Machine Learning and Hamilton-Jacobi-Bellman Equation for Optimal Decumulation: a Comparison Study

Marc Chen [a] Mohammad Shirazi [b] Peter A. Forsyth[c] Yuying Li [d]

June 29, 2023

Abstract

Without resorting to dynamic programming, we determine the decumulation strategy for the holder of a defined contribution (DC) pension plan. We formulate this as a constrained stochastic optimal control problem. Our approach is based on data-driven neural network (NN) optimization. Customized activation functions for the output layers of the NN are applied, which permits training via standard unconstrained optimization. The optimal solution yields a multi-period decumulation and asset allocation strategy, useful for a holder of a (DC) pension plan. The objective function of the optimal control problem is a weighted expected wealth withdrawn (EW) and expected shortfall (ES) that directly targets left-tail risk. The stochastic bound constraints enforce a guaranteed minimum withdrawal each year. We show that the proposed NN approach compares favorably with the numerical results from a Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) computational framework.

Keywords: Portfolio decumulation, neural network, stochastic optimal control
JEL codes: G11, G22
AMS codes: 93E20, 91G, 68T07, 65N06, 35Q93

1 Introduction

Access to traditional defined benefit (DB) pension plans continues to disappear for employees. In 2022, only 15% of private sector workers in the United States had access to a defined benefit plan, while 66% had access to a defined contribution (DC) plan [51]. In other countries, DB plans have become a thing of the past.

Defined contribution plans leave the burden of creating a withdrawal and allocation strategy to the individual investor, which Nobel Laureate William Sharpe referred to as "the nastiest, hardest problem in finance" [40]. Indeed, a review of the literature on decumulation strategies [3, 28] shows that balancing all of retirees' concerns with a single strategy is exceedingly difficult. To address these concerns and find an optimal balance between maximizing withdrawals and minimizing the

[a]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1, `marcandre.chen@uwaterloo.ca`

[b]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1, `mmkshirazi@uwaterloo.ca`

[c]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1, `paforsyt@uwaterloo.ca`

[d]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1, `yuying@uwaterloo.ca`

risk of depletion, while guaranteeing a minimum withdrawal, the approach in [17] determines a decumulation and allocation strategy for a standard 30-year investment horizon by formulating it as a constrained optimal stochastic control problem. Numerical solutions are obtained in [17] using dynamic programming, which results in a Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE).

The HJB PDE framework developed in [17] maximizes expected withdrawals and minimizes the risk of running out of savings, measured by the left-tail in the terminal wealth distribution. Maximizing withdrawals and minimizing risk are conflicting goals. Consequently, in order to determine Pareto optimal points, we will use a scalarization technique. A fixed lower bound is imposed on the withdrawal, providing a guaranteed income. An upper bound on withdrawal is also imposed, which can be viewed as the target withdrawal. The investment allocation is also constrained to prohibit shorting and leverage.

This constrained stochastic optimal control problem yields a dynamic stochastic strategy as a solution, which naturally aligns with retirees' concerns and objectives. Note that cash flows are not mortality weighted, consistent with [2]. This can be justified on the basis of *planning to live, not planning to die* as discussed in [35].

Our dynamic strategy can be contrasted to traditional strategies such as the *Bengen Rule* (4% Rule), which recommends withdrawing a constant 4% of initial capital each year (adjusted for inflation) and investing equal amounts into stocks and bonds [2]. Initially proposed in 1994, the 4% Rule is found in [43] to still be a popular strategy 14 years later, and remains as the near-universal recommendation of the top brokerage and retirement planning groups. Recently there has been acknowledgment in the asset management industry that the 4% Rule is sub-optimal, but wealth managers still recommend variations of the same constant withdrawal principle [55]. The strategy proposed by [17] is shown to be far more efficient than the Bengen 4% Rule. Unfortunately, the PDE solution in [17] is restricted to low dimensions (i.e. a small number of stochastic factors).

In order to remedy some of the deficiencies of PDE methods (such as in [17]), we propose a neural network (NN) based framework without using dynamic programming. In contrast to the PDE solution approach, our proposed NN approach has the following advantages:

(i) It is data-driven and does not depend on availability of a parametric model for traded assets. This makes the framework versatile in selecting training data, and less susceptible to model misspecification.

(ii) The control is learned directly by solving original multi-period optimal control problem and explicitly exploiting the low dimensionality of the control [52]. This technique thus avoids dynamic programming and the associated error propagation. The NN approach can also be applied to higher dimensional problems, such as those with a large number of assets.

(iii) The control generated from NN is a continuous function of time, which fits naturally if the optimal control has the same continuity property. If the optimal control is discontinuous in time [1], the NN appears capable of producing a smooth, but quite accurate, approximation.[2]

The NN generates an approximate solution to complicated stochastic optimal control problem. Consequently, it is imperative to assess accuracy and robustness. Rarely is the quality of an NN solution assessed rigorously, since an accurate solution to the optimal control problem is often not readily available. In this paper, we compare the NN solution to the decumulation problem against the ground-truth solutions from the provably convergent HJB PDE method.

[1] *Bang-bang* controls, frequently encountered in optimal control, are discontinuous as a function of time.

[2] For a possible explanation of this, see [22].

Although unusual, similar a comparison assessment exists in different applications, see, e.g., [26] for a comparison study on a fishing control problem. As machine learning and artificial intelligence based methods continue to proliferate in finance and investment management, it is crucial to demonstrate that these methods are reliable and explainable in the financial domain [7]. We believe that our proposed framework and test results make a step forward in demonstrating deep learning's potential for stochastic control problems in finance.

We summarize the main contributions of this paper are as follows:

- Proposing an NN framework with suitable activation functions for decumulation and allocation controls, which yields an approximate solution to the constrained stochastic optimal decumulation problem in [17] by solving a standard unconstrained optimization problem;
- Demonstrating that the NN solution achieves very high accuracy in terms of the efficient frontier and the decumulation control when compared to the solution from the provably convergent HJB PDE method;
- Illustrating that, with a suitably small regularization parameter, the NN allocation strategy can differ significantly from the PDE allocation strategy in the region of high wealth and near the end of withdrawal time horizon, while the relevant risk-reward statistics remain unaffected. This is due to the fact that the problem is ill-posed, with objective function insensitive to the control in these regions, unless we add a small regularization term;
- Testing the NN solution's robustness on out-of-sample and out-of-distribution data, as well as its versatility in using different datasets for training.

Our work differs from other NN methods for stochastic optimal problems in finance in that one NN is used for the discontinuous decumulation control while the second NN represents allocation control. As a NN solution to an optimal control problem in general, while other neural network and deep learning methods for optimal stochastic control problems have been proposed before, they differ significantly from our approach in architecture. These previous approaches take a *stacked* neural network approach as in [8, 19, 50] or a hybrid dynamic programming and reinforcement learning approach [21]. In contrast, our framework uses the same two neural networks at all rebalancing times in the investment scenario. Since our NNs take time as an input, the solution will be continuous in time if the control is continuous. Note that the idea of using time as an input to the NN was also suggested in [26]. According to the taxonomy of sequential decision problems proposed in [39], our approach would most closely be described as Policy Function Approximation (PFA).

Furthermore, with the exception of [26], previous papers do not provide a benchmark for numerical methods, as we do in this work. Our results show that our proposed NN method is able to approximate the numerical results in [17] with high accuracy. Especially notable, and somewhat unexpected, is that the *bang-bang* control[3] for the withdrawal is reproduced very closely with the NN method.

2 Problem Formulation

2.1 Overview

The investment scenario described in [17] concerns an investor with a portfolio wealth of a specified size, upon retirement. The investment horizon is fixed with a finite number of equally spaced

[3] In optimal stochastic control, a bang-bang control is a discontinuous function of the state.

rebalancing times (usually annually). At each rebalancing time, the investor first chooses how much to withdraw from the portfolio and then how to allocate the remaining wealth. The investor must withdraw an amount within a specified range. The wealth in this portfolio can be allocated to any mix of two given assets, with no shorting or leverage. The assets the investor can access are a broad stock index fund and a constant maturity bond index fund.

In the time that elapses between re-balancing times, the portfolio's wealth will change according to the dynamics of the underlying assets. If the wealth of the portfolio goes below zero (due to minimum withdrawals), no stock purchases are permitted. From this point on, debt will grow at the borrowing rate, and withdrawals are restricted to the minimum amount. At the end of the time horizon, a final withdrawal is made and the portfolio is liquidated, yielding the terminal wealth.

We assume here that the investor has other assets, such as real estate, which are non-fungible with investment assets. These other assets can be regarded as a hedge of last resort, which can be used to fund any accumulated debt [36]. This is not a novel assumption and is in line with the mental bucketing idea proposed by [45]. The use of this assumption within literature targeting similar problems is also common (see [18]). Of course, the objective of the optimal control is to make running out of savings an unlikely event.

The investor's goal then is to maximize the weighted sum of expected total withdrawals and the mean of the worst 5% of the outcomes (in terms of terminal wealth). We term this tail risk measure as Expected Shortfall (ES) at the 5% level. In this section, this optimization problem will be described with the mathematical details common to both the HJB and NN methods.

2.2 Stochastic Process Model

Let S_t and B_t represent the real (i.e. inflation-adjusted) *amounts* invested in the stock index and a constant maturity bond index, respectively. These assets are modeled with correlated jump diffusion models, in line with [29]. These parametric stochastic differential equations (SDEs) allow us to model non-normal asset returns. The SDEs are used in solving the HJB PDE, and generating training data with Monte Carlo (MC) simulations in the proposed NN framework. For the remainder of this paper, we refer to simulated data using these models as *synthetic* data.

If a jump is triggered, $S_t = \xi^s S_{t^-}$, where ξ^s is a jump multiplier and $S_{t^-} = S(t-\epsilon), \epsilon \to 0^+$ (S_{t^-} is the time immediately before t). $\log(\xi^s)$ is assumed to follow a double exponential distribution [24, 25]. The jump is either upward or downward, with probabilities u^s and $1-u^s$ respectively. Let $y = \log(\xi^s)$, and y has density

$$f^s(y) = u^s \eta_1^s e^{-\eta_1^s y} \mathbf{1}_{y \geq 0} + (1-u^s)\eta_2^s e^{\eta_2^s y} \mathbf{1}_{y<0} \ . \tag{2.1}$$

We also define

$$\gamma_\xi^s \ = \ E[\xi^s - 1] = \frac{u^s \eta_1^s}{\eta_1^s - 1} + \frac{(1-u^s)\eta_2^s}{\eta_2^s + 1} - 1 \ . \tag{2.2}$$

The starting point for building the jump diffusion model is a standard geometric Brownian motion, with drift rate μ^s and volatility σ^s. A third term is added to represent the effect of jumps, and a compensator is added to the drift term to preserve the expected drift rate. For stocks, this gives the following stochastic differential equation (SDE) that describes how the amount in the stock account S_t (inflation adjusted) evolves between rebalancing times:

$$\frac{dS_t}{S_{t^-}} = \left(\mu^s - \lambda_\xi^s \gamma_\xi^s\right) dt + \sigma^s \, dZ^s + d\left(\sum_{i=1}^{\pi_t^s} (\xi_i^s - 1)\right), \quad t \in (t_i, t_{i+1}) \tag{2.3}$$

where dZ^s is the increment of a Wiener process, π_t^s is a Poisson process with positive intensity parameter λ_ξ^s. For all i, ξ_i^s are assumed i.i.d, positive, and with distribution (2.1). In addition, it is assumed that ξ_i^s, π_t^s, and Z^s are all mutually independent.

In the practitioner literature, it is usual to model the returns of a constant maturity (real, i.e. inflation adjusted) bond index fund, by an SDE. Following the lead of [29, 27], we model the constant maturity (real) bond index by a jump diffusion process. Let the amount in the constant maturity bond index be $B_{t^-} = B(t - \epsilon), \epsilon \to 0^+$. Between rebalancing times, the amount in the bond account B_t evolves as

$$\frac{dB_t}{B_{t^-}} = \left(\mu^b - \lambda_\xi^b \gamma_\xi^b + \mu_c^b \mathbf{1}_{\{B_{t^-} < 0\}}\right) dt + \sigma^b \, dZ^b + d\left(\sum_{i=1}^{\pi_t^b} (\xi_i^b - 1)\right), \quad t \in (t_i, t_{i+1}) \tag{2.4}$$

where the corresponding terms in Equation (2.4) are defined in similar fashion to Equation (2.3). π_t^b denotes a Poisson process, having non-negative intensity parameter λ_ξ^b, $\gamma_\xi^b = E[\xi^b - 1]$, and $y = \log(\xi^b)$ has the same distribution as in equation (2.1) (denoted by $f^b(y)$) with distinct parameters, u^b, η_1^b, and η_2^b. We make the assumption that ξ_i^b, π_t^b, and Z^b mutually independent, similar to assumptions placed on the SDE for S_t. The term $\mu_c^b \mathbf{1}_{\{B_{t^-} < 0\}}$ represents the borrowing spread (assumed non-negative).

The correlation between the two assets' diffusion processes is ρ_{sb}, i.e., $dZ^s \cdot dZ^b = \rho_{sb} \, dt$. The jump processes are assumed to be independent. For further details concerning the justification of this market model, refer to [17].

The total amount in the retirement account at time t, W_t is given by

$$\text{Total wealth } \equiv W_t = S_t + B_t. \tag{2.5}$$

With the exception of an insolvency state, shorting stock and using leverage (i.e., borrowing) are not permitted, a realistic constraint in the context of DC retirement plans. Furthermore, if the wealth ever goes below zero, due to the guaranteed withdrawals, all stock holdings are sold. Debt then grows at the bond rate plus a borrowing spread. We emphasize that we are assuming that the retiree has other assets (i.e., residential real estate) which can be used to fund any accumulated debt. In practice, this could be done using a reverse mortgage [36].

2.3 Notational Conventions

Let $\mathcal{T}$ denote the set of discrete times at which rebalancing and withdrawals are permitted

$$\mathcal{T} = \{t_0 = 0 < t_1 < t_2 < \ldots < t_M = T\} \, . \tag{2.6}$$

The beginning of the investment period is $t_0 = 0$. We assume each rebalancing time is evenly spaced, meaning $t_i - t_{i-1} = \Delta t = T/M$ is constant. For notational simplicity, it will be convenient to denote time dependence in two forms, i.e. $S_t \equiv S(t), B_t \equiv B(t)$ and $W_t \equiv W(t)$. At each rebalancing time, $t_i \in \mathcal{T}$, we consider the following ordering of events. First, the investor withdraws an amount of cash q_i from the portfolio. Subsequently, the portfolio is then rebalanced. At time T, there is one final withdrawal, q_T, and then the portfolio is liquidated. We assume no taxes or transaction costs are incurred on rebalancing. The no-tax assumption is reasonable since retirement accounts are typically tax-advantaged. In addition, since trading is infrequent, we assume transaction costs to be negligible [11]. For any function $f(t)$, we denote

$$f(t_i^+) \equiv \lim_{\epsilon \to 0^+} f(t_i + \epsilon) \ , \qquad f(t_i^-) \equiv \lim_{\epsilon \to 0^+} f(t_i - \epsilon) \ . \tag{2.7}$$

Let $X(t) = (S(t), B(t))$, $t \in [0,T]$ denote the multidimensional controlled underlying process. Following typical notation, let $x = (s, b)$ denote the the realized state of the system.

At each rebalancing time t_i, the investor first withdraws the amount $q_i(\cdot)$, determined by the control at time t_i; that is, $q_i(\cdot) = q_i(X(t_i^-)) = q(X(t_i^-), t_i)$. This control is used to evolve the investment portfolio from W_t^- to W_t^+

$$W(t_i^+) = W(t_i^-) - q_i \ , \qquad t_i \in \mathcal{T} \ . \tag{2.8}$$

The withdrawal and allocation controls are formally functions of state before withdrawal, $X(t_i^-)$. However, it is useful to note that the allocation control is specifically a function of state after withdrawal. This is simply due to the fact that rebalancing occurs after the withdrawal. Let $p_i(\cdot)$ represent the fraction of wealth in stocks, after rebalancing

$$\begin{aligned} S(t_i^+) &= p(X(t_i^+), t_i) W(t_i^+) \\ B(t_i^+) &= (1 - p(X(t_i^+), t_i)) W(t_i^+) \ . \end{aligned} \tag{2.9}$$

As formulated, assuming no transaction costs, it is shown in [17] that the control depends on wealth only, i.e., $p_i(\cdot) = p(X(t_i^+), t_i) = p_i(W_i^+))$. Therefore, we make another notational adjustment for the sake of simplicity and consider $q_i(\cdot)$ to be a function of wealth before withdrawal, W_i^-, and $p_i(\cdot)$ to be a function of wealth after withdrawal, W_i^+.

We assume instantaneous rebalancing, with the implication that the control at time t_i is described by a pair $(q_i(\cdot), p_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i)$, where $\mathcal{Z}(W_i^-, W_i^+, t_i)$ represents the set of admissible control values for t_i. The constraints on the allocation control are no shorting, no leverage (except in an insolvent state). There are minimum and maximum values for the withdrawal. In the normal course of events, the no-shorting and no-leverage constraints imply that wealth is always positive. However, due to minimum withdrawals at rebalancing times, it is possible for insolvency to occur. In this case, no stock holdings are permitted, and debt accumulates at the borrowing rate. Any subsequent withdrawals are restricted to the minimum amounts. Any non-zero stock stock positions are liquidated at terminal time. We can mathematically state these constraints by imposing suitable bounds on the value of the controls as follows:

$$\mathcal{Z}_q(W_i^-, t_i) = \begin{cases} [q_{\min}, q_{\max}], & \text{if } t_i \in \mathcal{T}, \quad W_i^- > q_{\max} \\ [q_{\min}, W_i^-], & \text{if } t_i \in \mathcal{T}, \quad q_{\min} < W_i^- < q_{\max} \\ \{q_{\min}\}, & \text{if } t_i \in \mathcal{T}, \quad W_i^- < q_{\min} \end{cases} , \tag{2.10}$$

$$\mathcal{Z}_p(W_i^+, t_i) = \begin{cases} [0,1], & \text{if } W_i^+ > 0, \quad t_i \in \mathcal{T}, \quad t_i \neq t_M \\ \{0\}, & \text{if } W_i^+ \leq 0, \quad t_i \in \mathcal{T}, \quad t_i \neq t_M \\ \{0\}, & \text{if } t_i = t_M \end{cases} , \tag{2.11}$$

$$\mathcal{Z}(W_i^-, W_i^+, t_i) = \mathcal{Z}_q(W_i^-, t_i) \times \mathcal{Z}_p(W_i^+, t_i) \ . \tag{2.12}$$

At each t_i, we seek the optimal control for all possible combinations of $(S(t), B(t))$ having the same total wealth [17]. Hence, the controls for both withdrawal and allocation are formally a function of wealth and time before withdrawal (W_i^-,t_i), but for implementation purposes it will be helpful to write the allocation as a function of wealth and time after withdrawal (W_i^+,t_i). The admissible control set $\mathcal{A}$ can be written as

$$\mathcal{A} = \left\{ (q_i, p_i)_{0 \le i \le M} : (q_i, p_i) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \right\} . \tag{2.13}$$

An admissible control $\mathcal{P} \in \mathcal{A}$, can be written as

$$\mathcal{P} = \{ (q_i(\cdot), p_i(\cdot)) : \ (q_i(\cdot), p_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i), \ i = 0, \ldots, M \} . \tag{2.14}$$

It will sometimes be necessary to refer to the tail of the control sequence at $[t_n, t_{n+1}, \ldots, t_M]$, which we define as

$$\mathcal{P}_n = \{ (q_n(\cdot), p_n(\cdot)) \ldots, (p_M(\cdot), q_M(\cdot)) \} . \tag{2.15}$$

The essence of the problem, for both the HJB and NN methods outlined in this paper, will be to find an optimal control $\mathcal{P}^*$.

2.4 Risk: Expected Shortfall

Let $\mathcal{G}(W_T)$ be the probability density of terminal wealth W_T at $t = T$. For $0 < \alpha < 1$, typically $\alpha = 5\%$, let W'_α satisfy

$$\int_{-\infty}^{W'_\alpha} \mathcal{G}(W_T) \, dW_T = \alpha \ , \tag{2.16}$$

i.e., $Pr[W_T < W'_\alpha] = \alpha$. W'_α can be interpreted as the Value at risk (VAR) at the level α. We then define the Expected Shortfall (ES) as the mean of the worst α fraction of the terminal wealth. Mathematically,

$$\text{ES}_\alpha = \frac{\int_{-\infty}^{W'_\alpha} W_T \ \mathcal{G}(W_T) \ dW_T}{\alpha} . \tag{2.17}$$

As formulated, a higher ES is more desirable than a smaller ES (equation (2.17) is formulated in terms of final wealth not losses). For computational purposes, it is useful to use the definition of ES as devised in [41],

$$\text{ES}_\alpha \quad = \quad \sup_{W'} E \left[W' + \frac{1}{\alpha} \min(W_T - W', 0) \right] . \tag{2.18}$$

Under a control $\mathcal{P}$, and initial state X_0, this becomes:

$$\text{ES}_\alpha(X_0^-, t_0^-) \quad = \quad \sup_{W'} E_{\mathcal{P}}^{X_0^-, t_0^-} \left[W' + \frac{1}{\alpha} \min(W_T - W', 0) \right] . \tag{2.19}$$

The candidate values of W' can be taken from the set of possible values of W_T. It is important to note here that we define $\text{ES}_\alpha(X_0^-, t_0^-)$ which is the value of ES_α as seen at t_0^-. Hence, W' is fixed throughout the investment horizon. In fact, we are considering the induced time consistent strategy, as opposed to the time inconsistent version of an expected shortfall policy [47, 14]. This issue is addressed in more detail in Appendix A.

2.5 Reward Measure: Total Expected Withdrawals (EW)

As a measure of reward, we will use total expected withdrawals. Mathematically, total expected withdrawals (EW) is defined as

$$\text{EW}(X_0^-, t_0^-) = E_{\mathcal{P}}^{X_0^-, t_0^-} \left[\sum_{i=0}^{M} q_i \right] . \tag{2.20}$$

Remark 2.1 (No discounting, no mortality weighting). *Note that we do not discount the future cash flows in Equation (2.20). We remind the reader that all quantities are assumed real (i.e. inflation-adjusted), so that we are effectively assuming a real discount rate of zero, which is a conservative assumption. This is also consistent with the approach used in the classical work of [2]. In addition, we do not mortality weight the cash flows, which is also consistent with [2]. See [35] for a discussion of this approach (i.e.* plan to live, not plan to die*).*

2.6 Defining a Common Objective Function

In this section, we describe the common objective function used by both the HJB method and the NN method.

Since increasing Expected Withdrawals (EW) typically causes a simultaneous decrease in Expected Shortfall (ES), we determine Pareto optimal points for this multi-objective problem. For a given scalarization parameter κ, we seek the optimal control $\mathcal{P}_0$ such that the following is maximized,

$$\text{EW}(X_0^-, t_0^-) + \kappa \text{ES}_\alpha(X_0^-, t_0^-) . \tag{2.21}$$

We define (2.21) as the pre-commitment EW-ES problem ($PCEE_{t_0}(\kappa)$) and write the problem formally as

$$(PCEE_{t_0}(\kappa)):$$

$$J\left(s,b,t_0^-\right) = \sup_{\mathcal{P}_0 \in \mathcal{A}} \sup_{W'} \Bigg\{ E_{\mathcal{P}_0}^{X_0^-, t_0^-} \Bigg[\sum_{i=0}^{M} q_i \; + \; \kappa \Big(W' + \frac{1}{\alpha} \min(W_T - W', 0) \Big) \overbrace{+ \epsilon W_T}^{stabilization} \Bigg| X(t_0^-) = (s,b) \Bigg] \Bigg\}$$

$$\text{subject to} \begin{cases} (S_t, B_t) \text{ follow processes (2.3) and (2.4)}; \quad t \notin \mathcal{T} \\ W_i^+ = S_i^- + B_i^- - q_i \,, \; X_i^+ = (S_i^+, B_i^+) \\ S_i^+ = p_i(\cdot) W_i^+ \,, \; B_i^+ = (1 - p_i(\cdot)) W_i^+ \\ (q_i(\cdot), p_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \\ i = 0, \ldots, M, \quad t_i \in \mathcal{T} \end{cases} . \tag{2.22}$$

The ϵW_T stabilization term serves to avoid ill-posedness in the problem when $W_t \gg W'$, $t \to T$, and has little effect on optimal (ES, EW) or other summary statistics when $|\epsilon| \ll 1$. Further details about this stabilization term and its effects on both the HJB and NN framework will be discussed in Section 6. The objective function in (2.22) serves as the basis for the value function in the HJB framework and the loss function for the NN method.

Remark 2.2 (Induced time consistent policy)**.** *Note that a strategy based on ($PCEE_{t_0}(\kappa)$) is formally a pre-commitment strategy (i.e., time inconsistent). However, we will assume that the retiree actually follows the induced time consistent strategy [47, 14, 17], which is identical to the pre-commitment control at the initial time. See Appendix A for more discussion of this subtle point. Subsequently we will refer to the strategy from (2.22) as the EW-ES optimal control, noting that it is equivalent to an induced time consistent control at any time $t_i > t_0$.*

3 HJB Dynamic Programming Optimization Framework

The HJB framework uses dynamic programming, creating sub-problems from each time step in the problem and moving backward in time. For the convenience of the reader, we will summarize the algorithm in [17] here.

3.1 Deriving Auxiliary Function from $PCEE_{t_0}(\kappa)$

The HJB framework begins with defining auxiliary functions based on the objective function (2.22) and the underlying stochastic processes. An equivalent problem is then formulated, which will then be solved to find the optimal value function.

We begin by interchanging the $\sup_{\mathcal{P}_0}$ and $\sup_{W'}$ operators. This will serve as the starting point for the HJB solution

$$\begin{aligned} J\left(s,b,t_0^-\right) &= \sup_{W'} \sup_{\mathcal{P}_0 \in \mathcal{A}} \Bigg\{ E_{\mathcal{P}_0}^{X_0^-,t_0^-} \Bigg[\sum_{i=0}^{M} q_i \;+\; \kappa \bigg(W' + \frac{1}{\alpha} \min(W_T - W', 0) \bigg) \\ &\qquad + \epsilon W_T \bigg| X(t_0^-) = (s,b) \Bigg] \Bigg\} \quad . \end{aligned} \tag{3.1}$$

The auxiliary function which needs to be computed in the dynamic programming framework at each time t_n will have an associated strategy for any $t_n > 0$ that is equivalent with the solution of $PCEE_{t_0}(\kappa)$ for a fixed W'. For a full discussion of pre-commitment and time-consistent ES strategies, we refer the reader to [14], which also includes a proof with similar steps of how the following auxiliary function is derived from (3.1). Including W' in the state space gives us the expanded state space $\hat{X} = (s,b,W')$. Define the problem domain $\Omega = [0,\infty) \times (-\infty,+\infty) \times (-\infty,+\infty) \times [0,\infty)$. The auxiliary function $V(s,b,W',t) \in \Omega$ is then defined as,

$$\begin{aligned} V(s,b,W',t_n^-) &= \sup_{\mathcal{P}_n \in \mathcal{A}_n} \Bigg\{ E_{\mathcal{P}_n}^{\hat{X}_n^-,t_n^-} \Bigg[\sum_{i=n}^{M} q_i + \kappa \bigg(W' + \frac{1}{\alpha} \min((W_T - W'),0) \bigg) \\ &\qquad + \epsilon W_T \bigg| \hat{X}(t_n^-) = (s,b,W') \Bigg] \Bigg\} \\ &\text{subject to} \quad \begin{cases} (S_t, B_t) \text{ follow processes (2.3) and (2.4)}; \quad t \notin \mathcal{T} \\ W_i^+ = S_i^- + B_i^- - q_i \,, \;\; \hat{X}_i^+ = (S_i^+, B_i^+, W') \\ S_i^+ = p_i(\cdot) W_i^+ \,, \;\; B_i^+ = (1 - p_i(\cdot)) W_i^+ \\ (q_i(\cdot), p_i(\cdot)) \in \mathcal{Z}(W_i^-, W_i^+, t_i) \\ i = n, \ldots, M, \quad t_i \in \mathcal{T} \end{cases} \quad . \end{aligned} \tag{3.2}$$

3.2 Applying Dynamic Programming at Rebalancing Times

The principle of dynamic programming is applied at each $t_n \in \mathcal{T}$ on (3.2). As usual, the optimal control needs to be computed in reverse time order. We split the $\sup_{\mathcal{P}_n}$ operator into $\sup_{q\in\mathcal{Z}_q} \sup_{p\in\mathcal{Z}_p(w^- -q,t)}$.

$$\begin{aligned} V(s,b,W',t_n^-) &= \sup_{q\in\mathcal{Z}_q} \sup_{p\in\mathcal{Z}_p(w^- -q,t)} \left\{ q + \left[V((w^- - q)p, (w^- - q)(1-p), W', t_n^+) \right] \right\} \\ &= \sup_{q\in\mathcal{Z}_q} \left\{ q + \left[\sup_{p\in\mathcal{Z}_p(w^- -q,t)} V((w^- - q)p, (w^- - q)(1-p), W', t_n^+) \right] \right\} \\ & \quad w^- = s + b \ . \end{aligned} \tag{3.3}$$

Let $\overline{V}$ denote the upper semi-continuous envelope of V, which will have already been computed as the algorithm progresses backward through time. The optimal allocation $p_n(w,W')$ at time t_n is then given by

$$p_n(w,W') = \begin{cases} \operatorname*{arg\,max}_{p'\in[0,1]} \overline{V}(wp', w(1-p'), W', t_n^+), & w > 0, \quad t_n \neq t_M \\ 0, & w \leq 0 \quad \text{or } t_n = t_M \end{cases} . \tag{3.4}$$

Since we proceed backwards in time, the allocation control is determined first (in backwards time) followed by the withdrawal control q

$$q_n(w,W') = \operatorname*{arg\,max}_{q'\in\mathcal{Z}_q} \left\{ q' + \overline{V}((w-q')p_n(w-q',W'), (w-q')(1-p_n(w-q',W')), W', t_n^+) \right\} . \tag{3.5}$$

Using these controls for t_n, the solution then moves from from t_n^+ to t_n^-

$$\begin{aligned} V(s,b,W',t_n^-) &= q_n(w^-,W') + \overline{V}(w^+ p_n(w^+,W'), w^+(\ 1 - p_n(w^+,W')\ ), W', t_n^+) \\ & \quad w^- = s + b, \quad w^+ = s + b - q_n(w^-,W') \ . \end{aligned} \tag{3.6}$$

At $t = T$, we have the terminal condition

$$V(s,b,W',T^+) = \kappa \left(W' + \frac{\min((s+b-W'),0)}{\alpha} \right) . \tag{3.7}$$

3.3 Conditional Expectations between Rebalancing Times

For $t \in (t_{n-1},t_n)$, the tower property gives, for $0 < h < (t_n - t_{n-1})$,

$$V(s,b,W',t) = E\left[V(S(t+h), B(t+h), W', t+h) \middle| S(t) = s, B(t) = b \right] \ ; \ t \in (t_{n-1}, t_n - h) \ . \tag{3.8}$$

Assuming a parametric model of stock and bond SDEs, Ito's Lemma for jump processes [49] is first applied assume SDEs (2.3) and (2.4). The gives a partial integro differential equation (PIDE), as shown in [17] and Appendix B. In computational practice, the resulting PIDE is solved using Fourier methods discussed in [16].

3.4 Equivalence with $PCEE_{t_0}(\kappa)$

Proceeding backward in time, the auxiliary function $V(s,b,W',t_0^-)$ is determined at time zero. Problem $PCEE_{t_0}(\kappa)$ is then solved using a final optimization step

$$J(s,b,t_0^-) = \sup_{W'} V(s,b,W',t_0^-) \ . \tag{3.9}$$

Notice that $V(s,b,W',t_0^-)$ denotes the auxiliary function for the beginning of the investment period, and represents the last step (going backward) in solving the dynamic programming formulation. To obtain this, we begin with Equation (3.7) and recursively work backwards in time. In the final step (going backwards), interchanging $\sup_{W'} \sup_{\mathcal{P}}$ gives Equation (2.22).

This formulation (3.2-3.8) is equivalent to problem $PCEE_{t_0}(\kappa)$. For a summary of computational details, refer to Appendix C or see [17].

4 Neural Network Formulation

As an alternative to the HJB framework, we develop a neural network framework to solve the stochastic optimal control problem (2.22), which has the following characteristics:

(i) The NN framework is data driven, which does not require a parametric model for traded assets being specified. This avoids explicitly postulating parametric stochastic processes and the estimation of associated parameters. In addition, this allows us to add auxiliary market signals/variables (although we do not exploit this idea in this work).

(ii) The NN framework avoids the computation of high-dimensional conditional expectations by solving for the control at all times directly from a single standard unconstrained optimization, without dynamic programming (see [52] for a discussion of this). Since the control is low-dimensional, the approach avoids the *curse of dimensionality* by solving for the control directly, instead of via value iteration such as in the HJB dynamic programming method [52]. Such an approach also eliminates backward error propagation through rebalancing times.

(iii) If the optimal control is a continuous function of time and state, the NN control will naturally reflect this property. If the optimal control is discontinuous, NN representation produces a smooth approximation. While not required by the original problem formulation in (2.22), this continuity property likely leads to practical implementation benefits.

(iv) The NN method is further scalable and can be easily adapted to problems with longer horizons or higher rebalancing frequency without significantly increasing the computational complexity of the problem. This is in contrast to existing approaches using a stacked neural network approach [50].

We now formally describe the proposed NN framework and demonstrate the aforementioned properties. We approximate the control in $\mathcal{P}$ directly by using feed-forward, fully-connected neural networks. Given parameters $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$, i.e. NN weights and biases, $\hat{p}(W(t_i), t_i, \boldsymbol{\theta}_p)$ and $\hat{q}(W(t_i), t_i, \boldsymbol{\theta}_q)$ approximate the controls p_i and q_i respectively,

$$\begin{gathered}
\hat{q}(W_i^-, t_i^-, \boldsymbol{\theta}_q) \simeq q_i(W_i^-), \quad i = 0, \ldots, M \\
\hat{p}(W_i^+, t_i^+, \boldsymbol{\theta}_p) \simeq p_i(W_i^+), \quad i = 0, \ldots, M-1 \\
\hat{\mathcal{P}} = \{(\hat{q}(\cdot), \hat{p}(\cdot))\} \simeq \mathcal{P}
\end{gathered}$$

The functions $\hat{p}$ and $\hat{q}$ take time as one of the inputs, and therefore we can use just two NN functions to approximate control $\mathcal{P}$ across time instead of defining a NN at each rebalancing time. In this section, we discuss how we solve problem (2.22) using this approximation and then provide a description of the NN architecture that is used. We discuss the precise formulation used by the NN, including activation functions that encode the stochastic constraints.

4.1 Neural Network Optimization for $PCEE_{t_0}(\kappa)$

We begin by describing the NN optimization problem based on the stochastic optimal control problem (2.22). We first recall that, in the formulation in Section 3, controls q_i and p_i are functions of wealth only. Our goal is to choose NN weights $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}_q$ by solving (2.22), with $\hat{q}(W_i^-,t_i^-,\boldsymbol{\theta}_q)$ and $\hat{p}(W_i^+,t_i^+,\boldsymbol{\theta}_p)$ approximating feasible controls $(q_i,p_i) \in \mathcal{Z}(W_i^-,W_i^+,t_i)$ for $t_i \in \mathcal{T}$. For an arbitrary set of controls $\hat{\mathcal{P}}$ and wealth level W', we define the NN performance criteria V_{NN} as

$$
\begin{aligned}
V_{NN}(\hat{\mathcal{P}},W',s,b,t_0^-) \;\; &= \;\; E_{\hat{\mathcal{P}}_0}^{X_0^-,t_0^-}\Bigg[\sum_{i=0}^{M} \hat{q}_i \; + \; \kappa\Big(W' + \frac{1}{\alpha}\min(W_T - W',0)\Big) \\
&\qquad + \epsilon W_T \Big| X(t_0^-) = (s,b) \Bigg]
\end{aligned}
$$

$$
\text{subject to } \begin{cases} (S_t,B_t) \text{ follow processes (2.3) and (2.4)}, \quad t \notin \mathcal{T} \\ W_i^+ = S_i^- + B_i^- - q_i\,,\;\; X_i^+ = (S_i^+,B_i^+) \\ S_i^+ = \hat{p}_i(\cdot)W_i^+\,,\;\; B_i^+ = (1-\hat{p}_i(\cdot))W_i^+ \\ (\hat{q}_i(\cdot),\hat{p}_i(\cdot)) \in \mathcal{Z}(W_i^-,W_i^+,t_i) \\ i = 0,\ldots,M, \quad t_i \in \mathcal{T} \end{cases} \tag{4.1}
$$

The optimal value function J_{NN} (at t_0^-) is then given by

$$
J_{NN}(s,b,t_0^-) = \sup_{W'} \sup_{\hat{\mathcal{P}} \in \mathcal{A}} \; V_{NN}(\hat{\mathcal{P}},W',s,b,t_0^-) \;. \tag{4.2}
$$

Next we describe the structure of the neural networks and feasibility encoding.

4.2 Neural Network Framework

Consider two fully-connected feed-forward NNs, with $\hat{p}$ and $\hat{q}$ determined by parameter vectors $\boldsymbol{\theta}_p \in \mathbb{R}^{\nu_p}$ and $\boldsymbol{\theta}_q \in \mathbb{R}^{\nu_q}$, representing NN weights and biases respectively. The two NNs can differ in the choice of activation functions and in the number of hidden layers and nodes per layer. Each NN takes input of the same form $(W(t_i),t_i)$, but the withdrawal NN $\hat{q}$ takes the state variable observed before withdrawal, $(W(t_i^-),t_i)$, and the allocation NN $\hat{p}$ takes the state variable observed after withdrawal, $(W(t_i^+),t_i)$.

In order for the NN to generate a feasible control as specified in (4.4), we use a modified sigmoid activation function to scale the output from the withdrawal NN $\hat{q}$ according to the $PCEE_{t_0}(\kappa)$ problem's constraints on the withdrawal amount q_i, as given in Equation (2.10). This ultimately allows us to perform unconstrained optimization on the NN training parameters.

Specifically, assuming $x \in [0,1]$, the function $a + (b-a)x$ scales the output to $[a,b]$. We restrict withdrawal to $\hat{q}$ in $[q_{\min},q_{\max}]$. We note that this withdrawal range $q_{\max} - q_{\min}$ depends on wealth W^-, see from (2.10). Specifically, define the range of permitted withdrawal as follows,

$$\text{range} = \begin{cases} q_{\max} - q_{\min}, & \text{if } W_i^- > q_{\max} \\ W^- - q_{\min}, & \text{if } q_{\min} < W_i^- < q_{\max} \\ 0, & \text{if } W_i^- < q_{\min} \end{cases} .$$

More concisely, we have the following mathematical expression:

$$\text{range} = \max\left((\min(q_{\max}, W^-) - q_{\min}), 0\right) .$$

Let $z \in \mathbb{R}$ be the NN output before the final output layer of $\hat{q}$. Note that z depends on the input features, state and time, before being transformed by the output activation function. We then have the following expression for the withdrawal,

$$\begin{aligned} \hat{q}(W^-, t, \boldsymbol{\theta_q}) &= q_{\min} + \text{range} \cdot \left(\frac{1}{1+e^{-z}}\right) \\ &= q_{\min} + \max\left((\min(q_{\max}, W^-) - q_{\min}), 0\right) \left(\frac{1}{1+e^{-z}}\right) . \end{aligned}$$

Note that the sigmoid function $\frac{1}{1+e^{-z}}$ is a mapping from $\mathbb{R} \to [0,1]$.

Similarly, we use a softmax activation function on the NN output for $\hat{p}$, ensuring no-shorting and no-leverage constraints are automatically satisfied.

With these output activation functions, it can be easily verified that $(\hat{q}_i(\cdot), \hat{p}_i(\cdot)) \in \mathcal{Z}\left(W_i^-, W_i^+, t_i\right)$ always. Using the defined NN, this transforms the problem (4.2) of finding an optimal $\hat{\mathcal{P}}$ into the optimization problem:

$$\begin{aligned} \hat{J}_{NN}(s, b, t_0^-) &= \sup_{W' \in \mathbb{R}} \sup_{\boldsymbol{\theta_q} \in \mathbb{R}^{\nu_q}} \sup_{\boldsymbol{\theta_p} \in \mathbb{R}^{\nu_p}} \hat{V}_{NN}(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p, W', s, b, t_0^-) \\ &= \sup_{(W', \boldsymbol{\theta_q}, \boldsymbol{\theta_p}) \in \mathbb{R}^{\nu_q + \nu_p + 1}} \hat{V}_{NN}(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p, W', s, b, t_0^-) . \end{aligned} \tag{4.3}$$

It is worth noting here that, while the original control $\mathcal{P}$ is constrained in (2.13), the formulation (4.3) is an unconstrained optimization over $\boldsymbol{\theta}_q$, $\boldsymbol{\theta}_p$, and W'. Hence we can solve problem (4.3) directly using a gradient descent method. In the numerical experiments detailed in Sections 6 and 7, we use Adam stochastic gradient descent [23] to determine the optimal points $\boldsymbol{\theta}_{\boldsymbol{q}}^*$, $\boldsymbol{\theta}_{\boldsymbol{p}}^*$, and W'.

Note that the output of NN $\hat{q}$ yields the amount to withdraw, while the output of NN $\hat{p}$ produces asset allocation weights.

Figure 4.1 presents the proposed NN. We emphasize the following key aspects of this NN structure.

(i) Time is an *input* to both NNs in the framework. The parameter vectors $\boldsymbol{\theta}_q$ and $\boldsymbol{\theta}_p$ are constant and do not vary with time.

(ii) At each rebalancing time, the wealth observation before withdrawal is used to construct the feature vector for $\hat{q}$. The resulting withdrawal is then used to calculate wealth after withdrawal, which is an input feature for $\hat{p}$.

(iii) Standard sigmoid activation functions are used at each *hidden layer* output.

(iv) The output activation function for withdrawal is different from the activation function for allocation. Control $\hat{q}$ uses a modified sigmoid function, which is chosen to transform its output according to (2.10). Control $\hat{p}$ uses a softmax activation which ensures that its output gives only positive weights for each portfolio asset and the weights sum to one, as specified in (2.11). By constraining the NN output this way through proposed activation functions, we can use unconstrained optimization to train NN.

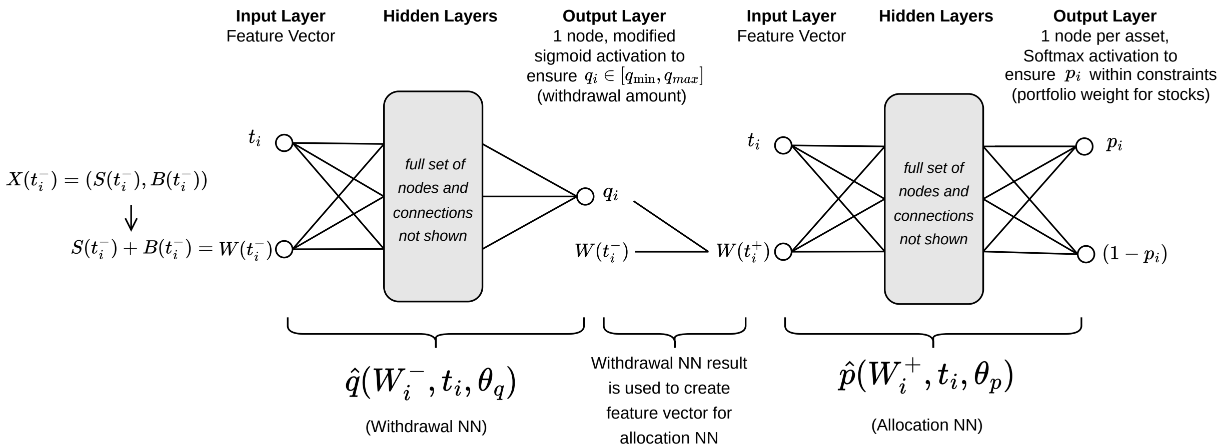


Figure 4.1: *Illustration of the NN framework as per Section 4.2. Additional technical details can be found in Appendix D.*

4.3 NN Estimate of the Optimal Control

Now we describe the NN training optimization problem for the decumulation problem, which is independent of the underlying data generation process. We assume that a set of asset return trajectories are available, which are used to approximate the expectation in (4.1) for any given control. For NN training, we approximate the expectation in (4.1) based on a finite number of samples as follows:

$$
\begin{aligned}
&\hat{V}_{NN}(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p, W', s, b, t_0^-) = \\
&\frac{1}{N} \sum_{j=1}^{N} \left[\sum_{i=0}^{M} \hat{q}((W_i)^j, t_i; \boldsymbol{\theta}_q) \; + \; \kappa \left(W' + \frac{1}{\alpha} \min((W_T)^j - W', 0) \right) + \epsilon (W_T)^j \middle| X(t_0^-) = (s, b) \right] \\
&\text{subject to} \begin{cases} ((S_t)^j, (B_t)^j) \text{ drawn from the } j^{th} \text{ sample of returns}; \quad t \notin \mathcal{T} \\ (W_i^+)^j = (S_i^-)^j + (B_i^-)^j - \hat{q}\left((W_{t_i}^-)^j, t_i, \boldsymbol{\theta_q}\right), \; (X_i^+)^j = (S_i^+, B_i^+)^j \\ (S_i^+)^j = \hat{p}\left((W_i^+)^j, t_i, \boldsymbol{\theta_p}\right) \; (W_i^+)^j, \; (B_i^+)^j = (1 - \hat{p}\left((W_i^+)^j, t_i, \boldsymbol{\theta_p}\right)) \; (W_i^+)^j \\ (\hat{q}_i(\cdot), \hat{p}_i(\cdot)) \in \mathcal{Z}\left((W_i^-)^j, (W_i^+)^j, t_i\right) \\ i = 0, \ldots, M, \quad t_i \in \mathcal{T} \end{cases} ,
\end{aligned}
\tag{4.4}
$$

where the superscript j represents the j^{th} path of joint asset returns and N is the total number of sampled paths. For subsequent benchmark comparison, we generate price paths using processes (2.3) and (2.4). However, any method can be used to generate these paths. We are not restricted to parametric SDEs. We assume that the random sample paths are independent, but that correlations can exist between returns of different assets. In addition, correlation between the returns of different time periods can also be represented, e.g., block bootstrap resampling is designed to capture autocorrelation in the time series data.

The optimal parameters obtained by training the neural network are used to generate the control functions $\hat{q}^*(\cdot) := \hat{q}(\cdot; \boldsymbol{\theta}_{\boldsymbol{q}}^*)$ and $\hat{p}^*(\cdot) := \hat{p}(\cdot; \boldsymbol{\theta}_{\boldsymbol{p}}^*)$, respectively. With these functions, we can evaluate the performance of the generated control on testing data sets that are out-of-sample or out-of-distribution. We present the detailed results of such tests in Section 7.

5 Data

For the computational study in this paper, we use data from the Center for Research in Security Prices (CRSP) on a monthly basis from 1926:1 to 2019:12.[4] The specific indices used are the CRSP 10-year U.S. Treasury index for the bond asset[5] and the CRSP cap-weighted total return index for the stock asset[6]. Retirees are, naturally, concerned with preserving real (not nominal) spending power. Hence, we use the US CPI index (from CSRP) to adjust these indexes for inflation. We use the above market data in two different ways in subsequent investigations:

(i) *Stochastic model calibration:* Any data set referred to in this paper as *synthetic data* is generated by parametric stochastic models (SDEs) (as described in Section 2.2), whose parameters are calibrated to the CRSP data using a threshold technique [30, 10, 12]. We divide the nominal CRSP data by the CPI as supplied by CRSP, so the the data is inflation adjusted. Calibration to the historical data generates the results in Table E.1. In order to compute the correlation ρ_{sb}, we first remove any returns which occur at jump times (in either series). See [12] for details of the technique for detecting jumps.

(ii) *Bootstrap resampling:* Any data set referred to in this paper as *historical data* is generated by using the stationary block bootstrap method [37, 38, 34, 13] to resample the historical CRSP data series. This method involves repeatedly drawing randomly sampled blocks of random size, with replacement, from the original data series. The block size follows a geometric distribution with a specified expected block size. We simultaneously draw returns from both series, in order to preserve correlation effects between asset returns. This, in effect, randomly shuffles the original data and can be repeated to obtain however many resampled paths one desires. Since the order of returns in the sequence is unchanged within the sampled block, this method accounts for some possible serial correlation in market data. Detailed pseudo-code for this method of block bootstrap resampling is given in [15].

We note that block resampling is commonly used by practitioners and academics (see for example [1, 13, 44, 46, 9]). Block bootstrap resampling will be used to carry out robustness checks in Section 7. Note that for any realistic number of samples and expected block size, the probability of repeating a resampled path is negligible [32].

[4]More specifically, results presented here were calculated based on data from Historical Indexes, ©2020 Center for Research in Security Prices (CRSP), The University of Chicago Booth School of Business. Wharton Research Data Services was used in preparing this article. This service and the data available thereon constitute valuable intellectual property and trade secrets of WRDS and/or its third-party suppliers.

[5]The 10-year Treasury index was calculated using monthly returns from CRSP dating back to 1941. The data for 1926-1941 were interpolated from annual returns in [20]. The bond index is constructed by (i) purchasing a 10-year Treasury at the start of each month, (ii) collecting interest during the month and (iii) selling the Treasury at the end of the month.

[6]The stock index includes all distributions for all domestic stocks trading on major U.S. exchanges.

One important parameter for the block resampling method is the expected block size. The algorithm in [34] is used to determine the optimal expected block size for the bond and stock returns separately; see Table F.1. For our data set here, a reasonable expected block size for paired resampling is about three months [17]. Subsequently, we will also test the sensitivity of the results to a range of block sizes from 1 to 12 months in numerical experiments.

To train the neural networks, we require that the number of sampled paths, N, be sufficiently large to fully represent the underlying market dynamics. Subsequently, we first generate training data through MC simulations of the calibrated parametric models in (2.3) and (2.4). We emphasize however that, in the proposed data driven NN framework, we only require return trajectories of the underlying assets. In later sections, we present results from NNs trained on non-parametrically generated data, e.g. resampled historical data. We also demonstrate the NN framework's robustness on test data.

6 Computational Results

We now present and compare performance of the optimal control from the HJB PDE and NN method respectively on synthetic data, with investment specifications given in Table 6.1. Each strategy's performance is measured w.r.t. to the objective function in (2.22), which is a weighted reward (EW) and risk (ES) measure. To trace out an efficient frontier in the (EW,ES) plane, we vary κ and the efficient frontier curve represents the (EW,ES) performance on a set of optimal Pareto points.

We first present strategies computed from the HJB framework described in Section 3. We verify that the numerical solutions are sufficiently accurate, which implies that this solution can be regarded as ground truth. We then present results computed using the NN framework of Section 4, and demonstrate the accuracy of the NN results by comparing to the ground truth computed from the HJB equation. We carry out further analysis by selecting an *interesting* point on the (EW,ES) efficient frontier, corresponding to $\kappa = 1.0$, to study in greater detail. The point $\kappa = 1.0$ is at the *knee* of the efficient frontier, which makes it desirable in terms of risk-reward tradeoff (picking the exact κ will be a matter of investor preference, however). This notion of the knee point is loosely based on the concept of a *compromise solution* of multi-objective optimization problems, which selects the point on the efficient frontier with the minimum distance to an unattainable ideal point [31]. For this knee point of $\kappa = 1.0$, we analyze the controls and wealth outcomes under both frameworks. We also discuss some key differences between the HJB and NN frameworks' results and their implications.

6.1 Strategies Computed from HJB Equation

We carry out a convergence test for the HJB framework by tracing the efficient frontier (i.e. varying the scalarization parameter κ) for solutions of varying refinement levels (i.e. number of grid points in the (s,b) directions). Figure 6.1 shows these efficient frontiers. As the efficient frontiers from various grid sizes all practically overlap each other, this demonstrates convergence of solutions computed from solving HJB equations. Table G.1 shows a convergence test for a single point on the frontier. The convergence is roughly first-order (for the value function). This convergence test justifies the use of the HJB framework results as a ground-truth.

Remark 6.1 (Effect of Stabilization Term ϵW_T)**.** *Recall the stabilization term, ϵW_T, introduced in (2.22), where the parameter ϵ has a small magnitude. We now provide motivation for its inclusion,*

Investment horizon T (years)	30
Equity market index	CPI adjusted CRSP US Total Market Index
Bond index	CPI adjusted US 10-year treasury
Initial portfolio value W_0	1000
Cash withdrawal times	$t = 0,1,\ldots,30$
Withdrawal range	$[35, 60]$
Equity fraction range	[0,1]
Borrowing spread μ_c^b	0.0
Rebalancing interval (years)	1
Market model parameters	See Appendix E

TABLE 6.1: *Problem setup and input data. Monetary units:* USD$ *in thousands.*

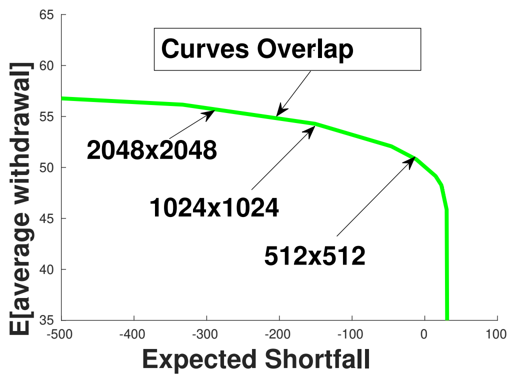


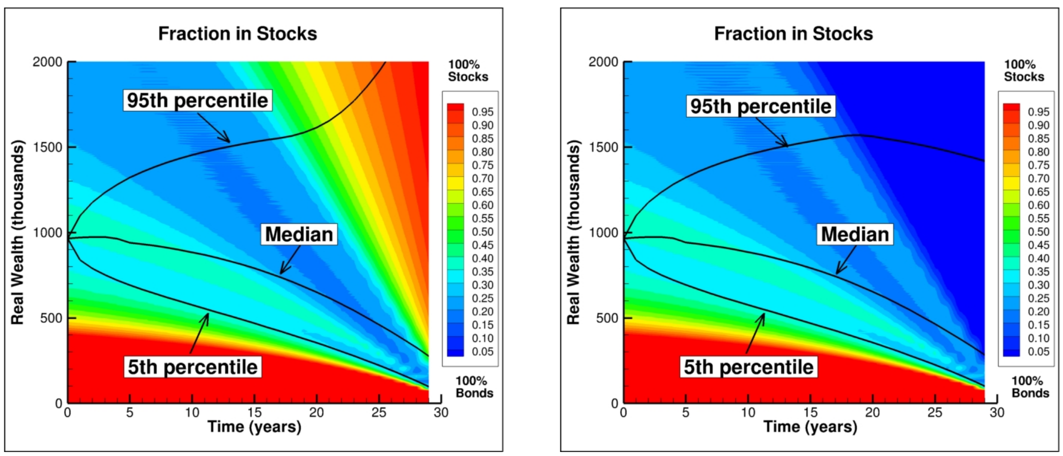
FIGURE 6.1: *EW-ES frontier, computed from problem (2.22). Note: Scenario in Table 6.1. Comparison of HJB solution performance with varying grid sizes. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury (see Table E.1). Minimum withdrawal: 35. Maximum withdrawal: 60.* $\epsilon = 10^{-6}$. *Monetary units:* USD$ *in thousands.*

and observe its effect on the control $\hat{\mathcal{P}}$. When $W_t \gg W'$ and $t \to T$, the objective function value is relatively insensitive to the change in control. This is because, in this situation, $Pr[W_T < W'] \simeq 0$ and thus the allocation control will have little effect on the ES term in the objective (recall that W' is held constant for the induced time consistent strategy, see Appendix A). In addition, the withdrawal is capped at $q_{\max}$, so the withdrawal control does not depend on W_t for very high value of W_t either. The stabilization term is used to alleviate this ill-posedness of the problem.

In Figure 6.2, we present the heat map of the allocation control computed from the HJB framework. Subplot (a) presents allocation control heat map for a small positive stabilization parameter $\epsilon = 10^{-6}$, while Subplot (b) presents allocation control heat map with $\epsilon = -10^{-6}$. In the ill-posed region (the top right region of the heat maps), the presence of ϵW_T, with $\epsilon = 10^{-6}$, forces the control to invest 100% in stocks to generate high terminal wealth. Conversely, changing the stabilization parameter to $\epsilon = -10^{-6}$ forces the control to invest completely in bonds.

We observe that the control behaves differently only at high level of wealth as $t \to T$ in both cases. The 5th and the 50th percentiles of control on the synthetic data set behave similarly both when ϵ is positive and when ϵ is negative. In contrast, the 95th percentile curves tend towards higher wealth during later phases of the investment period when the ϵ is positive (Figure 6.2(a)), whereas the curve tends downward when ϵ is negative (Figure 6.2(b)). When the magnitude of ϵ is sufficiently small, the inclusion of ϵW_T in the objective function does not change summary statistics (to four decimal places when $|\epsilon| = 10^{-6}$). While the choice in the sign of ϵ with small magnitude can lead to different allocation control scenarios at high wealth level near the end of time horizon, this choice makes little difference from the perspective of the problem $PCEE_{t_0}(\kappa)$. If the investor reaches very high wealth near T, the choice between 100% stocks and 100% bonds does not matter as the investor always ends with $W_T \gg W'$. Our experiments show that the control q is unaffected when the magnitude of ϵ is small and continues to call for maximum withdrawals at high wealth levels as $t \to T$, just as described in Remark 6.1.

Comparing the optimal withdrawal strategy from solving stochastic optimal control problem (2.22) with a fixed withdrawal strategy (both strategies with dynamic asset allocation), [17] finds that the stochastic optimal strategy (2.22) is much more efficient in balancing reward and risk. With a slight increase in risk, the retiree can expect to significantly increase total cash withdrawals. For a more detailed discussion of the optimal control, we refer the reader to [17].



(a) *Fraction in stocks, HJB Control,* $\epsilon = 10^{-6}$

(b) *Fraction in stocks, HJB Control,* $\epsilon = -10^{-6}$

FIGURE 6.2: *Effect of* ϵ*: fraction in stocks computed from the problem (2.22). Note: investment setup is as in Table 6.1. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on CPI adjusted CRSP US Total Market Index, CRSP US 10-year treasury (see Table E.1), Minimum withdrawal: 35. Maximum withdrawal: 60.* $\kappa = 1.0$. $W' = 58.0$ *for PIDE results. (a)* $\epsilon = 10^{-6}$*. (b)* $\epsilon = -10^{-6}$*. Monetary units:* USD$ *in thousands.*

6.2 Accuracy of NN Strategy

We compute the NN control as in Section 4. We compare the efficient frontiers obtained from the HJB equation solution and the NN solution. From Figure 6.3, the NN control efficient frontier is almost indistinguishable from the HJB control efficient frontier. In Appendix H.2, we also report numerical value for each computed point on the frontier. Objective function values corresponding to these points from the NN and HJB controls are presented in Appendix H.3. For most points on the frontier, the difference in objective function values, from NN and HJB, is less than 0.1%. This demonstrates that the accuracy of the NN framework approximation of the ground-truth solution is more than adequate, noting that the difference between the NN solution and the PDE solution is about the same as the estimated PDE error (see Table G.1).

We now further analyze the control $\hat{\mathcal{P}}$ produced by the NN framework for $\kappa = 1$. Comparing

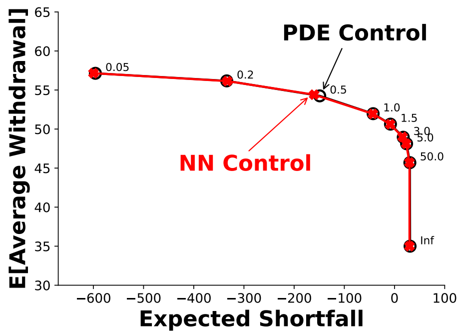


FIGURE 6.3: *Comparison of EW-ES frontier for the Neural Network (NN) and Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) methods, computed from the problem (2.22). Note: investment setup in Table 6.1. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury (see Table E.1). Control computed from the NN model, trained on* 2.56×10^6 *observations of synthetic data. Minimum withdrawal: 35. Maximum withdrawal: 60.* $\epsilon = 10^{-6}$. *Monetary units:* USD$ *in thousands. Labels on nodes indicate* κ *parameter.*

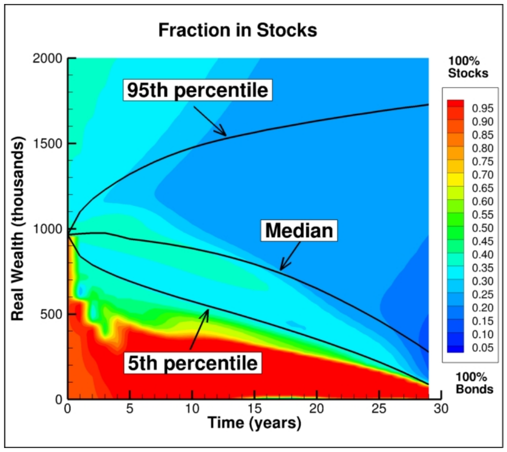
Figure 6.4(b) with Figure 6.4(d), we observe that the withdrawal control $\hat{q}$ produced by the NN is practically identical to the withdrawal control produced by the HJB framework. However, there are differences in the allocation control heat maps. The NN heat map for allocation control p (Figure 6.4(a)) appears most similar to that of the HJB allocation heat map for negative ϵ (Figure 6.2(b)), but it is clear that the NN allocation heat map differs significantly from the HJB heat map for positive ϵ (Figure 6.2(a)) at high level of wealth as $t \to T$. The NN allocation control behaves differently from the HJB controls in this region, choosing a mix of stocks and bonds instead of choosing a 100% allocation in a single asset. We emphasize that this difference is only at higher level of wealth near T and the 5th percentile and the median wealth curves are indistinguishable. The NN control's 95th percentile curve, however, is different. Indeed the NN 95th percentile curve is in between the 95th percentile curves from the negative and positive versions of the HJB-generated control.

Based on above observations, we attribute the NN framework's inability to fully replicate the HJB control to the ill-posedness of the optimal control problem, due to fact that the objective function is insensitive to the control in the (top-right) region of high wealth levels near T. The small value of ϵ means that the stabilization term contributes a very small fraction of the objective function value and thus has a very small gradient, relative to the first two terms in the objective function. Moreover, the data for high levels of wealth as $t \to T$ is very sparse. As a result, the NN appears to smoothly extrapolate in this region and therefore avoids investment into a single asset. Recall that in Section 6.1, we stated that the choice in the sign of ϵ, with small magnitude, in the stabilization term is somewhat arbitrary and does not affect summary statistics. Therefore, we see that the controls produced by the two methods only differ in irrelevant aspects, at least based on the EW and ES reward-risk consideration.

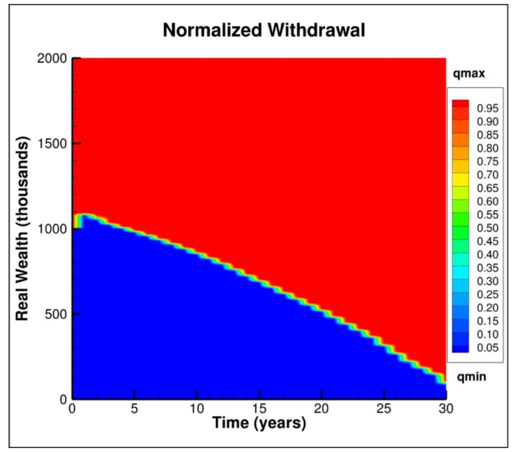
It is interesting to observe that the proposed neural network framework is able to produce the *bang-bang* withdrawal control computed in [17], especially since we are using the continuous function $\hat{q}$ as an approximation.[7] A *bang-bang* control switches abruptly as shown here: the optimal

[7]Note that [17] shows that that in the continuous withdrawal limit, the withdrawal control is bang-bang. Our computed HJB results show that for discrete rebalancing, the control appears to be bang-bang for all practical purposes.

NN Control Results

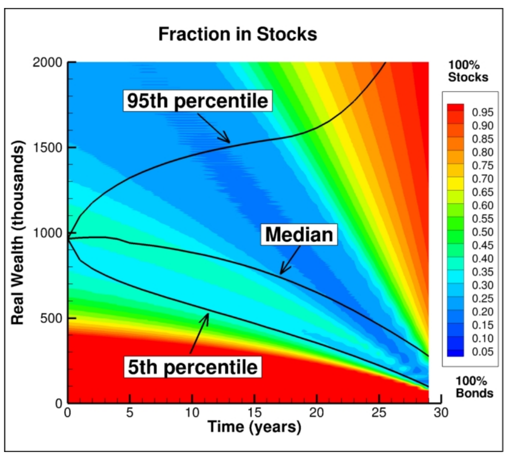


(a) *Fraction in stocks, NN Control*

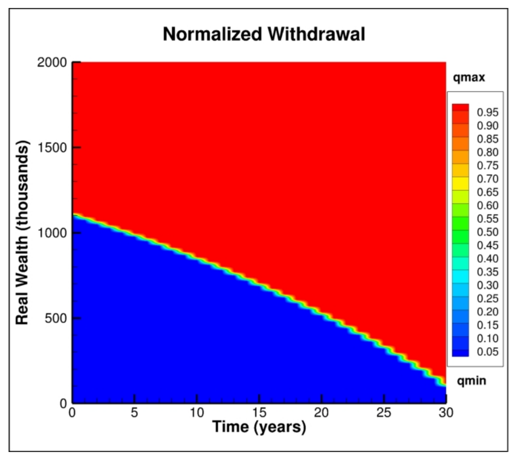


(b) *Withdrawals, NN Control*

HJB Control Results



(c) *Fraction in stocks, HJB Control*



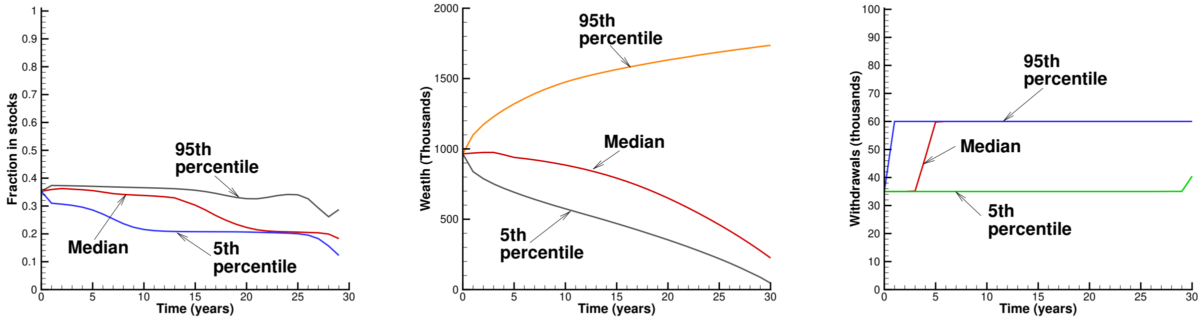
(d) *Withdrawals, HJB Control*

FIGURE 6.4: *Control heat map: withdrawals and fraction in stocks, determined by solving problem (2.22). Note: problem setup described in Table 6.1. HJB solution performance computed on* 2.56×10^6 *observations of synthetic data. Parameters for synthetic data based on CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury (see Table E.1). NN model trained on* 2.56×10^6 *observations of synthetic data. Minimum withdrawal: 35. Maximum withdrawal: 60.* $\kappa = 1.0$. $W' = 59.1$ *for NN results.* $W' = 58.0$ *for the HJB results.* $\epsilon = 10^{-6}$. *Relative withdrawal* $(q - q_{\text{min}})/(q_{\text{max}} - q_{\text{min}})$. *Monetary units: USD\$ in thousands*

strategy is to withdraw q_{min} if the wealth is less than a threshold, and withdraw q_{max} otherwise. As expected, the control threshold decreases as we move forward in time. We can see that the NN and HJB withdrawal controls behave very similarly at the 95th, 50th, and 5th percentiles of wealth (Figures 6.5(c) and 6.5(f)). Essentially, the optimal strategy withdraws at either q_{max} or q_{min}, with a very small transition zone. This is in line with our expectations. By withdrawing less and investing more initially, the individual decreases the chance of running out of savings.

We also note that the NN allocation control presents a small spread between the 5th and 95th percentile allocation control (Figure 6.5(a)). In fact, the maximum stock allocation for the 95th percentile never exceeds 40%, indicating that this is a stable low-risk strategy, which as we shall

NN Control Results

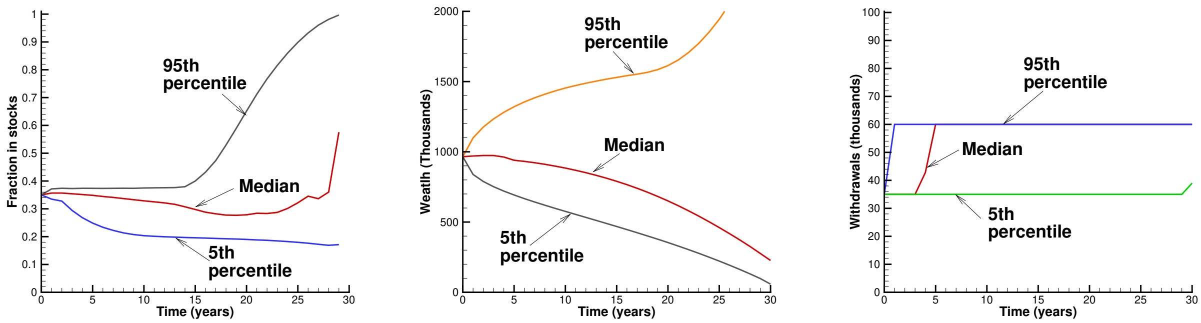


(a) *Stock allocation percentile, $\epsilon = 10^{-6}$, NN Control*

(b) *Wealth percentile, NN Control, $\epsilon = 10^{-6}$*

(c) *Withdrawal percentile, NN Control, $\epsilon = 10^{-6}$*

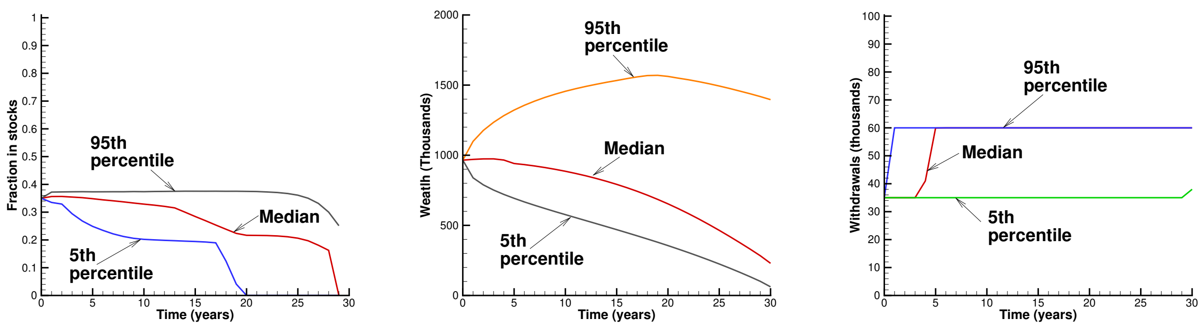
HJB Control Results (Positive and Negative Stabilization)



(d) *Stock allocation percentile, HJB Control, $\epsilon = 10^{-6}$*

(e) *Wealth percentile, HJB Control, $\epsilon = 10^{-6}$*

(f) *Withdrawal percentile, HJB Control, $\epsilon = 10^{-6}$*



(g) *Stock allocation percentile, HJB Control, $\epsilon = -10^{-6}$*

(h) *Wealth percentile, HJB Control, $\epsilon = -10^{-6}$*

(i) *Withdrawal percentile, HJB Control, $\epsilon = -10^{-6}$*

FIGURE 6.5: *Scenario in Table 6.1. NN and HJB controls computed from the problem (2.22). Parameters based on the inflation adjusted CRSP index, and the inflation adjusted 10-year treasuries (see Table E.1). NN model trained on 2.56×10^5 observations of synthetic data. HJB framework results from 2.56×10^6 observations of synthetic data. Minimum withdrawal: 35. Maximum withdrawal: 60. $\kappa = 1.0$. $W' = 59.1$ for NN results. $W' = 58.0$ for HJB results. Monetary units:* USD$ *in thousands.*

see, outperforms the Bengen strategy [2].

7 Model Robustness

A common potential pitfall of neural networks is over-fitting to the training data. Neural networks that are over-fitted do not have the ability to generalize to unseen data. Since future asset return paths cannot be predicted, it is important to ascertain that the computed strategy is not overfitted to the training data and can perform well on unseen return paths. In this section, we demonstrate the robustness of the NN model's generated controls.

We conduct three types of robustness tests: (i) out-of-sample testing, (ii) out-of-distribution testing, and (iii) control sensitivity to training distribution.

7.1 Out-of-sample testing

Out-of-sample tests involve testing model performance on an unseen data set sampled from the same distribution. In our case, this means training the NN on one set of SDE paths sampled from the parametric model, and testing on another set of paths generated using a different random seed. We present the efficient frontier generated by computed controls on this new data set in Figure 7.1, which shows almost unchanged performance on the out-of-sample test set.

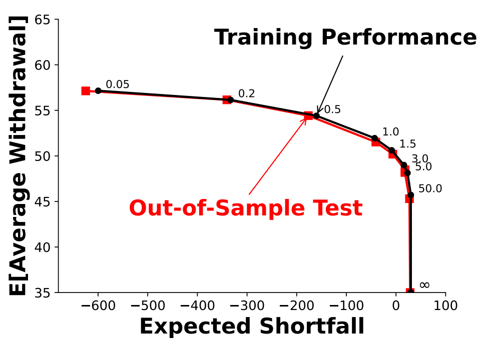


FIGURE 7.1: *Out-of-sample test. EW-ES frontiers, computed from the problem (2.22). Note: Scenario in Table 6.1. Comparison of NN training performance results vs. out-of-sample test. Both training and testing data are* 2.56×10^5 *observations of synthetic data, generated with a different random seed. Parameters for synthetic data based on CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury (see Table E.1). Minimum withdrawal: 35. Maximum withdrawal: 60.* $\epsilon = 10^{-6}$. *Monetary units:* USD$ *in thousands. Labels on nodes indicate* κ *parameter values.*

7.2 Out-of-distribution testing

Out-of-distribution testing involves evaluating the performance of the computed control on a data set sampled from a different distribution. Specifically, test data is not generated from the parametric model used to produce training data, but is instead bootstrap resampled from historical market returns via the method described in Section 5. We vary the expected block sizes to generate multiple testing data sets of 2.56×10^5 paths.

In Figure 7.2, we see that for each block size tested, the efficient frontiers are fairly close, indicating that the performance of control is relatively robust. Note that the efficient frontiers for test performance in the historical market with expected block size of 1 and 3 months plot slightly above the synthetic market frontier. We conjecture that this may be due to more pessimistic tail events in the synthetic market.

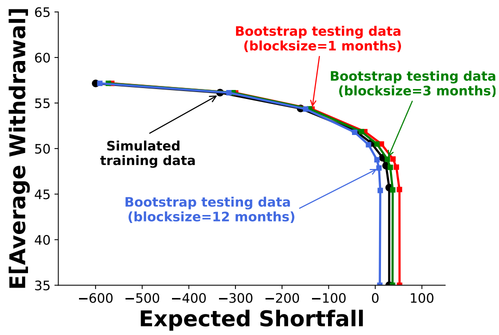


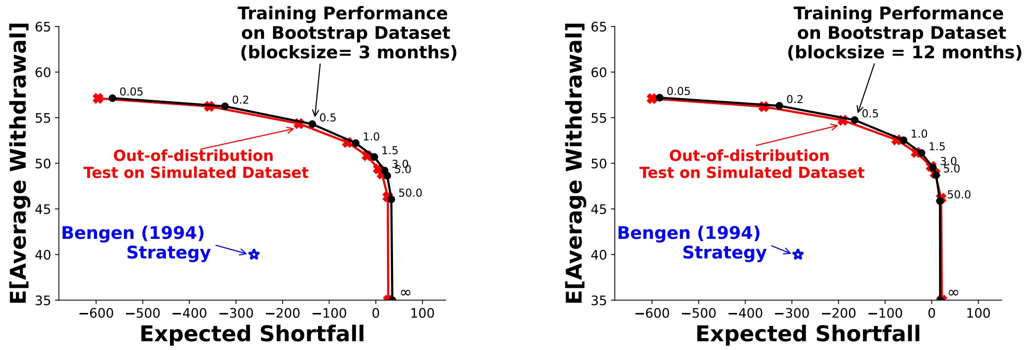
FIGURE 7.2: *Out-of-distribution test. EW-ES frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of synthetic data, tested on* 2.56×10^5 *observations of historical data with varying expected block sizes. Computed from the problem (2.22). Note: Setup as in Table 6.1. Parameters based on inflation adjusted CRSP index and inflation adjusted 10-year U.S. Treasuries (see Table E.1). Historical data between 1926:1 to 2019:12. Monetary units:* USD$ *in thousands. Minimum withdrawal: 35. Maximum withdrawal: 60. Simulated training data refers to MC simulations using the SDEs* (2.3) *and* (2.4).

The out-of-sample and out-of-distribution tests verify that the neural network is not over-fitting to the training data, and is generating an effective strategy, at least based on block resampling test data.

7.3 Sensitivity to training distribution

To test the NN framework's sensitivity to training data set, we train the NN framework on historical data (with expected block sizes of both 3 months and 12 months) and then test the resulting control on synthetic data. In Figure 7.3, we compare the training performance and the test performance. The EW-ES frontiers for the test results on the synthetic data are very close to the results on the bootstrap market data (training data set). This shows the NN framework's adaptability to use alternative data sets to learn, with the added advantage of not being reliant on a parametric model, which is prone to miscalibration. Figure 7.3 also shows that, the EW-ES control significantly outperforms the Bengen *4% Rule* [8] [2] in all cases, in the synthetic or historical market.

[8] The results for the Bengen strategy on the historical test data were computed with fixed withdrawals of 4% of initial capital, adjusted for inflation. We also used a constant allocation of 30% in stocks for expected block size of 3 months, and 35% in stocks for expected block size of 12 months. These were found to be the best performing constant allocations when paired with constant 4% real withdrawals, in terms of ES efficiency.



(a) *Historical training data, block size = 3 months*

(b) *Historical training data, block size = 12 months*

FIGURE 7.3: *Training on historical data. EW-ES frontiers of controls generated by NN model trained on* 2.56×10^5 *observations of historical data with expected block sizes of a) 3 months and b) 12 months, each tested on* 2.56×10^5 *observations of synthetic data. Parameters based on the inflation adjusted CRSP index and inflation adjusted 10-year U.S. Treasuries (see Table E.1). Historical data between 1926:1 to 2019:12. Monetary units:* USD$ *in thousands. Minimum withdrawal: 35. Maximum withdrawal: 60. Bengen point [2] is based on bootstrap resampling of the historical data. Labels on nodes indicate* κ *parameter values. Simulated testing data refers to MC simulations using the SDEs* (2.3) *and* (2.4). $\epsilon = +10^{-6}$.

8 Conclusion

In this paper, we put forward a neural network (NN) method to efficiently and accurately compute the optimal decumulation strategy for retirees with DC pension plans. This strategy is computed by directly solving a stochastically constrained optimal control problem based on a single standard unconstrained optimization, without using dynamic programming.

We began by highlighting the increasing prevalence of DC pension plans over traditional DB pension plans, and outlining the critical decumulation problem that faces DC plan investors. We examine a Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) based approach that can be shown to converge to an optimal solution for a dynamic withdrawal/allocation strategy. This provides an attractive balance of risk management and withdrawal efficiency for retirees. In this paper, we build upon this approach by developing a new, more versatile framework using NNs to solve the decumulation optimal control problem.

We conduct computational investigations to demonstrate the accuracy and robustness of the proposed NN solution, utilizing the unique opportunity to compare NN solutions with the HJB results as a ground truth. Of particular noteworthiness is that the continuous function approximation from the NN framework is able to approximate a bang-bang control with high accuracy. We extend our experiments to establish the robustness of our approach, testing the NN control's performance on both synthetic and historical data sets.

We demonstrate that the proposed NN framework produced solution accurately approximates the ground truth solution. We also note the following advantages of the proposed NN framework:

(i) The NN method is data driven, and does not require postulating and calibrating a parametric model for traded asset prices.

(ii) The NN method directly estimates the low dimensional control by solving a single unconstrained optimization problem, avoiding problems associated with dynamic programming methods, which require estimating high dimensional conditional expectations (see [52]).

(iii) The NN formulation maintains its simple structure (discussed in Section 4.2), immediately extendable to problems with more frequent rebalancing and/or withdrawal events. In fact, for the problem presented in (2.22), each control NN only requires two hidden layers for 30 rebalancing and withdrawal periods.

(iv) The approximated NN control maintains continuity in time and/or space, a natural choice if solution has this continuity property. Otherwise NN control provides a smooth approximation. Continuity in the allocation control p can be a practical implementation benefit.

Due to the ill-posedness of the stochastic optimal control problem in the region of high wealth near the end of the decumulation horizon, we observe that the NN allocation can appear to be very different from the HJB PDE solution. We note, however, that both strategies yield indistinguishable performance when assessed with the expected withdrawal and ES reward-risk criteria. In other words, these differences hardly affect the objective function value, which is a weighted reward and risk value. In the region of high wealth level near the end of the time horizon, the retiree is free to choose whether to invest 100% in stocks or 100% in bonds, since this has a negligible effect on the objective function value (or reward-risk consideration).[9]

To conclude, the NN solution framework provides a more versatile method, in comparison the HJB PDE approach. We expect that the NN approach can be readily extended to problems of higher complexity, e.g., involving a higher number of assets. In addition, the NN method can be applied to other proposed retirement planning problem formulations (for example, see [18]). We leave such extension to future work.

9 Acknowledgements

Both Forsyth and Li's work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), under grant numbers RGPIN-2017-03760 and RGPIN-2020-04331 respectively. The author's are grateful to P. van Staden for supplying the initial software library for NN control problems.

10 Conflicts of interest

The authors have no conflicts of interest to report.

Appendix

A Induced Time Consistent Policy

In this section of the appendix, we review the concept of time consistency and relate its relevance to the $PCEE_{t_0}(\kappa)$ problem, (2.22).

Consider the optimal control $\mathcal{P}^*$ for problem (2.22),

$$(\mathcal{P}^*)^{t_0}(X(t_i^-), t_i) \ ; \ i = 0, \ldots, M \ . \tag{A.1}$$

[9] This can be termed the *Warren Buffet* effect. Buffet is the fifth richest human being in the world. He is 92 years old. Buffet can choose any allocation strategy, and will never run out of cash.

Equation (A.1) can be interpreted as the optimal control for any time $t_i \geq t_0$, as a function of the state variables $X(t)$, as computed at t_0.

Now consider if we were to solve the problem (2.22) starting at a later time $t_k, k > 0$. This optimal control starting at t_k is denoted by:

$$(\mathcal{P}^*)^{t_k}(X(t_i^-), t_i) \; ; \; i = k, \ldots, M\} \; . \tag{A.2}$$

In general, the solution of (2.22) computed at t_k is not equivalent to the solution computed t_0:

$$(\mathcal{P}^*)^{t_k}(X(t_i^-), t_i) \neq (\mathcal{P}^*)^{t_0}(X(t_i^-), t_i) \; ; \; i \geq k > 0. \tag{A.3}$$

This non-equivalence makes problem (2.22) *time inconsistent*, implying that the investor will be motivated to diverge from the control determined at time t_0 at later times. This type of control is considered a *pre-commitment* control since the investor would need to commit to following the strategy at all times following t_0. Some authors describe pre-commitment controls as non-implementable because of the incentive to diverge from the initial control.

In our case, however, the pre-commitment control from (2.22) can be shown to be identical to the time consistent control for an alternative version of the objective function. By holding W' fixed at the optimal value (at time zero), we can define the time consistent equivalent problem (TCEQ). We define the optimal value of W' as [10]

$$\mathcal{W}^*(s,b) \;\; = \;\; \operatorname*{arg\,max}_{W\prime} \sup_{\mathcal{P}_0 \in \mathcal{A}} \Biggl\{ E_{\mathcal{P}_0}^{X_0^-, t_0^-} \Biggl[\; \sum_{i=0}^{M} q_i \; + \; \kappa \biggl(W' + \frac{1}{\alpha} \min(W_T - W', 0) \biggr) \bigg| X(t_0^-) = (s,b) \; \Biggr] \Biggr\} \; . \tag{A.4}$$

With a given initial wealth of W_0^-, this gives the following result from [14]:

Proposition A.1 (Pre-commitment strategy equivalence to a time consistent policy for an alternative objective function)**.** *The pre-commitment EW-ES strategy found by solving $J\left(s,b,t_0^-\right)$ from (2.22), with fixed $W' = \mathcal{W}^*$ from Equation A.4, is identical to the time consistent strategy for the equivalent problem TCEQ (which has fixed $\mathcal{W}^*(0,W_0^-)$), with the following value function:*

$$\begin{aligned} &(\, TCEQ_{t_n}\,(\kappa/\alpha)) : \\ &\tilde{J}\left(s,b,t_n^-\right) \;\; = \;\; \sup_{\mathcal{P}_n \in \mathcal{A}} \Biggl\{ E_{\mathcal{P}_n}^{X_n^-, t_n^-} \Biggl[\; \sum_{i=n}^{M} q_i \; + \; \frac{\kappa}{\alpha} \min(W_T - \mathcal{W}^*(0, W_0^-), 0) \bigg| X(t_n^-) = (s,b) \; \Biggr] \Biggr\}. \end{aligned} \tag{A.5}$$

Proof. This follows similar steps as in [14], proof of Proposition (6.2). □

With fixed W', $TCEQ_{t_n}(\kappa/\alpha)$ uses a target-based shortfall as its measure of risk, which is trivially time consistent. W' has the convenient interpretation of a disaster level of final wealth, as specified at time zero. Since the optimal controls for $PCEE_{t_0}(\kappa)$ and $TCEQ_{t_n}(\kappa/\alpha)$ are identical, we regard $TCEQ_{t_n}(\kappa/\alpha)$ as the induced time consistent strategy [47] for problem EW-ES. The retiree has no motivation to diverge from the induced time consistent strategy, determined at time zero. Hence this policy is implementable.

For more detailed analysis concerning the subtle distinctions involved in pre-commitment, time consistent, and induced time consistent strategies, please consult [5, 6, 53, 54, 47, 14, 4].

[10] The arg max is well defined since $\sup_{\mathcal{P}}\{\cdot\}$ is a continuous function of W'.

B PIDE Between Rebalancing Times

Applying Ito's Lemma for jump processes [49], using Equations (2.3) and (2.4) in Equation (3.8) gives

$$V_t + \frac{(\sigma^s)^2 s^2}{2} V_{ss} + (\mu^s - \lambda_\xi^s \gamma_\xi^s) s V_s + \lambda_\xi^s \int_{-\infty}^{+\infty} V(e^x s, b, t) f^s(x)\, dx + \frac{(\sigma^b)^2 b^2}{2} V_{bb}$$
$$+ (\mu^b + \mu_c^b \mathbf{1}_{\{b<0\}} - \lambda_\xi^b \gamma_\xi^b) b V_b + \lambda_\xi^b \int_{-\infty}^{+\infty} V(s, e^x b, t) f^b(x)\, dx - (\lambda_\xi^s + \lambda_\xi^b) V + \rho_{sb} \sigma^s \sigma^b s b V_{sb} = 0\ ,$$
$$s \geq 0\ . \tag{B.1}$$

where the density functions $f^s(x), f^b(x)$ are as given in equation (2.1).

C Computational Details: Hamilton-Jacobi-Bellman (HJB) PDE Framework

For a detailed description of the numerical algorithm used to solve the HJB equation framework described in Section 3, we refer the reader to [17]. We summarize the method here.

First, we solve the auxiliary problem (3.2), with fixed values of W', κ and α. The state space in $s > 0$ and $b > 0$ is discretized using evenly spaced nodes in log space to create a grid to represent cases. A separate grid is created in a similar fashion to represent cases where wealth is negative. The Fourier methods discussed in [16] are used to solve the PIDE representing market dynamics between rebalancing times. Both controls for withdrawal and allocation are discretized using equally spaced grids. The optimization problem (3.4) is solved first for the allocation control by exhaustive search, storing the optimal for each discretized wealth node. The withdrawal control in (3.5) can then be solved in a similar fashion, using the previously stored allocation control to evaluate the right-hand side of (3.5). Linear interpolation is used where necessary. The stored controls are used to advance the solution in (3.7).

Since the numerical method just described assumes a constant W', an outer optimization step to find the optimal W' (candidate Value-at-Risk) is necessary. Given an approximate solution to (3.2) at $t = 0$, the full solution to $PCEE_{t_0}(\kappa)$ (2.22) is determined using Equation (3.9). A coarse grid is used at first for an exhaustive search. The coarse grid solution is used as an initial guess for a univariate optimization technique on finer grids.

D Computational Details: NN Framework

D.1 NN Optimization

The NN framework, as described in Section 4 and illustrated in Figure 4.1, was implemented using the PyTorch library [33]. The withdrawal network $\hat{q}$, and allocation network $\hat{p}$ were both implemented with 2 hidden layers of 10 nodes each, with biases. Stochastic Gradient Descent [42] was used in conjunction with the Adaptive Momentum optimization algorithm to train the NN framework [23]. The NN parameters and auxiliary training parameter W' were trained with different initial learning rates. The same decay parameters and learning rate schedule were used. Weight decay (ℓ_2 penalty) was also employed to make training more stable. The training loop utilizes the auto-differentiation capabilities of the PyTorch library. Hyper-parameters used for NN training in this paper's experiments are given in Table D.1.

The training loop tracks the minimum loss function value as training progresses and selects the model that had given the optimal loss function value based on the entire training dataset by the end of the specified number of training epochs.

D.2 Transfer learning between different κ points

For high values of κ, the objective function is weighted more towards optimizing ES (lower risk). In these cases, optimal controls are more difficult to compute. This is because the ES measure used (CVAR) is only affected by the sample paths below the 5^{th} percentile of terminal wealth, which are quite sparse. To overcome these training difficulties, we employ transfer learning [48] to improve training for the more difficult points on the efficient frontier. We begin training the model for the lowest κ from a random initialization ('cold-start'), and then initialize the models for each increasing κ with the model for the previous κ. Through numerical experiments, we found this method made training far more stable and less likely to terminate in local minima for higher values of κ.

D.3 Running minimum tracking

The training loop tracks the minimum loss function value as training progresses and selects the model that had given the optimal loss function value based on the entire training dataset by the end of the specified number of training epochs.

NN framework hyper-parameter	Value
Hidden layers per network	2
# of nodes per hidden layer	10
Nodes have biases	True
# of iterations (#itn)	50,000
SGD mini-batch size	1,000
# of training paths	2.56×10^5
Optimizer	Adaptive Momentum
Initial Adam learning rate for $(\boldsymbol{\theta}_q, \boldsymbol{\theta}_p)$	0.05
Initial Adam learning rate for W'	0.04
Adam learning rate decay schedule	$[0.70 \times \#\text{itn}, 0.97 \times \#\text{itn}]$, $\gamma = 0.20$
Adam β_1	0.9
Adam β_2	0.998
Adam weight decay (ℓ_2 Penalty)	0.0001
Transfer Learning between κ points	True
Take running minimum as result	True

TABLE D.1: *Hyper-parameters used in training the NN framework for numerical experiments presented in this paper.*

D.4 Standardization

To improve learning for the neural network, we normalize the input wealth using means and standard deviations of wealth samples from a reference strategy. We use the constant withdrawal and allocation strategy defined in [17] as the reference strategy with 2.56×10^5 simulated paths. Let W_t^b denote the wealth vector at time t based on simulations. Then $\bar{W}_t^b$ and $\sigma(W_t^b)$ denote the associated average wealth and standard deviation. Then we normalize the feature input to the neural network in the following way:

$$\tilde{W}_t = \frac{W_t - \bar{W}_t^b}{\sigma(W_t^b)}$$

For the purpose of training the neural network, the values $\bar{W}_t^b$ and $\sigma(W_t^b)$ are just constants, and we can use any reasonable values. This input feature normalization is done for both withdrawal and allocation NNs.

In Section 7, we show in out-of-sample and out-of-distribution tests that $\bar{W}_t^b$ and $\sigma(W_t^b)$ do not need to be related to the testing data as long as these are reasonable values. In Section 4, when referring to W as part of the input to the NN functions $\hat{q}$ and $\hat{p}$, we use the standardized $\tilde{W}$ for computation.

E Model Calibrated from Market Data

Table E.1 shows the calibrated model parameters for processes (2.3) and (2.4), from [17] using market data described in §5.

Calibrated Model Parameters

CRSP	μ^s	σ^s	λ^s	u^s	η_1^s	η_2^s	ρ_{sb}
	0.0877	0.1459	0.3191	0.2333	4.3608	5.504	0.04554
10-year Treasury	μ^b	σ^b	λ^b	u^b	η_1^b	η_2^b	ρ_{sb}
	0.0239	0.0538	0.3830	0.6111	16.19	17.27	0.04554

TABLE E.1: *Calibrated (annualized) parameters for double exponential jump diffusion model. CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury, also inflation adjusted. Data from 1926:1 to 2019:12.*

F Optimal expected block sizes for block resampling

Table F.1 shows our estimates of the optimal block size using the algorithm in [38, 34] using market data described in §5.

Optimal expected block size for bootstrap resampling historical data

Data	Optimal expected block size $\hat{b}$ (months)
CRSP US 10-year treasury	4.2
CPI adjusted CRSP US Total Market Index	3.1

TABLE F.1: *Optimal expected blocksize $\hat{b} = 1/v$, from [34]. Range of historical data is between 1926:1 and 2019:12. The blocksize is a draw from a geometric distribution with $Pr(b = k) = (1 - v)^{k-1}v$.*

G Convergence Test: HJB Equation

Table G.1 shows a detailed convergence test for a single point on the (EW, ES) frontier, using the PIDE method. The controls are computed using the HJB PDE, and stored, which are then used in MC simulations. These results are used to verify the PDE solution, and also generate various statistics of interest.

HJB Method in §3					MC Simulation	
Grid $n_x \times n_b$	ES (5%)	$E[\sum_i q_i]/(M+1)$	Value Function	W'	ES (5%)	$E[\sum_i q_i]/(M+1)$
512×512	-51.302	52.056	1.562430e+3	50.10	-45.936	52.07
1024×1024	-46.239	52.049	1.567299e+3	52.47	-45.102	52.05
2048×2048	-42.594	51.976	1.568671e+3	58.00	-42.623	51.97
4096×4096	-40.879	51.932	1.569025e+3	61.08	-41.250	51.93

TABLE G.1: *HJB convergence analysis. CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury. Investment setup up in Table 6.1. Calibrated jump model in Table E.1.* 2.56×10^6 *MC simulations.* $\kappa = 1.0, \alpha = .05$. *Discretization grid in Section 3.* n_x: *# of nodes in* $\log s$. n_b : *# of nodes* $\log b$. *Monetary units:* USD$ *in thousands.* $(M+1)$: *# of withdrawals.* M : *# of rebalancing dates. Minimum withdrawal: 35. Maximum withdrawal: 60. HJB method in Section 3.*

H Detailed efficient frontier comparisons

Table H.1 shows the detailed efficient frontier, computed using the HJB equation method, using the 2048×2048 grid. Table H.2 shows the efficient frontier computed from the NN framework. This should be compared to Table H.1. Table H.3 compares the objective function values, at various points on the efficient frontier, for the HJB and NN frameworks.

Efficient Frontier Details: HJB Framework

κ	ES (5%)	$E[\sum_i q_i]/(M+1)$	$Median[W_T]$
0.05	-596.00	57.14	124.36
0.2	-334.29	56.17	92.99
0.5	-148.99	54.25	111.20
1.0	-42.62	51.97	227.84
1.5	-8.05	50.63	298.20
3.0	17.42	48.95	380.36
5.0	24.09	48.12	414.60
50.0	30.60	45.70	519.03
∞	31.00	35.00	1003.47

TABLE H.1: *Details of training performance efficient frontier in Figure 6.3 for HJB optimal strategies based on calibrated jump model. Investment setup in Table 6.1. CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury. Jump model parameters from Table E.1. Monetary units:* USD$ *in thousands.* 2.56×10^6 *MC simulations. Control is computed using HJB method in §3 with* (2048×2048 *grid) stored, subsequently used in MC simulations. Minimum withdrawal: 35. Maximum withdrawal: 60.* $(M+1)$: # *of withdrawals.* M : # *of rebalancing dates.* $\epsilon = 10^{-6}$.

Detailed Efficient Frontier: NN Framework

κ	ES (5%)	$E[\sum_i q_i]/(M+1)$	$Median[W_T]$
0.05	-599.81	57.15	106.23
0.2	-333.01	56.14	78.59
0.5	-160.14	54.40	105.05
1	-43.02	51.95	227.79
1.5	-8.57	50.62	302.17
3	16.01	48.99	374.43
5	23.20	48.13	425.13
50	29.88	45.72	493.41
∞	29.90	35.00	947.60

TABLE H.2: *Details of training performance efficient frontier in Figure 6.3 for NN optimal strategies based on calibrated jump model. Investment setup in Table 6.1. CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury. Jump model parameters from Table E.1. Monetary units:* USD$ *in thousands.* 2.56×10^5 *MC simulations. Control is computed using NN in Section 4. Minimum withdrawal: 35. Maximum withdrawal: 60.* $(M+1)$: # *of withdrawals.* M : # *of rebalancing dates.* $\epsilon = 10^{-6}$.

Objective Function Value Comparison: HJB Framework vs. NN Framework

κ	HJB equation	NN	% difference
0.05	1741.54	1741.71	0.01%
0.2	1674.41	1673.81	-0.04%
0.5	1607.26	1606.44	-0.05%
1	1568.45	1567.34	-0.07%
1.5	1557.46	1556.22	-0.08%
3	1569.71	1566.86	-0.18%
5	1612.16	1607.86	-0.27%
50	2946.70	2911.10	-1.21%

TABLE H.3: *Comparison in objective function values between HJB equation and NN framework model for various κ values. Objective function values for both frameworks computed according to $PCEE_{t0}(\kappa)$ (higher is better). Investment setup in Table 6.1. CPI adjusted CRSP US Total Market Index and CRSP US 10-year treasury. Jump model parameters from Table E.1. HJB solution statistics based on 2.56×10^6 MC simulations. HJB control is computed as in Section 3, (2048×2048 grid) stored, and then used in the MC simulations. NN Training performance statistics based on 2.56×10^5 MC simulations. Control is computed using the NN framework in Section 4. Minimum withdrawal: 35. Maximum withdrawal: 60. $(M + 1)$: # of withdrawals. M : # of rebalancing dates. $\epsilon = 10^{-6}$.*

References

[1] Aizhan Anarkulova, Scott Cederburg, and Michael S. O'Doherty. Stocks for the long run? evidence from a broad sample of developed markets. *Journal of Financial Economics*, 143(1):409–433, 2022.

[2] W. Bengen. Determining withdrawal rates using historical data. *Journal of Financial Planning*, 7:171–180, 1994.

[3] T. Bernhardt and C. Donnelly. Pension decumulation strategies: A state of the art report. Technical Report, Risk Insight Lab, Heriot Watt University, 2018.

[4] T. Bjork, M. Khapko, and A. Murgoci. *Time inconsistent control theory with finance applications*. Springer Finance, New York, 2021.

[5] T. Bjork and A. Murgoci. A general theory of Markovian time inconsistent stochastic control problems. SSRN 1694759, 2010.

[6] T. Bjork and A. Murgoci. A theory of Markovian time inconsisent stochastic control in discrete time. *Finance and Stochastics*, 18:545–592, 2014.

[7] El Bachir Boukherouaa, Khaled AlAjmi, Jose Deodoro, Aquiles Farias, and Rangachary Ravikumar. Powering the digital economy: Opportunities and risks of artificial intelligence in finance. *IMF Departmental Papers*, 2021(024):A001, 2021.

[8] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[9] Philippe Cogneau and Valeri Zakamouline. Block bootstrap methods and the choice of stocks for the long run. *Quantitative Finance*, 13:9:1443–1457, 2013.

[10] R. Cont and C. Mancini. Nonparametric tests for pathwise properties of semimartingales. *Bernoulli*, 17:781–813, 2011.

[11] D.-M. Dang and P. A. Forsyth. Continuous time mean-variance optimal portfolio allocation under jump diffusion: a numerical impulse control approach. *Numerical Methods for Partial Differential Equations*, 30:664–698, 2014.

[12] D-M. Dang and P. A. Forsyth. Better than pre-commitment mean-variance portfolio allocation strategies: a semi-self-financing Hamilton-Jacobi-Bellman equation approach. *European Journal of Operational Research*, 250:827–841, 2016.

[13] Hubert Dichtl, Wolfgang Drobetz, and Martin Wambach. Testing rebalancing strategies for stock-bond portfolios across different asset allocations. *Applied Economics*, 48(9):772–788, 2016.

[14] P. A. Forsyth. Multi-period mean CVAR asset allocation: Is it advantageous to be time consistent? *SIAM Journal on Financial Mathematics*, 11:2:358–384, 2020.

[15] P. A. Forsyth and K. R. Vetzal. Optimal asset allocation for retirement savings: deterministic vs. time consistent adaptive strategies. *Applied Mathematical Finance*, 26:1:1–37, 2019.

[16] P.A. Forsyth and G. Labahn. $\epsilon-$Monotone Fourier methods for optimal stochastic control in finance. *Journal of Computational Finance*, 22:4:25–71, 2019.

[17] Peter A. Forsyth. A stochastic control approach to defined contribution plan decumulation: The nastiest, hardest problem in finance. *North American Actuarial Journal*, 26:2:227–251, 2022.

[18] Peter A. Forsyth, Kenneth R. Vetzal, and G. Westmacott. Optimal performance of a tontine overlay subject to withdrawal constraints. *arXiv 2211.10509*, 2022.

[19] Jiequn Han and Weinan E. Deep learning approximation for stochastic control problems. *CoRR*, abs/1611.07422, 2016.

[20] S. Homer and R. Sylla. *A History of Interest Rates*. Wiley, New York, 2005.

[21] Cô me Huré, Huyên Pham, Achref Bachouch, and Nicolas Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon: Convergence analysis. *SIAM Journal on Numerical Analysis*, 59:1:525–557, 2021.

[22] Vugar Ismailov. A three layer neural network can represent any multivariate function. 2022. arXiv:2012.03016.

[23] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. page arXiv:1412.6980, 2014.

[24] S. G. Kou. A jump-diffusion model for option pricing. *Management Science*, 48:1086–1101, 2002.

[25] S. G. Kou and H. Wang. Option pricing under a double exponential jump diffusion model. *Management Science*, 50:1178–1192, 2004.

[26] Mathieu Laurière, Olivier Pironneau, et al. Performance of a markovian neural network versus dynamic programming on a fishing control problem. *arXiv preprint arXiv:2109.06856*, 2021.

[27] Y. Lin, R. MacMinn, and R. Tian. De-risking defined benefit plans. *Insurance: Mathematics and Economics*, 63:52–65, 2015.

[28] B.-J. MacDonald, B. Jones, R. J. Morrison, R. L. Brown, and M. Hardy. Research and reality: A literature review on drawing down retirement financial savings. *North American Actuarial Journal*, 17:181–215, 2013.

[29] R. MacMinn, P. Brockett, J. Wang, Y. Lin, and R. Tian. The securitization of longevity risk and its implications for retirement security. In O. S. Mitchell, R. Maurer, and P. Brett Hammond, editors, *Recreating Sustainable Retirement*, pages 134–160. Oxford University Press, Oxford, 2014.

[30] C. Mancini. Non-parametric threshold estimation models with stochastic diffusion coefficient and jumps. *Scandinavian Journal of Statistics*, 36:270–296, 2009.

[31] R. Marler and Jasbir Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 04 2004.

[32] Chendi Ni, Yuying Li, Peter Forsyth, and Ray Carroll. Optimal asset allocation for outperforming a stochastic benchmark target. *Quantitative Finance*, 22:9:1595–1626, 2022.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] A. Patton, D. Politis, and H. White. Correction to: automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 28:372–375, 2009.

[35] W. D. Pfau. An overview of retirement income planning. *Journal of Financial Counseling and Planning*, 29:1:114:120, 2018.

[36] S. Pfeiffer, J. R. Salter, and H. E. Evensky. Increasing the sustainable withdrawal rate using the standby reverse mortgage. *Journal of Financial Planning*, 26:12:55–62, 2013.

[37] D. Politis and J. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89:1303–1313, 1994.

[38] D. Politis and H. White. Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, 23:53–70, 2004.

[39] Warren B. Powell. From reinforcement learning to optimal control: A unified framework for s equential decisions. In Kyriakos G. Vamvoudakis, Yan Wan, Frank L. Lewis, and Derya Cansever, editors, *Handbook of Reinforcement Learning and Control*, pages 29–74. Springer International Publishing, 2021.

[40] B. Ritholz. Tackling the 'nastiest, hardest problem in finance'. `www.bloomberg.com/view/articles/2017-06-05/tackling-the-nastiest-hardest-problem-in-finance`, 2017.

[41] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[42] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

[43] Jason Scott, William Sharpe, and John Watson. The 4% rule – at what price? *Journal of Investment Management*, 7:3:31–48, 2009.

[44] Louis Scott and Stefano Cavaglia. A wealth management perspective on factor premia and the value of downside protection. *Journal of Portfolio Management*, 43(3):33–41, 2017.

[45] Hersh M. Shefrin and Richard H. Thaler. The behavioral life-cycle hypothesis. *Economic Inquiry*, 26:4:609–643, 1988.

[46] Joseph Simonian and Anna Martirosyan. Sharpe parity redux. *The Journal of Portfolio Management*, 48:4:183–193, 2022.

[47] M. Strub, D. Li, and X. Cui. An enhanced mean-variance framework for robo-advising applications. SSRN 3302111, 2019.

[48] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. *arXiv 1808.01974*, 2018.

[49] P. Tankov and R. Cont. *Financial Modelling with Jump Processes*. Chapman and Hall/CRC, New York, 2009.

[50] Ka Ho Tsang and Hoi Ying Wong. Deep-learning solution to portfolio selection with serially-dependent returns. *SIAM Journal on Financial Mathematics*, 11:2:593–619, 2020.

[51] U.S. Bureau of Labor Statistics. Employee benefits survey: Latest numbers, 2022. https://www.bls.gov/ebs/latest-numbers.htm.

[52] Pieter van Staden, Peter Forsyth, and Yuying Li. Beating a benchmark: dynamic programming may not be the right numerical approach. *SIAM Journal on Financial Mathematics*, 14:2:407–451, 2023.

[53] E. Vigna. On efficiency of mean-variance based portfolio selection in defined contribution pension schemes. *Quantitative Finance*, 14:237–258, 2014.

[54] E. Vigna. Tail optimality and preferences consistency for intertemporal optimization problems. *SIAM Journal on Financial Mathematics*, 13:1:295–320, 2022.

[55] Rob Williams and Chris Kawashima. Beyond the 4% rule: How much can you spend in retirement? 2023. https://www.schwab.com/learn/story/beyond-4-rule-how-much-can-you-spend-retirement.