq 1/1000$ would lead to $D_n \in [40, 80]$. Note that $D_n$ is highly sensitive to $\eta$, hence the respective training range should reflect the bounds on $\eta$ values. The range for the resilience $\kappa$ covers the test case of $\kappa = 0.4$ and is simultaneously quite wide to cover a range of market conditions (calibration of $\kappa$ is known to be difficult). The range for the instantaneous impact $\eta$ is similarly chosen to cover a range of market conditions, with $\eta = 1/1000$ leading to impact that is 5 times stronger than that of $\eta = 1/5000$. The quantization of the conditional expectation in (27) uses $N' = 50$ knots.

The left panel of figure 3 shows the histogram of the difference between the realized execution costs $\check{v}^m$ coming from an NN approximator and the benchmark, across $10^4$ test trajectories. We observe that on average the former are 0.041% higher. Note that for about 11.6% of the paths, the realized costs from our approximate strategy were *less* than from the benchmark. Conversely, on more than 95.5% of the paths, the NN-based costs were less than 0.1% higher than those from the reference strategy, which is practically a very good level of accuracy.

The right panel of figure 3 compares the reference $u^\circ(y_{1:N}^{m'})$ and the NN-based control $\hat{u}(y_{1:N}^{m'})$ on one sample forward trajectory. We observe that the two strategies are very close in a pathwise sense as well, confirming the high approximation quality.

### 5.2. Nonlinear price impact

We now proceed to consider strategies with nonlinear price impact $\alpha \neq 1$. Two comparators are the linear feedback (LF) strategy $u^\circ$ defined by (17) and constrained to buys-only, which we denote by $u_n^{LF} := u_n^\circ \vee 0 \wedge X_n$, and the volume-weighted average policy, known as VWAP. The first $u^{LF}$ comparator utilizes the linear benchmark formula as a function of current $X_n, D_n$, in other way it postulates the counter-factual $\alpha = 1$ even when $\alpha$ is not unity. This means it will underestimate price impact when $\alpha > 1$, and will overestimate price impact when $\alpha < 1$. The above over- or under-estimation of price impact can lead to severe instability in the execution strategy. The second VWAP comparator fixes $u_n^{\text{VWAP}} \equiv X_0/N$. That strategy is completely model independent, and therefore its performance is little affected by $\alpha$. In that sense, the VWAP is the opposite of the linear feedback policy $u^{LF}$ which makes strong assumptions on the market environment; VWAP implements the same trades no matter the model parameters.

While for a fixed set of model parameters, there are multiple ways to construct a nonlinear solver, this would be
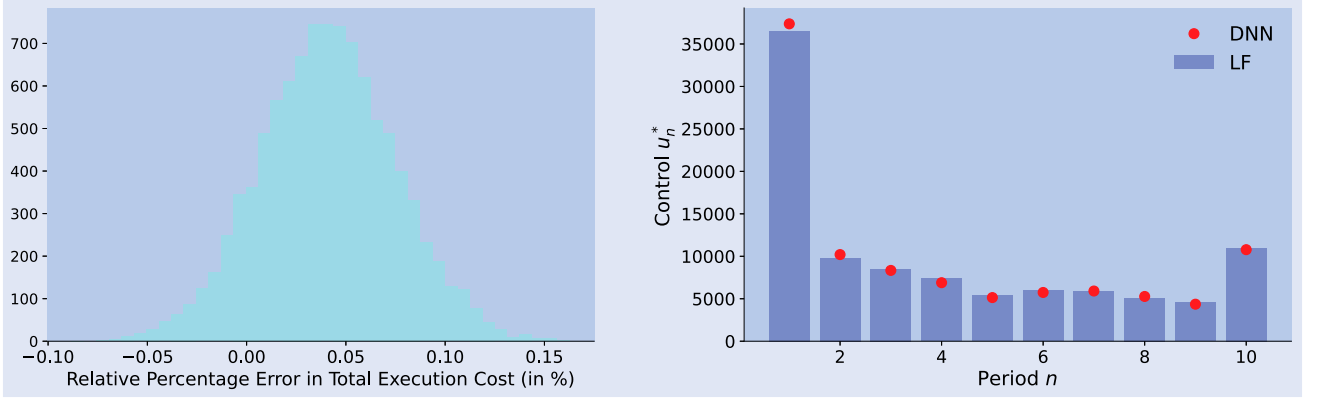
Figure 3. *Left:* Histogram of the relative error in final costs between the NN-strategy and the benchmark strategy based on $M' = 10^4$ forward trajectories. *Right:* Optimal policy versus the NN strategy along the same path (fixed set of $\epsilon_n$s). Parameters fixed at $\kappa = 0.4, \eta = 1/1000, \nu = 0.00005$ and $\sigma = 1$.
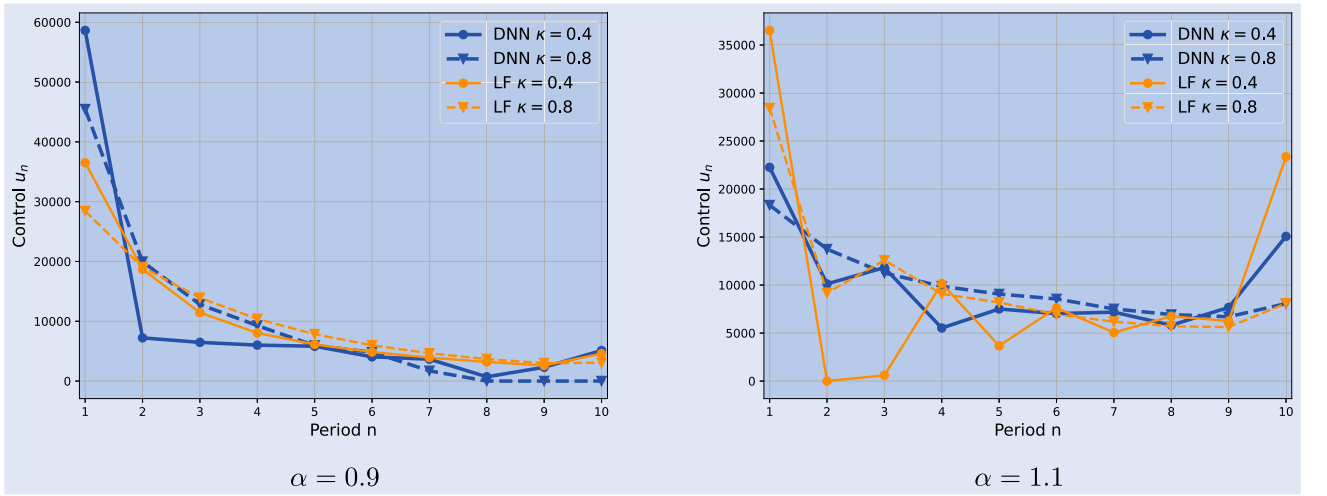


Figure 4. Execution strategy (mean values across $M' = 10^4$ simulations) for $X_0 = 100\,000, D_0 = 0$ and $N = 10$ with $\sigma = 1$, urgency parameter $\nu = 0.0001$, $\eta = 1/500$ and different price impact parameters $\kappa$ and $\alpha$, utilizing a 5D solver in $(x, d, \kappa, \eta, \alpha)$.

computationally intractable to do for a large collection of $(\kappa, \eta, \alpha)$. Consequently, our *parametric solver* is indispensable to provide a comprehensive solution across many parameter configurations. Moreover, since we train jointly across a range of $\alpha$, the linear case $\alpha = 1$ is covered and can be used to indirectly assess the fit quality. In other words, we expect the performance of the NN surrogate for $\alpha \neq 1$ to be *similar* to its performance when $\alpha = 1$.

Figure 4 compares the resulting NN-based strategies to the linear feedback program restricted to buy-only. We observe that the linear feedback strategy $u^{LF}$ is very sensitive to $\alpha$. For $\alpha > 1$, LF generates oscillatory strategies, cf. the right panel of figure 4. This occurs because the linear feedback underestimates the convex impact of trading on $(D_n)$, so that it first over-shoots relative to the target $D_{n+1}$, then under-shoots, etc. The unconstrained reference strategy actually tries to sell $u_2^\circ < 0$ in the second step $n = 2$, as well as in the 4*th* step. Forcing $u_n^{LF} \geq 0$, we still obtain strong oscillations, no buying at $n = 2$ and almost no buying at $n = 3$. In contrast, the (estimated) optimal strategy smoothly maintains $u_n > 0$ throughout. Conversely for $\alpha < 1$ (left panel), the LF strategy underpurchases in the first trade since it does not correctly

judge the reduced impact of large trades due to the concave price impact function.

For $\alpha = 0.9$, price impact is much weaker and as a result the inventory urgency penalty dominates and leads to an L-shaped strategy, with a lot of buying in the first step or two, and a trickle for the rest of the steps. For $\alpha = 1.1$, price impact is the dominant feature and yields U-shaped strategies, with largest trades at $n = 1$ and $n = N$. We also note that the dependence of the strategies on $\kappa, \eta$ remains complex for $\alpha \neq 1$. Since in all cases, total trades must add up to $X_0 = 100\,000$, as parameters change, the resulting effect on $u_n$ is non-monotone, i.e. more is traded at some steps and less in others. As a result, various (average) strategy curves cross each other, often more than once.

Table 1 reports the performance of an NN solver versus the LF one. Specifically, we use a 5D NN solver that is parametric in $(x, d, \kappa, \eta, \alpha)$. The NN strategy is found to lead to total execution costs that are 10–15% cheaper than LF, indicating that the differences observed in the respective strategies in figure 4 are material. Gains are larger for larger $\kappa$ and larger $\eta$, i.e. for configurations where price impact is more short-lived and more severe. While we do not have a 'gold standard' to

Table 1. Performance of NN 5D solver in $(x, d, \kappa, \eta, \alpha)$ versus LF; based on $10^4$ simulations with $\sigma = 1$ and inventory penalty of $\nu = 0.0001$.

| | $\kappa = 0.4$ | | $\kappa = 0.8$ | |
|---|---|---|---|---|
| | $\eta = 1/3000$ | $\eta = 1/500$ | $\eta = 1/3000$ | $\eta = 1/500$ |
| $\alpha = 0.9$ | 10.33 | 7.33 | 15.05 | 10.58 |
| $\alpha = 1.0$ | $-0.20$ | $-0.07$ | $-0.52$ | $-0.04$ |
| $\alpha = 1.1$ | 10.34 | 8.70 | 13.01 | 5.85 |

Note: We report relative percent difference of total execution cost using Linear Feedback as baseline. Thus positive values mean that the NN yields lower costs than LF, and negative means that LF outperforms. We expect positive values for $\alpha \neq 1$ and small negative values for $\alpha = 1$.

compare against, we may use the $\alpha = 1.0$ performance (where LF is exact up to the non-negativity constraint that is almost never binding for this parameter configuration) as a yardstick for NN solution quality, since the NN solver is trained across different choices of $\alpha$.

### 5.3. NN implementations

NN solvers necessarily contain multiple tuning parameters that must be chosen during implementation. It is a folk theorem that some finetuning is always necessary, one of the reasons that using NN is a bit of a 'black art'.

In our setting, the feed-forward neural networks for $\hat{V}_n$ and $\hat{u}_n$ are very straightforward and do not require any bells and whistles. As mentioned, this simplicity of implementation is one of the reasons that we advocate this problem as a good pedagogical case study for building NN solvers for stochastic control problems. In particular, the NN architecture plays little significance: as long as one has sufficient flexibility, the choice of the number of neurons, the number of layers, etc., is very much secondary. Consequently, we default to the 'canonical' setup with 16 neurons and 3 layers that has been used in multiple prior works.

Nevertheless, there are certainly some tuning parameters that affect performance, first and foremost the effort spent on training. The latter is driven by the size of the training set, and the number of training epochs. The next set of experiments investigates in more detail how these parameters impact solution quality. In table 2, we compare the average P&L of the NN strategy relative to LF for $\alpha = 1.0$ as we vary the number of epochs $E$ and the number of training points $M$. As expected, accuracy increases as either $M$ or $E$ increase. We observe that the running time is roughly linear in $M$ (since the latter is a straightforward loop) and sub-linear in $E$.

In addition, we also compare solvers that live in different dimensions, taking advantage of the fact that our implementation is fully dimension-agnostic and a single line change is needed to change $\ell$. For NN training purposes, that dimension $\ell$ does not matter, so the running time is constant as $\ell$ changes. However, as expected, smaller $\ell$ implies more dense training sets (since the volume of the training domain shrinks) and hence better accuracy. This reflects the fundamental property that learning a functional approximator is more laborious on a larger domain, and the respective 'volume' grows in $\ell$. Thus, for the same $M$, $E$, a 3D solver will be more accurate than a 4D one, and less accurate than a 2D one. This is the (computational) price to pay for learning simultaneously across multiple parameters.

Figure 5 further compares solvers in different dimensions against each other in the nonlinear price impact case $\alpha = 1.1$. We consider NN solvers in 2D that only take $(x, d)$ coordinates, as well as in 3D with $(x, d, \kappa)$, $(x, d, \eta)$ coordinates, in 4D with $(x, d, \kappa, \eta)$ and finally in 5D with $(x, d, \kappa, \eta, \alpha)$. The fact that all solvers yield very similar strategies is an empirical indication of the convergence of the NNs.

Finally, the right panel of figure 5 shows the fitted dependence of the control $u_n$ on $\kappa, \eta$ at $n = 3$. Such dependence plots are the raison d'être of parametric solvers, providing the modeler with a direct view of the sensitivity of the strategy to model parameters. Without a parametric solver, it would be prohibitively expensive to generate such surfaces through re-solving each configuration one-by-one. We observe that at this intermediate step $u_n$ shrinks in $\eta$ and in $\kappa$.

### 5.4. Square-Root price impact

We next investigate the special case where $\alpha = 0.5$. This 'square-root law' of price impact is advocated by some practitioners (c.f., Lillo *et al.* 2003, Almgren *et al.* 2005, Bacry *et al.* 2015) and has also been addressed numerically in Curato

Table 2. Average P&L error compared to the linear feedback reference strategy for $\alpha = 1$ across different NN implementations.

| | NN configuration | $\kappa = 0.4$ | $\kappa = 0.8$ | Time (min) |
|---|---|---|---|---|
| 4D w/$(x, d, \kappa, \eta)$ | $M = 1000, E = 1000$ | 0.10% | 1.43% | 2.73 |
| | $M = 2000, E = 1000$ | 0.20% | 0.88% | 5.39 |
| | $M = 2000, E = 2000$ | 0.05% | 0.46% | 8.66 |
| | $M = 4000, E = 2000$ | 0.03% | 0.28% | 16.97 |
| | $M = 8000, E = 3000$ | 0.02% | 0.27% | 42.36 |
| 3D w/$(x, d, \kappa)$ | $M = 2000, E = 2000$ | 0.04% | 0.24% | 8.38 |
| 2D w/$(x, d)$ | $M = 2000, E = 2000$ | 0.03% | 0.15% | 8.14 |

Note: In all cases, the NN has 3 layers with 16 neurons in each. $M$: number of training inputs; $E$: number of training epochs. The other test parameters are fixed at $\eta = 0.0001, \nu = 0.0001, \sigma = 1$, initial condition $X_0 = 100\,000, D_0 = 0, N = 10$, utilizing a 4D solver in $(x, d, \kappa, \eta)$, except for the last two rows. All running times are based on an Intel i7-11370H 3.0 GHz laptop with 32 GB RAM.
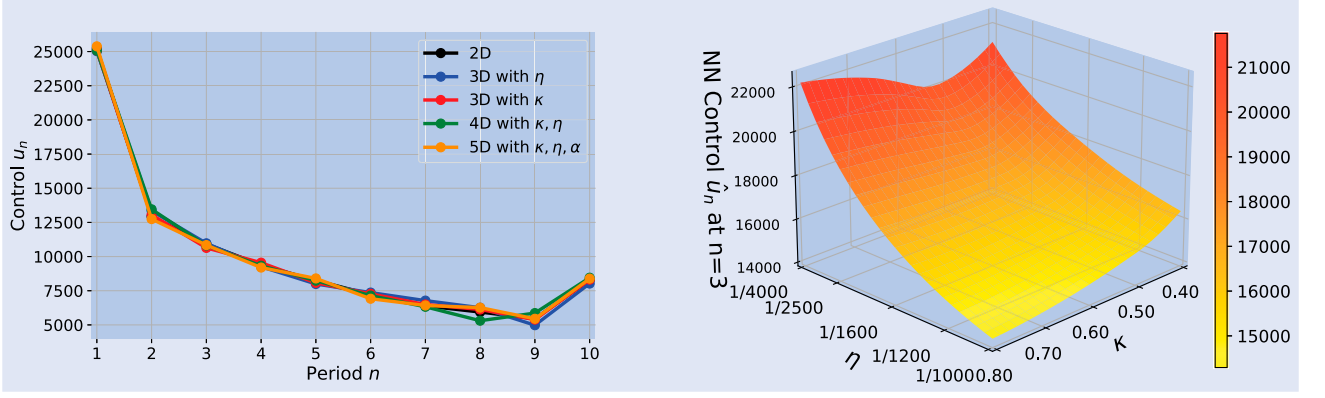
**Figure 5.** *Left:* Comparing different NN solvers with $\alpha = 1.1$ and $\kappa = 0.6, \eta = 0.001, \nu = 0.00005$. *Right:* Dependence of NN-based execution $u_n(x, d, \kappa, \eta)$ strategy on $\kappa$ and $\eta$, keeping other parameters fixed at $n = 3, x = 60\,000, d = 20$. We use the 5D solver shown on the left.
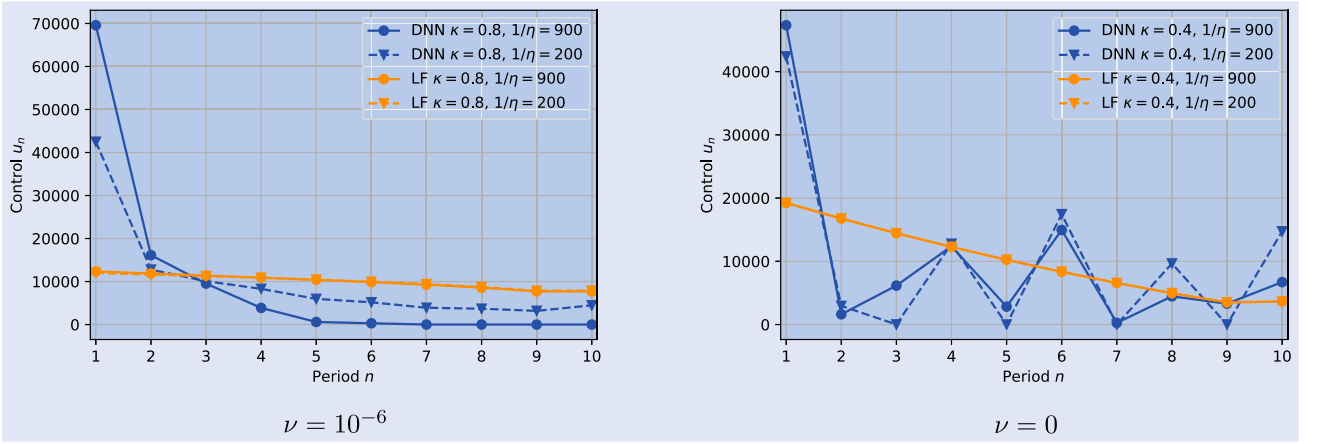


**Figure 6.** Execution strategy (mean values across $M' = 10^4$ simulations) for $X_0 = 100\,000, D_0 = 0$ and $N = 10$ in the square root case $\alpha = 0.5$ with $\sigma = 0.1$, different price impact parameters $\kappa$ and $\eta$, and different urgency parameters $\nu$, utilizing a 5D solver in $(x, d, \kappa, \eta, \nu)$.

*et al.* (2017) via a brute force optimization of the cost function. In contrast, our case study highlights once more the usefulness of our parametric solver in that it readily computes an optimal execution schedule jointly across a range of different price impact parameters $\kappa$ and $\eta$, as well as urgency rates $\nu$; and thus unveils with ease after a single round of training the solution's dependence on the latter under this square-root price impact regime. Specifically, for $\alpha = 0.5$ fixed, we train a 5D NN solver that is parametric in $(x, d, \kappa, \eta, \nu)$ over the hyper-rectangle $x \in [0, 10^5]$, $d \in [0, 1]$, $\kappa \in [0.35, 0.85]$, $\eta \in [1/100, 1/1000]$, $\nu \in [0, 10^{-5}]$ with $M = 8000$ i.i.d. uniformly selected training points, and set $\sigma = 0.1$.

Figure 6 compares the resulting NN-based strategies to the linear feedback benchmark case $u^{LF}$ and table 3 reports their performances. We first note that the obtained strategy is very sensitive to whether the urgency parameter $\nu$ is zero or not. This is sensible because with $\alpha = 0.5$, the magnitude of the incurred price impact is on the small scale and even further reduced by large values for $1/\eta$. As a consequence, as soon as $\nu$ becomes nonzero, the focus on rapidly reducing outstanding inventory dominates the overall order schedule; see the left panel in figure 6. In contrast, when inventory control is turned off (i.e. $\nu = 0$) the NN strategy exhibits an oscillatory behavior: peaks of large buy orders are interrupted by near to zero-volume orders; see the right panel in figure 6. This

**Table 3.** Performance of the 5D NN solver with $y = (x, d, \kappa, \eta, \nu)$ versus LF, based on $10^4$ simulations with $\sigma = 0.1$.

|  |  | $\eta = 1/900$ | $\eta = 1/200$ |
|---|---|---|---|
| $\nu = 10^{-6}$ | $\kappa = 0.8$ | 54.85 | 17.56 |
| $\nu = 0$ | $\kappa = 0.4$ | 30.40 | 28.11 |

Note: We report relative percent difference of total execution cost using Linear Feedback as baseline. Positive values mean that the NN yields lower costs than LF.

observation is somewhat consistent with the numerical results presented in Curato *et al.* (2017, section 4.4) where the computed strategy consists of a few bursts of buying interspersed with long periods of no trading. In terms of total execution costs, the NN solver always significantly outperforms the LF benchmark strategy in all considered parameter configurations for $\kappa, \eta, \nu$, see table 3. Apparently, the LF feedback policy overestimates the price impact which leads to an overall sub-optimal behavior.

### 5.5. Number of periods

Our NN algorithm trivially scales in the number of periods $N$ as it involves a simple loop over $n = N - 1, \ldots, 1$.
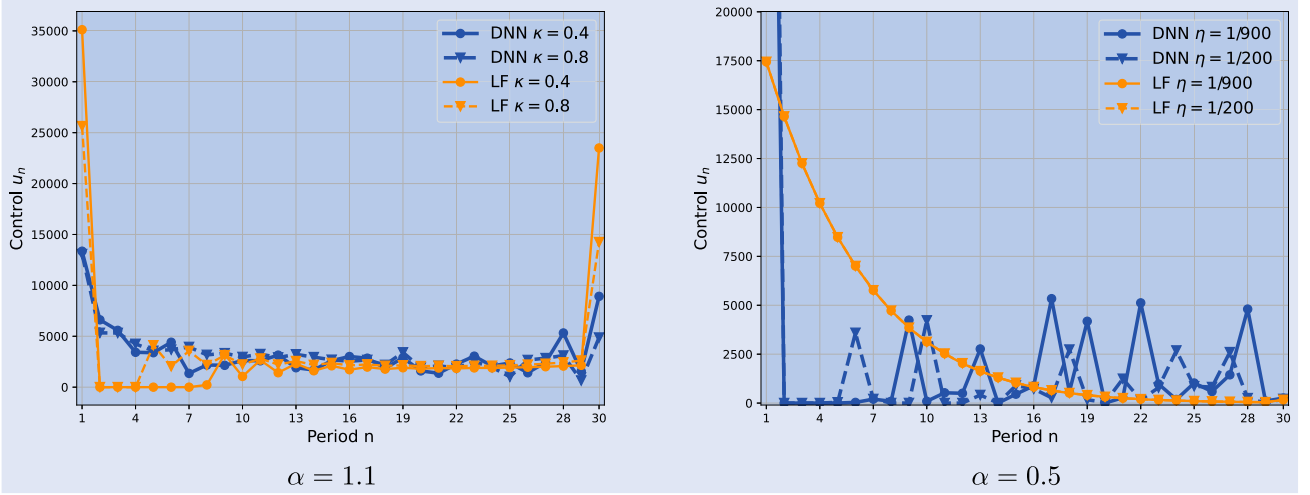
Figure 7. Execution strategy (mean values across $M' = 10^4$ simulations) for $X_0 = 100\,000$, $D_0 = 0$ and $N = 30$ periods. Left panel: $\alpha = 1.1$ with $\sigma = 1/\sqrt{3}$, urgency parameter $\nu = 0.0001/3$, $\eta = 1/500$, utilizing a 5D solver in $(x, d, \kappa, \eta, \alpha)$. Right panel: square-root price impact $\alpha = 0.5$ with $\sigma = 0.01$, $\nu = 0$ and $\eta = 1/200$, utilizing a 4D solver in $(x, d, \kappa, \eta)$.

Financially speaking, the most common interpretation is to fix the business time horizon $T$, and then pick the scheduling interval $\Delta t$, so that $N = T/\Delta t$. To illustrate this, we consider taking a larger number of intervals $N \gg 10$. To reflect the idea that $T$ is fixed, we need to re-scale some of the model parameters in terms of the frequency $\Delta t$. Specifically, since the dynamics of $(D_n)_{n=1,\ldots,N}$ are motivated by a discretization of a continuous-time kernel decay specification, we need to keep $\kappa^{(N)}\Delta t$ constant, in other words $\kappa^{(N)} \propto N^{-1}$, where we explicitly indicate the dependence of the resilience parameter on $N$. Indeed, as $N$ gets bigger, $\kappa^{(N)}$ should shrink so that the persistence of $D_n$ increases. Similarly, the continuous-time volatility $\sigma^{(N)}\sqrt{\Delta t}$ should be invariant, so that $\sigma^{(N)} \propto N^{-1/2}$. Finally, the inventory penalty is also proportional to business time, hence $\nu^{(N)} \propto N^{-1}$. In contrast, the impact parameter $\eta$ has no time-units, hence does not change as a function of $\Delta t$.

Returning to the closed-form formulas for $\alpha = 1$, we observe that more fine scheduling frequency does not alter the fundamental U-shape (in the case $\nu = 0$). In fact the initial and final trade amounts are only slightly affected by higher $N$, while the intermediate trades are roughly inversely proportional to $N^{-1}$, maintaining the same trading rate $u/\Delta t$ in business time. Similar intuition carries over to the solution when $\alpha > 1$. The left panel of figure 7 shows the execution strategy for $N = 30$ and $\alpha = 1.1$, keeping all other parameters as in figure 4 right, modulo re-scaling as described above. The right panel of figure 7 shows the execution strategy for $N = 30$ and $\alpha = 0.5$. The latter plot is comparable to the right panel of figure 6, taking zero inventory penalty $\nu = 0$. However, to prevent negative deviations $D_n$, we must significantly decrease the noise amplitude to $\sigma = 0.01$ (compared to $\sigma = 0.1$ in figure 6).

For $\alpha = 1.1$ and $N = 30$, the linear feedback strategy overshoots greatly, leading (for $\kappa = 0.133$) to no trades for periods $n = 2, \ldots, 7$, while the NN solution maintains a steady pace (with a decreasing trend due to the inventory penalty $\nu$) throughout. It gains about 10% in average cost savings compared to LF (11.2% for $\kappa = 0.4/3$ and 8.3% for $\kappa = 0.8/3$).

For $\alpha = 0.5$ and $N = 30$, we observe in figure 7 bursty trading taking advantage of the concave price impact which discourages making small trades. Instead, the strategy executes every 2–4 periods, letting $(D_n)$ mean-revert back to zero in between. As in figure 6, due to $D_0 = 0$ the first trade is very large, in our case about 65 000 for $\eta = 1/900$ and about 78 000 for $\eta = 1/200$.

While the proposed dynamic programming approach implies error back-propagation as the iterations over $n$ are stepped through, we see very reasonable performance as $N$ is increased *with exactly the same code*. We do recommend to employ a different strategy (e.g. RL-like with a single neural network that incorporates time-dependence) for $N \geq 50$ or so.

### 5.6. Multi-exponential decay kernel

We conclude our numerical experiments by investigating the performance of our NN algorithm on an extended version of our model formulated in section 2. Recall that the transient price impact process introduced in (2) for the studied buying program $u_n \geq 0$, $n = 1, \ldots, N$, is given by

$$D_n = (1-\kappa)^n d_0 + \sum_{j=1}^{n}(1-\kappa)^{n-j}\eta u_j^\alpha + \sum_{j=1}^{n}(1-\kappa)^{n-j}\epsilon_j$$

$$(n = 0, \ldots, N).$$

This deviation process belongs to a general class of decaying price impact processes, also called *propagator models* originally developed by Bouchaud *et al.* (2004, 2009), and can be written as

$$D_n = G_{n,0}d_0 + \sum_{j=1}^{n}G_{n,j}\eta u_j^\alpha + \sum_{j=1}^{n}G_{n,j}\epsilon_j \quad (n = 0, \ldots, N)$$

$$(30)$$

with an exponential decay kernel

$$G_{n,j} = (1-\kappa)^{n-j} \quad (0 \leq j \leq n \leq N). \tag{31}$$

Therefore, it is very sensible to consider extensions of our model by allowing for different decay kernels $(G_{n,j})_{0 \le j \le n \le N}$ in (31). Specifically, there is empirical evidence reported in the literature that price impact exhibits some memory effect and rather decays according to a power law function; cf., e.g. Bouchaud *et al.* (2004).

To retain a Markovian framework, we study in this section the generalization where the kernel $G$ is a convex combination of exponentially decaying kernels, namely it is of the form

$$G_{n,j} = \sum_{m=1}^{M} \zeta_m (1 - \kappa_m)^{n-j} \quad (0 \le j \le n \le N) \qquad (32)$$

for some $\zeta_m \in [0, 1]$ with $\sum_{m=1}^{M} \zeta_m = 1$, and $\kappa_m \in (0, 1]$. Plugging this back into (30) yields

$$D_n = \sum_{m=1}^{M} \zeta_m D_n^m \quad (n = 0, \dots, N), \qquad (33)$$

where

$$D_n^m := (1 - \kappa_m)^n d_0 + \sum_{j=1}^{n} (1 - \kappa_m)^{n-j} \eta u_j^\alpha + \sum_{j=1}^{n} (1 - \kappa_m)^{n-j} \epsilon_j$$

$$(n = 0, \dots, N).$$

In other words, the total price distortion $(D_n)_{n=0,\dots,N}$ in (33) is now driven by $M$ processes $(D_n^1, \dots, D_n^M)_{n=0,\dots,N}$ which decay at different timescales $\kappa_m$. The $D^m$'s are fully correlated with the dynamics $D_0^m = d_0$ and

$$D_n^m = (1 - \kappa_m) D_{n-1}^m + \eta u_n^\alpha + \epsilon_n.$$

Moreover, the dynamics in (2) generalize to $D_0 = d_0$,

$$D_n = \sum_{m=1}^{M} \zeta_m (1 - \kappa_m) D_{n-1}^m + \eta u_n^\alpha + \epsilon_n \quad (n = 1, \dots, N).$$

In the special case $\zeta_m = 1$, $\zeta_n = 0$ for all $n \ne m$, we retrieve the original setup from (2).

Augmenting the $d^m$'s to the state space and following the same reasoning as in section 2.1, the objective function in (8) becomes

$$\inf_{(u_n)_{n=1,\dots,N} \in \mathscr{A}_1} \mathbb{E}\left[ \sum_{n=1}^{N} \left\{ \left( \sum_{m=1}^{M} \zeta_m (1 - \kappa_m) D_{n-1}^m + \frac{\eta}{2} u_n^\alpha \right) u_n \right. \right.$$

$$\left. \left. + \nu (X_{n-1} - u_n)^2 \right\} \right];$$

the corresponding value functions in (9) are given by

$$V_n(x, d^1, \dots, d^M)$$

$$:= \inf_{(u_j)_{j=n,\dots,N} \in \mathscr{A}_n} \mathbb{E}\left[ \sum_{j=n}^{N} \left\{ \left( \sum_{m=1}^{M} \zeta_m (1 - \kappa_m) D_{n-1}^m + \frac{\eta}{2} u_j^\alpha \right) u_j \right. \right.$$

$$\left. + \nu (X_{j-1} - u_j)^2 \right\}$$

$$\left. \Big| X_{n-1} = x, D_{n-1}^m = d^m, m = 1, \dots, M \right]$$

for all $n \in \{1, \dots, N\}$.

We next illustrate the above extension in the case $M = 2$ so that the decay kernel is a mixture of two exponentials. Taking $\zeta_1 \equiv \zeta, \zeta_2 \equiv 1 - \zeta$ for the mixing weight $\zeta \in [0, 1]$, the associated dynamic programming (DP) equation in (10) and (11) modifies to

$$V_N(x, d^1, d^2) = \left( \zeta (1 - \kappa_1) \cdot d^1 + (1 - \zeta)(1 - \kappa_2) \cdot d^2 \right)$$

$$\cdot x + \frac{\eta}{2} x^{\alpha+1} \quad \text{and} \qquad (34)$$

$$V_n(x, d^1, d^2) = \inf_{u \in [0,x]} \mathbb{E}\left[ \left( \zeta (1 - \kappa_1) \cdot d^1 \right. \right.$$

$$+ (1 - \zeta)(1 - \kappa_2) \cdot d^2 \Big) \cdot u$$

$$+ \frac{\eta}{2} u^{\alpha+1} + \nu (x - u)^2$$

$$+ V_{n+1}(x - u, D_n^1, D_n^2) \Big|$$

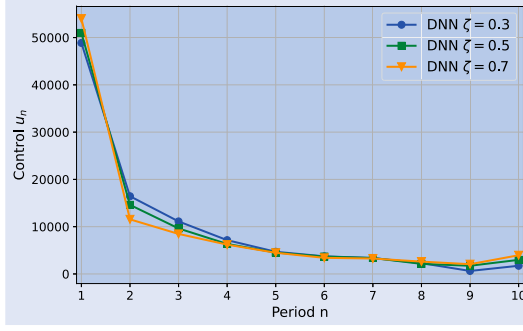$$X_{n-1} = x, D_{n-1}^1 = d^1, D_{n-1}^2 = d^2 \right] \qquad (35)$$

for all $n = N - 1, \dots, 1$.

Our NN algorithm can be easily extended to the DP equations in (34) and (35). In particular, we can also treat the mixing weight $\zeta$ as a model hyperparameter that can be trained upon. In figure 8, we illustrate the execution profiles of the NN solver $(\hat{u}_n(x, d^1, d^2, \zeta))_{n=1,\dots,N}$ for a model with a multi-exponential decay kernel as specified in (32) where $M = 2$, $\kappa_1 = 0.4$, $\kappa_2 = 0.8$ and several values of $\zeta$. We train the solver across the 4 inputs $(x, d^1, d^2, \zeta)$ with the latter in the range $\zeta \in [0.25, 0.75]$. The plot extends the configuration of the left panel in figure 4 to a multi-exponential propagator and shows the corresponding NN policy (mean values across $M' = 10^4$ simulations) in the concave price impact regime $\alpha = 0.9$ with $\sigma = 1$, $\nu = 0.0001$ and $\eta = 1/500$. As before, we take $X_0 = 100\,000$, $D_0 = 0$ and $N = 10$ periods and three different $\zeta$'s.

As the accompanying table shows, misspecifying the decay kernel is costly: the DNN solver beats an LF strategy that assumes linear price impact $\alpha = 1$ and a single decay parameter $\kappa$ by 9–12%. Of note, taking an exponential kernel with the larger $\kappa_2$ does better than using an averaged $\kappa = \zeta \kappa_1 + (1 - \zeta) \kappa_2$. Even when the impact is linear, $\alpha = 1.0$, the DNN beats the LF strategy (by 0.32–1.51%) due to its mis-specification via an exponential kernel.

## 6. Conclusion

In this article, we have investigated neural network surrogates for solving optimal execution problems across a range of model parameters. Our approach jointly learns an optimal strategy as a function of the stochastic system state and of the market configuration.

Figure 8. *Left*: Execution strategy (mean values across $M' = 10^4$ simulations) for $X_0 = 100\,000, D_0 = 0$ and $N = 10$ periods with bi-exponential decay kernel with $\kappa_1 = 0.4, \kappa_2 = 0.8$ and weight $\zeta \in \{0.3, 0.5, 0.7\}$. We use $\alpha = 0.9$ with $\sigma = 1, \nu = 0.0001$ and $\eta = 1/500$, utilizing a 4D solver in $(x, d^1, d^2, \zeta)$. *Right*: Percent gain of the DNN value function relative to the indicated LF comparator. Positive values mean that the DNN achieves lower execution costs.

The developed algorithm and the accompanying Jupyter Notebook can be used as a starting point for many other related analyses. For example, it would be straightforward to modify the code to handle parametric stochastic control problems of similar flavor (e.g. discrete-time hedging).

A further use case is to use the trained neural network as a building block in a more sophisticated setup. In particular, one may consider frameworks that explicitly account for model risk, in the sense of imprecisely known parameters. In the adaptive approach (including the Predictive Model Control popular in engineering), the modeler first develops *learning dynamics*, which convert static parameters, such as $\kappa$, into a stochastic process $(\hat{\kappa}_n)$, where $\hat{\kappa}_n$ is the best estimate of the book resilience at step $n$. Updating equations for $\hat{\kappa}_{n+1}$ in terms of the previous $\hat{\kappa}_n$ and new information from step $n + 1$ (such as using the Bayesian paradigm) yield dynamics that can be merged with those of $(X_n, D_n)$. One then *plugs-in* the resulting $\hat{\kappa}_n$ (and other similarly learned/updated parameters) into the NN-learned $\hat{u}(X_n, D_n, \hat{\kappa}_n)$ to obtain the adaptive strategy—which takes into account the latest parameter estimates, but does not solve the full Bellman equation. Conversely, one could also consider *robust* approaches that minimize $\hat{V}_n(\cdot)$ over feasible parameter settings to protect against a worst-case situation. The latter again requires access to the computed $\hat{V}_n$ as a building block. Finally, we may mention the *adaptive robust* approach (Bielecki *et al.* 2019) that combines dynamic learning with a worst-case min–max optimization to protect against incorrect estimates or mis-specified dynamics. The resulting numerical algorithms will be investigated in a separate, forthcoming sequel.

## 7. Proofs

We start with Lemma 7.1 which provides an intermediate computation relevant for showing $V_n^\circ(0, 0) < 0$ in (16) (existence of round-trips) and characterizing the optimal $u^\circ$ in in (17).

LEMMA 7.1 *For all $n = N, N - 1, \ldots, 2$ the constants $a_n, b_n, c_n$ recursively defined in (14) and (15) satisfy $2\eta + 4\nu + 4a_n - 4\eta b_n + 4\eta^2 c_n > 0$.*

*Proof* For $n = N$, we directly get from (14) that

$$2\eta + 4\nu + 4a_N - 4\eta b_N + 4\eta^2 c_N = 4\eta\kappa + 4\nu > 0.$$

For $n = N - 1$, using the definition in (15), one computes

$$2\eta + 4\nu + 4a_{N-1} - 4\eta b_{N-1} + 4\eta^2 c_{N-1}$$
$$= \frac{32\eta\kappa\nu + 16\nu^2 + 4\eta^2\kappa^2(4 - \kappa^2)}{4\eta\kappa + 4\nu} > 0.$$

The general claim can then be checked similarly with a tedious backward induction relying on the following recursive relation obtained from (15)

$$2\eta + 4\nu + 4a_n - 4\eta b_n + 4\eta^2 c_n$$
$$= 2\eta + 4\nu + 4(a_{n+1} + \nu) - 4\eta(1 - \kappa)b_{n+1}$$
$$+ 4\eta^2(1 - \kappa)^2 c_{n+1}$$
$$- \frac{(2(2\nu + 2a_{n+1} - \eta b_{n+1}) + 2\eta(1 - \kappa)(1 - b_{n+1} + 2c_{n+1}))^2}{2\eta + 4\nu + 4a_{n+1} - 4\eta b_{n+1} + 4\eta^2 c_{n+1}}$$
$$= 4\eta\kappa + 4\nu + 4\eta^2\kappa^2$$
$$\times \frac{2c_{n+1}(2a_{n+1} + 2\nu - \eta) - (1 - b_{n+1})^2}{2\eta + 4\nu + 4a_{n+1} - 4\eta b_{n+1} + 4\eta^2 c_{n+1}}. \quad (36)$$

■

*Proof of proposition 3.1.* In the linear case $\alpha = 1$, the unconstrained version of the optimal execution problem formulated in (12) is a linear quadratic stochastic control problem. Therefore, it is well known that for all $n \in \{1, \ldots, N\}$ the value functions $V_n^\circ(x, d)$ are linear quadratic in $x$ and $d$. This motivates the ansatz $V_n^\circ(x, d) = a_n x^2 + b_n xd + c_n d^2 + e_n$, where the coefficients $a_n, b_n, c_n, e_n \in \mathbb{R}$ are determined via backward induction by using the corresponding dynamic programming equations in (10) and (11).

First, the terminal condition in (10) yields $a_N = \frac{\eta}{2}, b_N = 1 - \kappa, c_N = 0$ as claimed in (14), as well as $e_N = 0$. Next, for the inductive step, let $n \in \{N - 1, \ldots, 1\}$. The dynamic programming equation in (11) (for the considered unconstrained version of the problem) yields

$$V_n^\circ(x, d) = \min_{u \in \mathbb{R}} \left\{ (1 - \kappa)du + \frac{\eta}{2}u^2 \right.$$

$$+ \nu(x-u)^2 + \mathbb{E}\left[ V_{n+1}^\circ(X_n, D_n) \mid \right.$$
$$\left. \times X_{n-1} = x, D_{n-1} = d \right]\Big\}$$
$$= \min_{u \in \mathbb{R}} \Big\{ (1-\kappa)du + \frac{\eta}{2}u^2 + \nu(x-u)^2$$
$$+ \mathbb{E}\left[ V_{n+1}^\circ \left(x-u, (1-\kappa)d + \eta u + \epsilon_n\right) \mid \right.$$
$$\left. X_{n-1} = x, D_{n-1} = d \right]\Big\}. \tag{37}$$

Plugging in $V_{n+1}^\circ(x,d) = a_{n+1}x^2 + b_{n+1}xd + c_{n+1}d^2 + e_{n+1}$ in (37) and solving the squares, we obtain

$$V_n^\circ(x,d)$$
$$= \min_{u \in \mathbb{R}} \Big\{ u^2\left(\frac{\eta}{2} + \nu + a_{n+1} - \eta b_{n+1} + \eta^2 c_{n+1}\right)$$
$$+ u\left((\eta b_{n+1} - 2\nu - 2a_{n+1})x\right.$$
$$+ (1 - b_{n+1} + 2\eta c_{n+1})(1-\kappa)d\big)$$
$$+ (\nu + a_{n+1})x^2 + (1-\kappa)b_{n+1}xd$$
$$+ (1-\kappa)^2 c_{n+1}d^2 + e_{n+1}$$
$$+ \left(\frac{1}{2}u\left(1 - 2b_{n+1} + 4\eta c_{n+1}\right)\right.$$
$$+ (b_{n+1}x + 2(1-\kappa)c_{n+1}d)\Big)$$
$$\times \mathbb{E}\left[\epsilon_n \mid X_{n-1} = x, D_{n-1} = d\right]$$
$$+ c_{n+1}\mathbb{E}\left[\epsilon_n^2 \mid X_{n-1} = x, D_{n-1} = d\right]\Big\}$$
$$= \min_{u \in \mathbb{R}} \Big\{ u^2\left(\frac{\eta}{2} + \nu + a_{n+1} - \eta b_{n+1} + \eta^2 c_{n+1}\right)$$
$$+ u\left((\eta b_{n+1} - 2\nu - 2a_{n+1})x\right.$$
$$+ (1 - b_{n+1} + 2\eta c_{n+1})(1-\kappa)d\big)$$
$$+ (\nu + a_{n+1})x^2 + (1-\kappa)b_{n+1}xd + (1-\kappa)^2 c_{n+1}d^2$$
$$+ e_{n+1} + c_{n+1}\sigma^2\Big\}, \tag{38}$$

where we used the fact that $\mathbb{E}[\epsilon_n \mid X_{n-1}, D_{n-1}] = \mathbb{E}[\epsilon_n] = 0$ as well as $\mathbb{E}[\epsilon_n^2 \mid X_{n-1}, D_{n-1}] = \mathbb{E}[\epsilon_n^2] = \sigma^2$. Minimizing (38) with respect to $u$ gives

$$u = -\frac{\begin{array}{c}(\eta b_{n+1} - 2\nu - 2a_{n+1})x \\ +(1 - b_{n+1} + 2\eta c_{n+1})(1-\kappa)d\end{array}}{\eta + 2\nu + 2a_{n+1} - 2\eta b_{n+1} + 2\eta^2 c_{n+1}} \tag{39}$$

and hence the feedback policy $u_n^\circ$ as claimed in (17). In particular, note that it follows from Lemma 7.1 that $\eta + 2\nu + 2a_{n+1} - 2\eta b_{n+1} + 2\eta^2 c_{n+1} > 0$ and that $u$ in (39) is indeed the unique minimum in (38). Moreover, inserting (39) back into (38) yields

$$V_n^\circ(x,d) = -\frac{\begin{array}{c}((\eta b_{n+1} - 2\nu - 2a_{n+1})x \\ + (1 - b_{n+1} + 2\eta c_{n+1})(1-\kappa)d)^2\end{array}}{2\eta + 4\nu + 4a_{n+1} - 4\eta b_{n+1} + 4\eta^2 c_{n+1}}$$
$$+ (\nu + a_{n+1})x^2 + (1-\kappa)b_{n+1}xd$$

$$+ (1-\kappa)^2 c_{n+1}d^2 + c_{n+1}\sigma^2 + e_{n+1}$$
$$= \left(\nu + a_{n+1} - \frac{(\eta b_{n+1} - 2\nu - 2a_{n+1})^2}{\begin{array}{c}2\eta + 4\nu + 4a_{n+1} \\ -4\eta b_{n+1} + 4\eta^2 c_{n+1}\end{array}}\right)x^2$$
$$+ (1-\kappa)\left(b_{n+1} - \frac{\begin{array}{c}(\eta b_{n+1} - 2\nu - 2a_{n+1}) \\ (1 - b_{n+1} + 2\eta c_{n+1})\end{array}}{\begin{array}{c}\eta + 2\nu + 2a_{n+1} \\ -2\eta b_{n+1} + 2\eta^2 c_{n+1}\end{array}}\right)xd$$
$$+ (1-\kappa)^2\left(c_{n+1} - \frac{(1 - b_{n+1} + 2\eta c_{n+1})^2}{\begin{array}{c}2\eta + 4\nu + 4a_{n+1} \\ -4\eta b_{n+1} + 4\eta^2 c_{n+1}\end{array}}\right)d^2$$
$$+ c_{n+1}\sigma^2 + e_{n+1},$$

which implies the desired recursive formulas provided in (15) and the representation of the value function in (16). Also note in (17) that $u_1^\circ$ is just a deterministic constant in $\mathbb{R}$ and that $u_n^\circ$ is normally distributed for all $n = 2, \ldots, N$. Indeed, since $u_n^\circ$ is linear in $X_{n-1}^\circ$ and $D_{n-1}^\circ$, and the state variables $X_{n-1}^\circ$ and $D_{n-1}^\circ$ are linear in $(u_j^\circ)_{j=1,\ldots,n-1}$ and the i.i.d. zero-mean Gaussian noise $(\epsilon_j)_{j=1,\ldots,n-1}$, one checks that $u_n^\circ$ is ultimately just a linear transformation of $(\epsilon_j)_{j=1,\ldots,n-1}$. As a direct consequence, we can conclude that $(u_n^\circ)_{n=1,\ldots,N}$ is an admissible strategy in the set $\mathscr{A}_1^\circ$ as defined in (13). Finally, the representation of $D^\circ$ in (18) follows directly from its state dynamics in (2). ∎

*Proof of proposition 3.2.* In the case $\sigma = 0$ the optimal control problem in (12) (with $\alpha = 1$) is deterministic. We can introduce the corresponding cost functional $C : \mathbb{R}^N \to \mathbb{R}$ given by

$$C(u_1, \ldots, u_N) \triangleq \sum_{n=1}^N \left\{\left((1-\kappa)D_{n-1} + \frac{\eta}{2}u_n\right)u_n\right.$$
$$+ \nu(X_{n-1} - u_n)^2\Big\}$$
$$= \frac{\eta}{2}\sum_{n=1}^N u_n^2 + (1-\kappa)\sum_{n=1}^N D_{n-1}u_n$$
$$+ \nu\sum_{n=1}^N (X_{n-1} - u_n)^2. \tag{40}$$

For all $n = 1, \ldots, N$, considering the state variables $X_n(u_1, \ldots, u_N) \triangleq X_n$ and $D_n(u_1, \ldots, u_N) \triangleq D_n$ in (3) and (2) as functions in $u_1, \ldots, u_N \in \mathbb{R}$, we note that

$$\frac{\partial X_n}{\partial u_i} = -1, \quad \frac{\partial D_n}{\partial u_i} = \eta(1-\kappa)^{n-i} \quad (1 \le i \le n).$$

In particular, we have the relation

$$\frac{\partial D_n}{\partial u_i} = (1-\kappa)\frac{\partial D_n}{\partial u_{i+1}} \quad (1 \le i \le n-1). \tag{41}$$

Therefore, for all $i \in \{1, \ldots, N-1\}$ we can compute

$$\frac{\partial C}{\partial u_i} = \eta u_i + (1-\kappa)D_{i-1}$$

$$+ (1 - \kappa) \sum_{n=i+1}^{N} \frac{\partial D_{n-1}}{\partial u_i} u_n - 2v \sum_{n=i}^{N} X_n$$

$$= \eta u_i + (1 - \kappa) D_{i-1}$$

$$+ (1 - \kappa) \left( \frac{\partial D_i}{\partial u_i} u_{i+1} + \sum_{n=i+2}^{N} \frac{\partial D_{n-1}}{\partial u_i} u_n \right)$$

$$- 2v \sum_{n=i}^{N} X_n$$

$$= \eta u_i + (1 - \kappa) D_{i-1} + (1 - \kappa) \eta u_{i+1}$$

$$+ (1 - \kappa)^2 \sum_{n=i+2}^{N} \frac{\partial D_{n-1}}{\partial u_{i+1}} u_n - 2v \sum_{n=i}^{N} X_n, \qquad (42)$$

where we used (41) in the last step. Similarly,

$$\frac{\partial C}{\partial u_{i+1}} = \eta u_{i+1} + (1 - \kappa) D_i + (1 - \kappa) \sum_{n=i+2}^{N} \frac{\partial D_{n-1}}{\partial u_{i+1}} u_n$$

$$- 2v \sum_{n=i+1}^{N} X_n. \qquad (43)$$

Hence, using (43) in (42) we obtain the recursive equation

$$\frac{\partial C}{\partial u_i} = (\eta u_i + (1 - \kappa) D_{i-1})(2\kappa + \kappa^2) - 2v\kappa \sum_{n=i+1}^{N} X_n$$

$$- 2v X_i + (1 - \kappa) \frac{\partial C}{\partial u_{i+1}} \quad (1 \le i \le N - 1). \qquad (44)$$

Next, minimizing (40) under the constraint $X_0 - \sum_{n=1}^{N} u_n = 0$ and denoting $\lambda \in \mathbb{R}$ the Lagrangian multiplier, we obtain the first-order conditions

$$\frac{\partial C}{\partial u_i} = \lambda \quad (i = 1, \ldots, N), \qquad (45)$$

which, together with (44), can be rewritten as

$$\lambda \kappa = \kappa (2 - \kappa)(\eta u_i + (1 - \kappa) D_{i-1}) - 2v\kappa \sum_{n=i+1}^{N} X_n$$

$$- 2v X_i \quad (i = 1, \ldots, N - 1). \qquad (46)$$

Computing the differences of the equations in (46) for successive $i$ and $i-1$ (for $i \in \{2, \ldots, N-1\}$) and rearranging the terms yields the recursive formula

$$u_i = \tilde{a} u_{i-1} + \tilde{b} D_{i-2} + \tilde{c} X_{i-2}$$

$$= \tilde{a} u_{i-1} + \tilde{b}(1 - \kappa)^{i-2} d_0 + \sum_{j=1}^{i-2} (\tilde{b} \eta (1 - \kappa)^{i-2-j} - \tilde{c}) u_j$$

$$+ \tilde{c} X_0 \quad (i = 2, \ldots, N - 1), \qquad (47)$$

where

$$\tilde{a} := \frac{\kappa(2\kappa\eta - \kappa^2\eta + 2v)}{2\kappa\eta - \kappa^2\eta - 2\kappa v + 2v},$$

$$\tilde{b} := \frac{\kappa^2(2 - 3\kappa + \kappa^2)}{2\kappa\eta - \kappa^2\eta - 2\kappa v + 2v},$$

$$\tilde{c} := \frac{-2\kappa v}{2\kappa\eta - \kappa^2\eta - 2\kappa v + 2v}. \qquad (48)$$

Due to its linear structure, the recursion in (47) can be solved explicitly. We obtain the representation

$$u_i = \mathfrak{a}_i u_1 + \mathfrak{b}_i d_0 + \mathfrak{c}_i X_0 \quad (i = 1, \ldots, N) \qquad (49)$$

where $\mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in \mathbb{R}^N$ are given by $\mathfrak{a}_1 := 1, \mathfrak{b}_1 := \mathfrak{c}_1 := 0, \mathfrak{a}_2 := \tilde{a}, \mathfrak{b}_2 := \tilde{b}, \mathfrak{c}_2 := \tilde{c},$

$$\mathfrak{a}_i := \tilde{a} \cdot \mathfrak{a}_{i-1} + \sum_{j=1}^{i-2} \left( \tilde{b} \eta (1 - \kappa)^{i-2-j} - \tilde{c} \right) \mathfrak{a}_j,$$

$$\mathfrak{b}_i := \tilde{a} \cdot \mathfrak{b}_{i-1} + \tilde{b}(1 - \kappa)^{i-2} + \sum_{j=1}^{i-2} \left( \tilde{b} \eta (1 - \kappa)^{i-2-j} - \tilde{c} \right) \mathfrak{b}_j,$$

$$\mathfrak{c}_i := \tilde{a} \cdot \mathfrak{c}_{i-1} + \tilde{c} + \sum_{j=1}^{i-2} \left( \tilde{b} \eta (1 - \kappa)^{i-2-j} - \tilde{c} \right) \mathfrak{c}_j, \qquad (50)$$

for $i = 3, \ldots, N - 1$, as well as $\mathfrak{a}_N := -\sum_{j=1}^{N-1} \mathfrak{a}_j, \mathfrak{b}_N := -\sum_{j=1}^{N-1} \mathfrak{b}_j, \mathfrak{c}_N := 1 - \sum_{j=1}^{N-1} \mathfrak{c}_j$, which comes from the terminal condition $u_N = X_{N-1}$. Moreover, (49) implies the representation

$$X_i = \mathfrak{a}_{i+1}^x u_1 + \mathfrak{b}_{i+1}^x d_0 + \mathfrak{c}_{i+1}^x X_0 \quad (i = 0, \ldots, N - 1), \qquad (51)$$

where $\mathfrak{a}^x, \mathfrak{b}^x, \mathfrak{c}^x \in \mathbb{R}^N$ are given by $\mathfrak{a}_1^x := \mathfrak{b}_1^x := 0, \mathfrak{c}_1^x := 1$ and

$$\mathfrak{a}_{i+1}^x := -\sum_{j=1}^{i} \mathfrak{a}_j, \quad \mathfrak{b}_{i+1}^x := -\sum_{j=1}^{i} \mathfrak{b}_j, \quad \mathfrak{c}_{i+1}^x := 1 - \sum_{j=1}^{i} \mathfrak{c}_j$$

$$(i = 1, \ldots, N - 1); \qquad (52)$$

as well as

$$D_i = \mathfrak{a}_{i+1}^d u_1 + \mathfrak{b}_{i+1}^d d_0 + \mathfrak{c}_{i+1}^d X_0 \quad (i = 0, \ldots, N - 1), \qquad (53)$$

where $\mathfrak{a}^d, \mathfrak{b}^d, \mathfrak{c}^d \in \mathbb{R}^N$ are defined as $\mathfrak{a}_1^d := 0, \mathfrak{b}_1^d := 1, \mathfrak{c}_1^d := 0$ and

$$\mathfrak{a}_{i+1}^d := \sum_{j=1}^{i} \eta (1 - \kappa)^{i-j} \mathfrak{a}_j,$$

$$\mathfrak{b}_{i+1}^d := (1 - \kappa)^i + \sum_{j=1}^{i} \eta (1 - \kappa)^{i-j} \mathfrak{b}_j$$

$$\mathfrak{c}_{i+1}^d := \sum_{j=1}^{i} \eta (1 - \kappa)^{i-j} \mathfrak{c}_j \quad (i = 1, \ldots, N - 1). \qquad (54)$$

In other words, the first-order conditions in (46), together with the terminal state constraint $X_N = 0$, allow to express $u_2, \ldots, u_N$ explicitly in terms of $u_1$ via (49) and it remains to

minimize the costs in (40) as a linear quadratic function in $u_1$ only, namely

$$C(u_1, \ldots, u_N) = \frac{\eta}{2} \sum_{n=1}^{N} u_n^2 + \nu \sum_{n=1}^{N} (X_{n-1} - u_n)^2$$

$$+ (1 - \kappa) \sum_{n=1}^{N} D_{n-1} u_n$$

$$= \frac{\eta}{2} \sum_{n=1}^{N} (\mathfrak{a}_n u_1 + \mathfrak{b}_n d_0 + \mathfrak{c}_n X_0)^2$$

$$+ \nu \sum_{n=1}^{N-1} (\mathfrak{a}_{n+1}^x u_1 + \mathfrak{b}_{n+1}^x d_0 + \mathfrak{c}_{n+1}^x X_0)^2$$

$$+ (1 - \kappa) \sum_{n=1}^{N} (\mathfrak{a}_n^d u_1 + \mathfrak{b}_n^d d_0 + \mathfrak{c}_n^d X_0)$$

$$\times (\mathfrak{a}_n u_1 + \mathfrak{b}_n d_0 + \mathfrak{c}_n X_0)$$

$$= \left[ \frac{1}{2} \eta \mathfrak{a}^\top \mathfrak{a} + (1 - \kappa)(\mathfrak{a}^d)^\top \mathfrak{a} \right.$$

$$\left. + \nu (\mathfrak{a}^x)^\top \mathfrak{a}^x \right] u_1^2$$

$$+ \left[ \left( \mathfrak{a}^\top (\eta \mathfrak{b} + (1 - \kappa) \mathfrak{b}^d) \right. \right.$$

$$\left. + (1 - \kappa)(\mathfrak{a}^d)^\top \mathfrak{b} + 2\nu (\mathfrak{a}^x)^\top \mathfrak{b}^x \right) d_0$$

$$+ \left( \mathfrak{a}^\top (\eta \mathfrak{c} + (1 - \kappa) \mathfrak{c}^d) \right.$$

$$\left. \left. + (1 - \kappa)(\mathfrak{a}^d)^\top \mathfrak{c} + 2\nu (\mathfrak{a}^x)^\top \mathfrak{c}^x \right) X_0 \right] u_1$$

$$+ \left( \frac{1}{2} \eta \mathfrak{b}^\top \mathfrak{b} + \nu (\mathfrak{b}^x)^\top \mathfrak{b}^x \right.$$

$$\left. + (1 - \kappa)(\mathfrak{b}^d)^\top \mathfrak{b} \right) d_0^2$$

$$+ \left( \frac{1}{2} \eta \mathfrak{c}^\top \mathfrak{c} + \nu (\mathfrak{c}^x)^\top \mathfrak{c}^x \right.$$

$$\left. + (1 - \kappa)(\mathfrak{c}^d)^\top \mathfrak{c} - \nu \right) X_0^2$$

$$+ \left( \eta \mathfrak{b}^\top \mathfrak{c} + 2\nu (\mathfrak{b}^x)^\top \mathfrak{c}^x + (1 - \kappa)((\mathfrak{b}^d)^\top \mathfrak{c} \right.$$

$$\left. + (\mathfrak{c}^d)^\top \mathfrak{b}) \right) X_0 d_0$$

which yields the claims in (20), (21), (22). ∎

*Proof of corollary 3.2.* In the case $\nu = 0$, the constants introduced in (48) reduce to $\tilde{a} = \kappa$, $\tilde{b} = \kappa(2 - 3\kappa + \kappa^2)/(2\eta - \kappa\eta)$ and $\tilde{c} = 0$. Moreover, direct computations reveal that the coefficients in (50) simplify to

$$\mathfrak{a}_i = \kappa, \quad \mathfrak{b}_i = \tilde{b}, \quad \mathfrak{c}_i = 0 \quad (i = 2, \ldots, N - 1),$$

as well as $\mathfrak{a}_N = -1 - (N-2)\kappa$, $\mathfrak{b}_N = -(N-2)\tilde{b}$, $\mathfrak{c}_N = 1$. This yields the claim in (24). For the coefficients in (52) we obtain

$$\mathfrak{a}_i^x = -1 - (i-1)\kappa, \quad \mathfrak{b}_i^x = -(i-1)\tilde{b}, \quad \mathfrak{c}_i^x = 1$$

$$(i = 2, \ldots, N),$$

and for the coefficients in (54) we have

$$\mathfrak{a}_i^d = \eta, \quad \mathfrak{b}_i^d = 1 - \kappa, \quad \mathfrak{c}_i^d = 0 \quad (i = 2, \ldots, N).$$

This yields the claim in (25). Finally, directly computing $\hat{b}$ and $\hat{c}$ as defined in (19) gives the expressions in (23). ∎

## 8. Open Scholarship

This article has earned the Center for Open Science badge for Open Materials. The materials are openly accessible at https://github.com/moritz_voss/Parametric_Optimal_Execution_ML.

### Disclosure statement

No potential conflict of interest was reported by the author(s).

### ORCID

*Mike Ludkovski* http://orcid.org/0000-0001-7887-3870
*Moritz Voß* http://orcid.org/0000-0002-5047-6280

## References

Ackermann, J., Kruse, T. and Urusov, M., Càdlàg semimartingale strategies for optimal trade execution in stochastic order book models. *Finance Stoch.*, 2021a, **25**(4), 757–810.

Ackermann, J., Kruse, T. and Urusov, M., Optimal trade execution in an order book model with stochastic liquidity parameters. *SIAM J. Financial Math.*, 2021b, **12**(2), 788–822.

Al-Aradi, A., Correia, A., Naiff, D., Jardim, G. and Saporito, Y., Solving nonlinear and high-dimensional partial differential equations via deep learning. Preprint, 2018. arXiv:1811.08782.

Al-Aradi, A., Correia, A., Naiff, D. d. F., Jardim, G. and Saporito, Y., Applications of the deep Galerkin method to solving

partial integro-differential and Hamilton-Jacobi-Bellman equations. Preprint, 2019. arXiv:1912.01455.

Alfonsi, A. and Acevedo, J.I., Optimal execution and price manipulations in time-varying limit order books. *Appl. Math. Finance*, 2014, **21**(3), 201–237.

Alfonsi, A. and Blanc, P., Dynamic optimal execution in a mixed-market-impact Hawkes price model. *Finance Stoch.*, 2016, **20**(1), 183–218.

Alfonsi, A., Fruth, A. and Schied, A., Optimal execution strategies in limit order books with general shape functions. *Quant. Finance*, 2010, **10**(2), 143–157.

Alfonsi, A., Fruth, A. and Schied, A., Constrained portfolio liquidation in a limit order book model. In *Advances in Mathematics of Finance*, Polish Acad. Sci. Inst. Math, 83, pp. 9–25, 2008 (Banach Center Publications: Warsaw, Poland).

Alfonsi, A. and Schied, A., Optimal trade execution and absence of price manipulations in limit order book models. *SIAM J. Financial Math.*, 2010, **1**(1), 490–522.

Alfonsi, A. and Schied, A., Capacitary measures for completely monotone kernels via singular control. *SIAM J. Control Optim.*, 2013, **51**(2), 1758–1780.

Alfonsi, A., Schied, A. and Slynko, A., Order book resilience, price manipulation, and the positive portfolio problem. *SIAM J. Financial Math.*, 2012, **3**(1), 511–533.

Almgren, R. and Chriss, N., Optimal execution of portfolio transactions. *J. Risk*, 2001, **3**(2), 5–39.

Almgren, R., Thum, C., Hauptmann, E. and Li, H., Direct estimation of equity market impact. *Risk*, 2005, **18**(7), 58–62.

Bachouch, A., Huré, C., Langrené, N. and Pham, H., Deep neural networks algorithms for stochastic control problems on finite horizon, part 2: Numerical applications. Preprint, 2018. arXiv:1812.05916.

Bacry, E., Iuga, A., Lasnier, M. and Lehalle, C.-A., Market impacts and the life cycle of investors orders. *Mark. Microstruct. Liquidity*, 2015, **01**(02), 1550009.

Bally, V., Pagès, G. and Printems, J., A quantization tree method for pricing and hedging multidimensional American options. *Math. Financ.*, 2005, **15**(1), 119–168.

Bank, P. and Fruth, A., Optimal order scheduling for deterministic liquidity patterns. *SIAM J. Financial Math.*, 2014, **5**(1), 137–152.

Becherer, D., Bilarev, T. and Frentrup, P., Optimal liquidation under stochastic liquidity. *Finance Stoch.*, 2018, **22**(1), 39–68.

Bertsimas, D. and Lo, A.W., Optimal control of execution costs. *J. Financial Mark.*, 1998, **1**(1), 1–50.

Bielecki, T.R., Chen, T., Cialenco, I., Cousin, A. and Jeanblanc, M., Adaptive robust control under model uncertainty. *SIAM J. Control Optim.*, 2019, **57**(2), 925–946.

Bouchaud, J.-P., Farmer, J.D. and Lillo, F., How markets slowly digest changes in supply and demand. In *Handbook of Financial Markets: Dynamics and Evolution*, edited by T. Hens, and K. R. Schenk-Hoppe, Handbooks in Finance, pp. 57–160, 2009 (North-Holland: San Diego).

Bouchaud, J.-P., Gefen, Y., Potters, M. and Wyart, M., Fluctuations and response in financial markets: The subtle nature of 'random' price changes. *Quant. Finance*, 2004, **4**(2), 176–190.

Cartea, Á and Jaimungal, S., Incorporating order-flow into optimal execution. *Math. Financ. Econ.*, 2016, **10**(3), 339–364.

Chen, Y., Horst, U. and Tran, H.H., Portfolio liquidation under transient price impact – Theoretical solution and implementation with 100 NASDAQ stocks. Preprint, 2019. arXiv:1912.06426.

Curato, G., Gatheral, J. and Lillo, F., Optimal execution with nonlinear transient market impact. *Quant. Finance*, 2017, **17**(1), 41–54.

Dang, N.-M., Optimal execution with transient impact. *Mark. Microstruct. Liquidity*, 2017, **03**(01), 1750008.

Forde, M., Sánchez-Betancourt, L. and Smith, B., Optimal trade execution for Gaussian signals with power-law resilience. *Quant. Finance*, 2021, **0**(0), 1–12.

Fruth, A., Schöneborn, T. and Urusov, M., Optimal trade execution and price manipulation in order books with time-varying liquidity. *Math. Financ.*, 2014, **24**(4), 651–695.

Fruth, A., Schöneborn, T. and Urusov, M., Optimal trade execution in order books with stochastic liquidity. *Math. Financ.*, 2019, **29**(2), 507–541.

Gatheral, J., No-dynamic-arbitrage and market impact. *Quant. Finance*, 2010, **10**(7), 749–759.

Gatheral, J., Schied, A. and Slynko, A., Transient linear price impact and Fredholm integral equations. *Math. Financ.*, 2012, **22**(3), 445–474.

Germain, M., Pham, H. and Warin, X., Neural networks-based algorithms for stochastic control and PDEs in finance. Preprint, 2018. arXiv:2101.08068.

Graewe, P. and Horst, U., Optimal trade execution with instantaneous price impact and stochastic resilience. *SIAM J. Control Optim.*, 2017, **55**(6), 3707–3725.

Gramacy, R.B., *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*, 2020 (Chapman and Hall/CRC).

Hambly, B., Xu, R. and Yang, H., Recent advances in reinforcement learning in finance. Preprint, 2021. arXiv:2112.04553.

Han, J. and E, W., Deep learning approximation for stochastic control problems. Preprint, 2016. arXiv:1611.07422, Deep Reinforcement Learning Workshop, NIPS 2016.

Horst, U. and Xia, X., Multi-dimensional optimal trade execution under stochastic resilience. *Finance Stoch.*, 2019, **23**(4), 889–923.

Huberman, G. and Stanzl, W., Price manipulation and quasi-arbitrage. *Econometrica*, 2004, **72**(4), 1247–1275.

Huré, C., Pham, H., Bachouch, A. and Langrené, N., Deep neural networks algorithms for stochastic control problems on finite horizon, part i: Convergence analysis. Preprint, 2018. arXiv:1812.04300.

Ismail, A. and Pham, H., Robust Markowitz mean–variance portfolio selection under ambiguous covariance matrix. *Math Financ.*, 2019, **29**(1), 174–207.

Jaimungal, S., Reinforcement learning and stochastic optimisation. *Finance Stoch.*, 2022, **26**(1), 103–129.

Keshavarz, A. and Boyd, S., Quadratic approximate dynamic programming for input-affine systems. *Int. J. Robust Nonlinear Control.*, 2014, **24**(3), 432–449.

Leal, L., Laurière, M. and Lehalle, C.-A., Learning a functional control for high-frequency finance. Preprint, 2021. arXiv:2006.09611.

Lehalle, C.-A. and Neuman, E., Incorporating signals into optimal trading. *Finance Stoch.*, 2019, **23**(2), 275–311.

Lillo, F., Farmer, J.D. and Mantegna, R.N., Master curve for price-impact function. *Nature*, 2003, **421**(6919), 129–130.

Lorenz, C. and Schied, A., Drift dependence of optimal trade execution strategies under transient price impact. *Finance Stoch.*, 2013, **17**(4), 743–770.

Neuman, E. and Voß, M., Optimal signal-adaptive trading with temporary and transient price impact. *SIAM J. Financial Math.*, 2022, **13**(2), 551–575.

Obizhaeva, A.A. and Wang, J., Optimal trading strategy and supply/demand dynamics. *J. Financial Mark.*, 2013, **16**(1), 1–32.

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

Papanicolaou, A., Fu, H., Krishnamurthy, P., Healy, B. and Khorrami, F., An optimal control strategy for execution of large stock orders using long short-term memory networks. *J. Comput. Finance*, 2023, **26**(4), 37–65.

Predoiu, S., Shaikhet, G. and Shreve, S., Optimal execution in a general one-sided limit-order book. *SIAM J. Financial Math.*, 2011, **2**(1), 183–212.

Rasmussen, C.E. and Williams, C.K.I., *Gaussian Processes for Machine Learning*, 2006 (The MIT Press).

Schied, A., Schöneborn, T. and Tehranchi, M., Optimal basket liquidation for cara investors is deterministic. *Appl. Math. Finance*, 2010, **17**(6), 471–489.

Webster, K.T. *Handbook of Price Impact Modeling*, 2023 (Chapman and Hall/CRC).