Draft version July 14, 2025
Typeset using LaTeX default style in AASTeX631

Optimal Asset-Liability Management for Insurers

XU YUNPENG, Zhang Jinggong[1]

[1]*Nanyang Technological University, Singapore*

Abstract

Insurers' asset portfolios are crucial to company-growth and liability management. In previous works, under the framework of time-consistent planning, such portfolio optimization problems are usually done by assuming Markovian processes for asset movements, formulating the problem, finding a closed-form solution of the optimal control to the problem under simplistic assumptions, constructing a neural network approximation of the solution under realistic assumptions, and comparing the difference between the neural network output and the output of closed-form solution given some testing data. In this work, we assume the asset portfolio consists of stock and bonds, the liability consists of claim amounts, and they are modeled using VARIMA time series. Optimal asset allocation and dividend payout rates (optimal control laws) of an insurer will be found using a feedforward neural network surrogate, without a closed-form solution. We will then prove that the surrogate serves as a valid approximation of the optimal control law without empirically comparing it to a closed-form solution.

Keywords: Time-Consistent Planning, Dynamic Programming, Neural Network Surrogate

1. INTRODUCTION

Insurers need investment porfolio optimisations in order to manage solvency issues by increasing the portolio value, and to improve company-level P/L figures by distributing dividends. It is important for an insurer to be able to find the optimal portfolio trading strategy and dividend payout strategies.

In this work, a superscript with a bracket denotes the label, and a subscript denotes the time index. A lower-case term denotes a constant, unless been specified as a control variable, and an upper-case term denotes a random variable or process, unless otherwise specified. A lower-case bolded term denotes a vector, and an upper-case bolded term denotes a matrix. Vectors are assumed to be vertical. A subscript with a square bracket denotes the index of the elements in a vector or tuple. A left superscript denotes the index of the sample path.

For this research, We plan to find the optimal control for an insurer that maximizes the utility of periodic dividend payouts and the terminal portfolio value.

2. PROBLEM SETUP

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a discrete time filtered probability space with a finite discrete index set $N = \{0, 1, ..., T\}$ as time points when the controls are executed. Assume there exist m number of assets in the insurer's portfolio, in which one is government bond, and its price is denoted as $S_t^{(1)}$, and the market is complete and free of arbitrage. Let L_t be the dollar claim amount (the liability) to be paid at $t \in N$. Let $S_t^{(i)}$, $i = 2, 3, ..., m$ be the $m - 1$ number of risky assets' price at time t.

In this work, we assume all assets and the liability are non-negative, and follow a VARIMA(p, d, q) process with Gaussian noise, therefore, we model the log vector process using VARIMA like such

$$\phi(B) \cdot \Delta^d (\ln \boldsymbol{x}_t - \boldsymbol{\mu}_{\boldsymbol{x}}) = \psi(B) \cdot \boldsymbol{w}_t$$

where

$$\boldsymbol{x}_t := (S_t^{(1)}, S_t^{(2)}, \ldots, S_t^{(m)}, L_t)^\intercal, \boldsymbol{w}_t := (W_t^{(1)}, W_t^{(2)}, \ldots, W_t^{(m)}, W_t^{(L)})^\intercal \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}),$$

$$\Delta = 1 - B, \phi(B) = \mathbf{I} - \sum_{k=1}^{q} \boldsymbol{\Phi}^{(k)} \cdot B^k, \psi(B) = \mathbf{I} - \sum_{k=1}^{p} \boldsymbol{\Psi}^{(k)} \cdot B^k.$$

The matrices $\boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{\Sigma} \in \mathbb{R}^{(m+1)\times(m+1)}$ establish the dependence between bond, asset prices and claim amounts, and the white noise $\boldsymbol{w}_t$ introduces the randomness. As the result of the definition, we can find

$$\begin{aligned}\Delta \ln \boldsymbol{x}_{t+1} &= \sum_{k=1}^{d-1} \Delta^k \ln \boldsymbol{x}_t + \Delta^d \ln \boldsymbol{x}_{t+1} \\ &= \sum_{k=1}^{d-1} \Delta^k \ln \boldsymbol{x}_t + \boldsymbol{\mu}_{\boldsymbol{x}} + \sum_{k=1}^{p} \boldsymbol{\Phi}^{(k)} \cdot \big(\Delta^d \ln \boldsymbol{x}_{t-k+1} - \boldsymbol{\mu}_{\boldsymbol{x}}\big) + \sum_{k=1}^{q} \boldsymbol{\Psi}^{(k)} \cdot \boldsymbol{w}_{t-k+1} + \boldsymbol{w}_{t+1}\end{aligned}$$

and therefore

$$\boldsymbol{x}_{t+1} = \exp\Big(\ln \boldsymbol{x}_t + \Delta \ln \boldsymbol{x}_{t+1}\Big).$$

On the construction of the insurer's asset portfolio, let the units of stock i in the portfolio at t be $h_t^{(i)}$, and for now, let $h_t^{(i)}$ be the control. To be consistent with regulations, we assume no shorting is allowed, meaning $\forall i, h_t^{(i)} \geq 0$. In this work, we assume the portfolio value is a controlled process with the same index set N as $\boldsymbol{x}$, and the portfolio value at any time point is the total price of assets evaluated at the previous number of holdings, deducted by the claim occured at that time point. Thus, we define the portfolio value C_t at time $t = 1, 2, ..., T$ as below

$$C_t = \sum_{i=1}^{m} h_{t-1}^{(i)} \cdot S_t^{(i)} - L_t.$$

Let the dividend payout rate be $u_t^{(0)}$, and the insurer requires it to be constrained within $[\underline{u}^{(0)}, \overline{u}^{(0)}] \subseteq [0,1]$. Let the constant premium rate received at every time t to be π. When making the decision of the control $h_t^{(i)}$ at every t, the following constraint must be satisfied

$$\sum_{i=1}^{m} h_t^{(i)} \cdot S_t^{(i)} = C_t + \pi - u_t^{(0)} \cdot C_t,$$

that is, the total updated value of holdings must equal the current portfolio value, with the premium added and dividend deducted.

For the ease of calculation, we define the proportion of portfolio value of each stock in the portfolio as

$$u_t^{(i)} := \frac{h_t^{(i)} \cdot S_t^{(i)}}{C_t + p - u_t^{(0)} \cdot C_t}, i = 1, 2, ..., m,$$

and the work thereafter treats the above dividend payout rate and proportion $\mathbf{u}_t = (u_t^{(0)}, ..., u_t^{(m-1)})^\intercal$ as the control variables. Let $\underline{\mathbf{u}} = (\underline{u}^{(0)}, \ldots, \underline{u}^{(m-1)})^\intercal, \overline{\mathbf{u}} = (\overline{u}^{(0)}, \ldots, \overline{u}^{(m-1)})^\intercal$, we define the set of admissible strategy as below

$$\mathcal{U}_t := \Big\{ (\mathbf{u}_t)_{t=0,\ldots,T-1} : \forall t \in N, \mathbf{u}_t \in [\underline{\mathbf{u}}, \overline{\mathbf{u}}] \Big\}.$$

Since $\sum_{i=1}^{m} u_t^{(i)} = 1$ from the definition of $u_t^{(i)}$, $u_t^{(m)} = 1 - \sum_{i=1}^{m-1} u_t^{(i)}$, that is why the control is of dimension m.

Deducting the claims occured at $t+1$, L_{t+1}, we are led to the following dynamics of portfolio value

$$C_{t+1} = \sum_{i=1}^{m} u_t^{(i)} \cdot R_{t+1}^{(i)} \cdot (C_t + \pi - u_t^{(0)} \cdot C_t) - L_{t+1}$$

where $R_{t+1}^{(i)} = \exp(\Delta \ln \boldsymbol{x}_{t+1})_{[i]}$ is the return rate.

3. VALUE FUNCTION

The aim of the insurer is to maximise the utility from periodic dividend payouts as well as discounted terminal portfolio value at the discount rate of β. Assuming the insurer faces the utility function of the below form

$$U(\cdot) = \frac{1}{\gamma}(\cdot)^{\gamma}, 0 < \gamma < 1,$$

the second derivative of U is negative and therefore exhibits a risk-averse appetite toward dividend and terminal portfolio value.

Define $\mathbb{Y} \subseteq \mathbb{R}^{(m+1)\cdot(d-1+p+q)+2}$ as the state space of $\boldsymbol{y}_t$, where $\boldsymbol{y}_t$ is the stochastic state variable, and can be realized as $\boldsymbol{y}$ as below,

$$\boldsymbol{y} := (B^0\Delta^d \ln \boldsymbol{x}^\intercal, \dots, B^{p-1}\Delta^d \ln \boldsymbol{x}^\intercal, B^0\boldsymbol{w}^\intercal, \dots, B^{q-1}\boldsymbol{w}^\intercal, \Delta^{d-1}\ln \boldsymbol{x}^\intercal, \dots, \Delta^1 \ln \boldsymbol{x}^\intercal, \ln L, C)^\intercal.$$

Assume the insurer adopts constant risk discount rate β, the Markov decision problem that the insurer wishes to maximize at each t is formulated as below

$$\mathbb{E}\left(\sum_{s=t}^{T-1} U(u_s^{(0)} \cdot C_s) + \beta^{T-t} \cdot U(C_T) \middle| \boldsymbol{Y}_t = \boldsymbol{y}\right), t \in N.$$

It follows that the value function is of the below form

$$V_t(\boldsymbol{y}) = \sup_{\forall (\mathbf{u}_s)_{s=t,\dots,T-1} \in \mathcal{U}_t} \mathbb{E}\left(\sum_{s=t}^{T-1} U(u_s^{(0)} \cdot C_s) + \beta^{T-t} \cdot U(C_T) \middle| \boldsymbol{Y}_t = \boldsymbol{y}\right), t \in N.$$

Using the terminal condition at $t = T$, we have

$$V_T(\boldsymbol{y}) = \frac{1}{\gamma} \cdot (C)^\gamma,$$

therefore by constructing the dynamic programming equation, we have

$$V_t(\boldsymbol{y}) = \sup_{\forall \mathbf{u}_t \in \mathcal{U}_t} \left\{ \frac{1}{\gamma} \cdot (u_t^{(0)} \cdot C)^\gamma + \beta \cdot \mathbb{E}\left(V_{t+1}\big(\sum_{i=1}^m u_t^{(i)} \cdot B^{-1}R^{(i)} \cdot (C + \pi - u_t^{(0)} \cdot C) - \exp(B^{-1}\ln L)\big) \middle| \boldsymbol{Y}_t = \boldsymbol{y}\right)\right\}, t \leq T-1 \quad (1)$$

where

$$B^{-1}R^{(i)} = \exp(\Delta \ln B^{-1}\boldsymbol{x})_{[i]} \quad (2)$$

$$= \exp\Big(\sum_{k=1}^{d-1} \Delta^k \ln \boldsymbol{x} + \boldsymbol{\mu}_{\boldsymbol{x}} + \sum_{k=1}^{p} \boldsymbol{\Phi}^{(k)} \cdot \big(\Delta^d \ln B^{k-1}\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{x}}\big) + \sum_{k=1}^{q} \boldsymbol{\Psi}^{(k)} \cdot B^{k-1}\boldsymbol{w} + B^{-1}\boldsymbol{w}\Big)_{[i]}, \quad (3)$$

$$B^{-1}\ln L = \ln L + \Delta \ln B^{-1}\boldsymbol{x}_{[m+1]} \quad (4)$$

with the forward shifted white noise $B^{-1}\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$.

Let $\hat{\mathbf{u}}_t = \arg\max_{\forall \mathbf{u}_t \in \mathcal{U}_t}\{V_t(\boldsymbol{y})\}$, one can tell it is extremely time consuming to find the closed-form solution of $\hat{\mathbf{u}}_t$ and V_t. And we are led to the neural network surrogate of them.

4. SURROGATE

In this section, we approximate the solution to the dynamic programming problem using a neural network surrogate. The surrogation adopts the idea of [Tao Chen, Mike Ludkovski, and Moritz Voß (19 Dec 2023): On parametric optimal execution and machine learning surrogates, Quantitative Finance, DOI: 10.1080/14697688.2023.2282657]. The goal is to construct a sequence of neural networks $\{\mathring{V}^{\hat{\theta}_t} : \mathbb{Y} \to \mathbb{R}^{1+m}\}_{t=1}^{T-1}$ such that each network approximates the value function $V_t(\boldsymbol{y})$ at time t, and outputs the optimal control vector $\mathbf{u}_t \in \mathcal{U}_t$ as auxiliary outputs. The overall procedure mimics backward dynamic programming through a data-driven approach. With the approximated mapping $\mathring{V}^{\hat{\theta}_t}$, for any sample path of $\boldsymbol{y}_t = \boldsymbol{y}$, one standing at time t can replace the true V_{t+1} in Equation 1 by the approximated mapping of it, input $\boldsymbol{y}$, and use an optimizer of Equation 1 to find the value of the optimal controls. Below is a description of the pipeline:

1. **Training Data for Predictors.** A VARIMA model is fitted to the historical data of $\ln \boldsymbol{x}$. Using the fitted model, we simulate multiple sample paths of $\boldsymbol{y}$ forward T time steps. From these simulations, we estimate the bounds for each component of $\boldsymbol{y}$, which are then used to generate training data for predictors.

2. **One-Step Transition Function.** A one-step transition function $g(\boldsymbol{y}, \mathbf{u} \mid \tilde{\boldsymbol{w}})$ is defined to shift $\boldsymbol{y}$ forward by one time step with a control $\mathbf{u}$, given a realization $\tilde{\boldsymbol{w}}$ of the noise.

3. **Target Generation via Quantized Dynamic Programming.** Assuming the surrogate for V_{t+1} is already trained, we approximate the expectation in the Bellman equation 1 represented using g at time t using quantization of the distribution of the noise. Substituting each training data of $\boldsymbol{y}$ into this approximation and optimizing over $\mathbf{u}$, we compute the numerical value function for each training data.

4. **Neural Network Design and Two-Stage Optimization.** To ensure convergence and improve generalization, we adopt a partially concave neural network architecture. For each training data of $\boldsymbol{y}$, we solve a two-stage optimization problem to determine the control $\mathbf{u}_t$ that maximizes the numerical value function. The resulting pair $(v_t, \mathbf{u}_t)$ corresponding to that training data of $\boldsymbol{y}$ is the training data of the target.

5. **Backward Training of Surrogate Networks.** Surrogates $\mathring{V}_t^{\hat{\theta}_t}$ are trained iteratively recursively, from $t = T-1$ to $t = 1$, using the computed values and controls as targets. Each network depends on the previously trained surrogate for V_{t+1}.

By the end of this process, we obtain a sequence of trained neural networks that collectively approximate the value function across all intermediate time steps. The surrogate models enable efficient evaluation and control selection without the need for full forward simulation during deployment.

4.1. *Training data for Predictors.*

Assuming a VARIMA model is been fitted to the historical prices of some desired assets and the historical claim amount, and the matrices $\boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{\Sigma}$ are estimated. The first step to train a neural network surrogate is to generate n number of bounded training samples of the predictor $\boldsymbol{y} \in \mathbb{Y}$, each is denoted as ${}^{j}\check{\boldsymbol{y}}$, so that we have the training data $\check{\boldsymbol{Y}} \subseteq \mathbb{Y}$ of the state variables,

$$\check{\boldsymbol{Y}} = \begin{pmatrix} {}^{1}\check{\boldsymbol{y}}^{\intercal} \\ \vdots \\ {}^{n}\check{\boldsymbol{y}}^{\intercal} \end{pmatrix}$$

The bounds of $\check{\boldsymbol{y}}$ are of careful choices that should cover almost all possible values in the next T time points and cannot be impractical (e.g. though the absolute value of liability comes from a normal distribution and hence can go close to ∞, if T is chosen to be a few months, then the bounds must be reasonably large numbers but not ∞). Here we choose the bounds for $\boldsymbol{y}_{[1:(m+1)\cdot(d-1+p+q)+1]}$ (all state variables except the C) by simulating n sample paths of it T time steps forward. Let each sample at time t and path j to be denoted as ${}^{j}\tilde{\boldsymbol{y}}_{t[1:(m+1)\cdot(d-1+p+q)+1]}$. Based on the simulated maximum and minimum of each state variable, we can determine the bounds of the state variables other than C by observing the maximum and minimum of the samples, and the lower bound of C can be determined by

$$\min_{\forall i,j}\{\exp({}^{j}\Delta\tilde{S}_T^{(i)})\} \cdot \left(\dots \cdot \left(\min_{\forall i,j}\{\exp({}^{j}\Delta\tilde{S}_1^{(i)})\} \cdot (C_0 + \pi - \overline{u}^{(0)} \cdot C_0) - \max_{\forall j}\{\exp({}^{j}\ln L_1)\}\right) - \dots\right) - \max_{\forall j}\{\exp({}^{j}\ln L_T)\}$$

the upper bounds of C can be determined similarly,

$$\max_{\forall i,j}\{\exp({}^{j}\Delta\tilde{S}_T^{(i)})\} \cdot \left(\dots \cdot \left(\max_{\forall i,j}\{\exp({}^{j}\Delta\tilde{S}_1^{(i)})\} \cdot (C_0 + \pi - \underline{u}^{(0)} \cdot C_0) - \min_{\forall j}\{\exp({}^{j}\ln L_1)\}\right) - \dots\right) - \min_{\forall j}\{\exp({}^{j}\ln L_T)\}$$

Each element in training sample ${}^{j}\check{\boldsymbol{y}}$ is generated using sobol sequence sampling given that we obtained the bounds of all state variables in $\boldsymbol{y}$. Sobol sequence is a type of quasi-random sequence used to generate samples that uniformly cover a multi-dimensional space evenly, and hence will avoid clusters and gaps of sampling in the input space. In our implementation, we applied `qmc.Sobol` from `scipy.stats` to do so.

4.2. *One Step Transition Function.*

The second step is to define the one-step-forward transition function of y and $\mathbf{u}$, given the future random noise $B^{-1}\boldsymbol{w}$ is realised at some value $\tilde{\boldsymbol{w}}$. Define

$$B^{-1}\Delta^d \ln \boldsymbol{x}|\tilde{\boldsymbol{w}} = \boldsymbol{\mu_x} + \sum_{k=1}^{p} \boldsymbol{\Phi}^{(k)} \cdot \left(\Delta^d \ln B^{k-1}\boldsymbol{x} - \boldsymbol{\mu_x}\right) + \sum_{k=1}^{q} \boldsymbol{\Psi}^{(k)} \cdot B^{k-1}\boldsymbol{w} + \tilde{\boldsymbol{w}}$$

$$B^{-1}\Delta^s \ln \boldsymbol{x}|\tilde{\boldsymbol{w}} = \sum_{r=s}^{d-1} \Delta^r \ln \boldsymbol{x} + B^{-1}\Delta^d \ln \boldsymbol{x}|\tilde{\boldsymbol{w}}, \forall 0 \le s < d,$$

$$B^{-1}C(\mathbf{u})|\tilde{\boldsymbol{w}} = \sum_{i=2}^{m+1} \mathbf{u}_{[i]} \cdot \exp\left(B^{-1}\Delta^1 \ln \boldsymbol{x}|\tilde{\boldsymbol{w}}\right)_{[i-1]} \cdot (C + p - \mathbf{u}_{[1]} \cdot C) - \exp(\ln L + (B^{-1}\Delta^1 \ln \boldsymbol{x}|\tilde{\boldsymbol{w}})_{[m+1]})$$

this transition function g acts similarly to the forward shift operator B^{-1}:

$$\begin{aligned} g(\boldsymbol{y}, \mathbf{u}|B^{-1}\boldsymbol{w} = \tilde{\boldsymbol{w}}) := \big(& \\ & B^{-1}\Delta^d \ln \boldsymbol{x}|\tilde{\boldsymbol{w}}^\intercal, B^0\Delta^d \ln \boldsymbol{x}^\intercal, \ldots, B^{p-2}\Delta^d \ln \boldsymbol{x}^\intercal, \\ & \tilde{\boldsymbol{w}}, B^0\boldsymbol{w}^\intercal, \ldots, B^{q-2}\boldsymbol{w}^\intercal, \\ & (B^{-1}\Delta^{d-1} \ln \boldsymbol{x}|\tilde{\boldsymbol{w}})^\intercal, \ldots, (B^{-1}\Delta^1 \ln \boldsymbol{x}|\tilde{\boldsymbol{w}})^\intercal, \\ & \ln L + (B^{-1}\Delta^1 \ln \boldsymbol{x}|\tilde{\boldsymbol{w}})_{[m+1]}, B^{-1}C(\mathbf{u})|\tilde{\boldsymbol{w}} \\ & \big). \end{aligned}$$

4.3. *Target Generation via Quantized Dynamic Programming.*

The next step is to find the expectation component of the optimal value function numerically, we apply a quantization approach to do so. Specifically, we use the Gaussian quadrature in http://quantize.maths-fi.com/ with the estimated $\boldsymbol{\Sigma}^{1/2}$ multiplied to the quantized values. Given a level of quantization M, let $\boldsymbol{l} = (l^{(1)}, ..., l^{(M)})$ denotes the weights vector of occurring $(\tilde{\boldsymbol{w}}^{(1)\intercal}, ..., \tilde{\boldsymbol{w}}^{(M)\intercal})$, we define a numerically computed $\mathring{V}_t(\boldsymbol{y})$ such that $\mathring{V}_t \simeq V_t$,

$$\mathring{V}_t(\boldsymbol{y}) = \sup_{\forall \mathbf{u}_t \in \mathcal{U}_t} \left\{ \frac{1}{\gamma} \cdot (u_t^{(0)} \cdot C)^\gamma + \beta \cdot \sum_{r=1}^{M} l^{(r)} \cdot \mathring{V}_{t+1}\big(g(\boldsymbol{y}, \mathbf{u}_t | B^{-1}\boldsymbol{w} = \tilde{\boldsymbol{w}}^{(r)})\big) \right\}, \text{ whenever } \boldsymbol{Y}_t = \boldsymbol{y}$$

$$\mathring{V}_T(\boldsymbol{y}) = \frac{1}{\gamma} \cdot (C)^\gamma = V_T(\boldsymbol{y})$$

4.4. *Neural Network Design and Two-Stage Optimization.*

The final step is to find the approximation of $\mathring{V}_t(\boldsymbol{y})$ using backward iteration. To begin, we introduce a specific construction of our neural network that suits our purpose.

Consider a neural network $f(\boldsymbol{y}; \theta), \boldsymbol{y} \in \mathbb{R}^b$ of the following structure:

$$\begin{aligned} \boldsymbol{z}^{(0)} &= \boldsymbol{y}_{[:v]}, v \le b - 1. \\ \boldsymbol{g}^{(0)} &= \boldsymbol{y}_{[v+1:]} \\ \boldsymbol{z}^{(k^{(1)}+1)} &= \sigma^{(1)}(\boldsymbol{W}^{(k^{(1)}+1)} \cdot \boldsymbol{z}^{(k^{(1)})} + \boldsymbol{b}^{(k^{(1)}+1)}), 0 \le k^{(1)} \le K^{(1)} - 1 \\ \boldsymbol{g}^{(k^{(2)}+1)} &= \sigma^{(2)}(\boldsymbol{A}^{(k^{(2)}+1)} \cdot \boldsymbol{g}^{(k^{(2)})} + \boldsymbol{c}^{(k^{(2)}+1)}), 0 \le k^{(2)} \le K^{(2)} - 1 \\ \boldsymbol{r}^{(0)} &= (\boldsymbol{z}^{(K^{(1)})\intercal}, \boldsymbol{g}^{(K^{(2)})\intercal})^\intercal \\ \boldsymbol{r}^{(k^{(3)}+1)} &= \sigma^{(2)}(\boldsymbol{A}^{(K^{(2)}+k^{(3)}+1)} \cdot \boldsymbol{r}^{(k^{(3)})} + \boldsymbol{c}^{(K^{(2)}+k^{(3)}+1)}), 0 \le k^{(3)} \le K^{(3)} - 1 \\ f(\boldsymbol{y}; \theta) &= \boldsymbol{A}^{(K^{(2)}+K^{(3)})} \cdot \boldsymbol{r}^{(K^{(3)})} + \boldsymbol{c}^{(K^{(2)}+K^{(3)})} \end{aligned}$$

where $\theta = \{\boldsymbol{W}^{(k^{(1)})}, \boldsymbol{b}^{(k^{(1)})}, \boldsymbol{A}^{(s)}, \boldsymbol{c}^{(s)} | 0 \le k^{(1)} \le K^{(1)}, 0 \le s \le K^{(2)} + K^{(3)}\}$.

Proposition 1 (Partial Concave Neural Network)**.** *The function f is concave in $\boldsymbol{y}_{[v+1:]}$ if $\boldsymbol{A}^{(s)}$, $0 \le s \le K^{(2)} + K^{(3)}$ are element-wise non-negative and $\sigma^{(2)}$ is concave and non-decreasing.*

Proof. The proof is simple and follows from the fact that the composition of a concave, non-decreasing function and an affine, non-decreasing transformation is concave and non-decreasing. Such $\sigma^{(2)}$ can be obtained by rotating a convex, non-decreasing activation function (e.g. softplus, elu) by 180 degrees about the origin.

□

Now, let $\mathring{V}^{\theta_t} : \mathbb{Y} \to \mathbb{R}^{1+m}$ to be a neural network of the above design with $v = -2, f(\boldsymbol{y}; \theta) \in \mathbb{R}^{1+(m-1)}$ that outputs the value function as well as the controls. The key motivation to include the controls to the outputs instead of only to output the value function is that, when one state variable is dominating the value function, it is possible that the NN only 'learnt' the dominating state variable while completely ignoring the others. Besides, the regression tasks are not conflicting with each other, the control head serves as the auxillary output nodes that enforces the shared layers to 'see' the state variables other than the dominating one.

Denote the feedforward neural network surrogate of the closed-form solution of V_t as $\mathring{V}^{\hat{\theta}_t}{}_{[1]}$ where $\hat{\theta}_t$ is the trained neural network weights and biases. Then, define

$$v_t(\boldsymbol{y}, \mathbf{u}_t) = \begin{cases} \frac{1}{\gamma} \cdot (\mathbf{u}_{t_{[1]}} \cdot C)^{\gamma} + \beta \cdot \sum_{r=1}^{M} l^{(r)} \cdot \mathring{V}_{t+1}\big(g(\boldsymbol{y}, \mathbf{u}_t | \tilde{\boldsymbol{w}}^{(r)})\big)_{[1]}, t = T-1, \\ \frac{1}{\gamma} \cdot (\mathbf{u}_{t_{[1]}} \cdot C)^{\gamma} + \beta \cdot \sum_{r=1}^{M} l^{(r)} \cdot \mathring{V}^{\hat{\theta}_{t+1}}\big(g(\boldsymbol{y}, \mathbf{u}_t | \tilde{\boldsymbol{w}}^{(r)})\big)_{[1]}, t < T-1, \end{cases}$$

Given the closed-from expression of V_T, from $T-1$ to 1, for each $j = 1, ..., n$, we find the minimized value and the minimizer of the function $-1 \cdot v_t({}^{j}\check{\boldsymbol{y}}, \mathbf{u}_t)$ to obtain the training data for the target. Taking $v = -2$, we would like to state the following algorithm and proposition in order to attain the global minimum of $v_t(\boldsymbol{y}, \mathbf{u}_t)$ numerically,

Algorithm 1 Two-Stage Optimization for $\mathbf{u}_t$

Input: Current time t, state variables $\boldsymbol{y}$, function g, v_t, weight vector $\{l^{(r)}\}_{r=1}^{M}$, bounds $\underline{u}^{(0)}, \overline{u}^{(0)}, \underline{\mathbf{u}}_{[2:]} = (\underline{u}^{(1)}, \dots, \underline{u}^{(m-1)})^{\intercal}, \overline{\mathbf{u}}_{[2:]} = (\overline{u}^{(1)}, \dots, \overline{u}^{(m-1)})^{\intercal}$.
Initialize $\mathbf{u}_t \in \mathbb{R}^m$.

Step 1: Optimize $\mathbf{u}_{t_{[2:]}}$
Define the linear objective:

$$J(\mathbf{u}_{t_{[2:]}}) := -\sum_{r=1}^{M} l^{(r)} \cdot \sum_{i=2}^{m} \mathbf{u}_{t_{[i]}} \cdot \exp\big(B^{-1} \Delta^1 \ln \boldsymbol{x} \,|\tilde{\boldsymbol{w}}^{(r)}\big)_{[i-1]}$$

Use `scipy.optimize.minimize` or linear program solver to solve for optimal:

$$\mathbf{u}^{\star}_{t_{[2:]}} = \arg \min_{\mathbf{u}_{t_{[2:]}} \in [\underline{\mathbf{u}}_{[2:]}, \overline{\mathbf{u}}_{[2:]}]} J(\mathbf{u}_{t_{[2:]}})$$

Step 2: Optimize $\mathbf{u}_{t_{[1]}}$ with $\mathbf{u}^{\star}_{t_{[2:]}}$ fixed
Define scalar objective:

$$f(u_{[1]}) := -v_t\left(\boldsymbol{y}, [\mathbf{u}_{t_{[1]}}, \mathbf{u}^{\star}_{[2:]}]\right)$$

Use `scipy.optimize.minimize_scalar` to solve:

$$\mathbf{u}^{\star}_{t_{[1]}} = \arg \min_{\mathbf{u}_{t_{[1]}} \in [\underline{u}^{(0)}, \overline{u}^{(0)}]} f(\mathbf{u}_{t_{[1]}})$$

Output: $\mathbf{u}^{\star}_t = [\mathbf{u}^{\star}_{t_{[1]}}, \mathbf{u}^{\star}_{t_{[2:]}}]$
=0

Proposition 2 (Independence of the maximizer $\mathbf{u}_{t_{[1]}}$ and $\mathbf{u}_{t_{[2:]}}$)**.** *The numerically optimized value of $\min_{\mathbf{u}_{t_{[1]}}} \big\{\min_{\mathbf{u}_{t_{[2:]}}} \{-1 \cdot v_t(\boldsymbol{y}, \mathbf{u}_t)\}\big\}$ using Algorithm 1 converges to the global minimum of $-1 \cdot v_t(\boldsymbol{y}, \mathbf{u}_t)$.*

Proof. For any two control vectors $\mathbf{u}^{(a)}_{t_{[2:]}}, \mathbf{u}^{(b)}_{t_{[2:]}} \in [\underline{\mathbf{u}}_{[2:]}, \overline{\mathbf{u}}_{[2:]}]$, we have

$$\forall \mathbf{u}_{t_{[1]}} \in [\underline{u}^{(0)}, \overline{u}^{(0)}], J(\mathbf{u}^{(a)}_{t_{[2:]}}) < J(\mathbf{u}^{(b)}_{t_{[2:]}}) \Rightarrow -v_t(\boldsymbol{y}, [\mathbf{u}_{t_{[1]}}, \mathbf{u}^{(a)}_{t_{[2:]}}]) < -v_t(\boldsymbol{y}, [\mathbf{u}_{t_{[1]}}, \mathbf{u}^{(b)}_{t_{[2:]}}]),$$

because $C + p - C \cdot \mathbf{u}_{t_{[1]}} > 0$ for all admissible $\mathbf{u}_{t_{[1]}}$, larger $\sum_{i=2}^{m} \mathbf{u}_{t_{[i]}} \cdot \exp\left(B^{-1}\Delta^1 \ln \boldsymbol{x} | \tilde{\boldsymbol{w}}^{(r)}\right)_{[i-1]}$ implies larger $B^{-1}C(\mathbf{u}_t)|\tilde{\boldsymbol{w}}^{(r)}$ for any $\tilde{\boldsymbol{w}}^{(r)}$, and since $\mathring{V}^{\hat{\theta}_{t+1}}{}_{[1]}$ is concave and non-decreasing in $B^{-1}C(\mathbf{u}_t)|\tilde{\boldsymbol{w}}^{(r)}$, larger $B^{-1}C(\mathbf{u}_t)|\tilde{\boldsymbol{w}}^{(r)}$ implies larger $\mathring{V}^{\hat{\theta}_{t+1}}\big(g(\boldsymbol{y}, \mathbf{u}_t|\tilde{\boldsymbol{w}}^{(r)})\big)_{[1]}$; also, since all weights $l^{(r)}$ are positive, the strict inequality propagates through the summation, establishing the implication above.

Therefore, the optimal $\mathbf{u}_{t_{[2:]}}$ of the linear program $J(\mathbf{u}_{t_{[2:]}})$, denote as $\mathbf{u}^{\star}_{t_{[2:]}}$, is also optimal for $-v_t$, and is independent in the choice of $\mathbf{u}_{t_{[1]}}$. $\mathbf{u}^{\star}_{t_{[2:]}}$ can be easily obtained via `scipy.optimize` in a few iterations.

By substituting $\min_{\mathbf{u}_{t_{[1]}}} \left\{-1 \cdot v_t(\boldsymbol{y}, \mathbf{u}_{t_{[1]}}, \mathbf{u}^{\star}_{t_{[2:]}})\right\}$, given the concavity and non-decreasing property of $\frac{1}{\gamma}(\cdot)^{\gamma}$, and the fact that $\mathring{V}^{\hat{\theta}_{t+1}}{}_{[1]}$ is concave and non-decreasing in $\mathbf{u}_{t_{[1]}}$ when $\mathbf{u}_{t_{[2:]}}$ is fixed at $\mathbf{u}^{\star}_{t_{[2:]}}$ (recall $\mathbf{u}_{t_{[1]}}$ enters $\mathring{V}^{\hat{\theta}_{t+1}}{}_{[1]}$ via $B^{-1}C(\mathbf{u}_t)|\tilde{\boldsymbol{w}}^{(r)}$, the composition of a concave, non-decreasing function and an affine, non-decreasing transformation is concave and non-decreasing), $\mathbf{u}^{\star}_{t_{[1]}}$ can be easily obtained via `scipy.optimize` in a few iterations.

□

4.5. *Backward Training of Surrogate Networks.*

Using the above result, denote the minimized value and the minimizer as $(-1 \cdot {}^{j}\check{v}_t, {}^{j}\check{\boldsymbol{u}}_t)$. When training the mapping $\mathring{V}^{\hat{\theta}_t}$, we let $({}^{j}\check{v}_t, {}^{j}\check{u}_t)_{j=1}^{n}$ be the targets, and $({}^{j}\check{\boldsymbol{y}})_{j=1}^{n}$ be the predictors, we train the weights and biases $\hat{\theta}_t$ using the targets and predictors above mentioned to obtain $\hat{\theta}_t$ such that the training MSE is minimised, so the mapping of $\mathring{V}_t$ can be approximated by $\mathring{V}^{\hat{\theta}_{t+1}}$ and it is used in the next backward iteration.

Here, before the training of $\mathring{V}^{\theta_t}$, we allow it to be a function of y and $\mathring{V}^{\hat{\theta}_{t+1}}$; whereas after the training, the θ_t that minimises the MSE, i.e. the $\hat{\theta}_t$ is found, thus $\mathring{V}^{\hat{\theta}_t}$ is only a function of $\boldsymbol{y}$. The entire procedure is summarised as below,

Algorithm 2 Neural Network Surrogation Training for Value Function

```
1: Input: Historical asset and liability log-returns $\ln \boldsymbol{x}$; risk aversion parameter $\gamma$; discount factor $v$; simulation parameters;
   bounds $\underline{\mathbf{u}}, \overline{\mathbf{u}}$
2: Fit a VARIMA model to $\Delta \ln \boldsymbol{x}$, obtaining the estimated $\phi(B), \psi(B), \boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\Sigma}$
3: Generate $n$ simulated paths of $\Delta \ln \boldsymbol{x}$ forward $T$ time steps using the VARIMA and multivariate normal shocks given the
   estimated $\boldsymbol{\Sigma}$
4: Find the maximum and minimum of each simulated state variables in $\Delta \ln \boldsymbol{x}$, and calculate an estimate of bounds for $C_t$
5: Generate training inputs $\check{\boldsymbol{Y}}$ using Sobol sequences
6: Initialize: Empty neural network array V_hat_theta of dimension $T$; empty array target_train of dimension $(T, n, m)$
7: for $t = T-1, T-2, \ldots, 1$ do
8:     for each training sample $j = 1, 2, \ldots, n$ do
8:         if $t < T-1$ then $\mathring{V}^{\hat{\theta}_{t+1}} \leftarrow$ V_hat_theta[t+1]
9:         Define surrogate objective $v_t({}^{j}\check{\boldsymbol{y}}, \mathbf{u})$, using learned $\mathring{V}^{\hat{\theta}_{t+1}}$ if $t < T-1$, or explicit formula if $t = T-1$
10:        Solve $\mathbf{u}^{\star}_t = \arg\max_{\mathbf{u}} v_t({}^{j}\check{\boldsymbol{y}}, \mathbf{u})$ using Algorithm 1
11:        ${}^{j}\check{\boldsymbol{u}}_t \leftarrow \mathbf{u}^{\star}_t, {}^{j}\check{v}_t \leftarrow v_t^{(j)}(\boldsymbol{y}^{(j)}, \mathbf{u}^{\star}_t)$
12:        Store target_train[t][j] $({}^{j}\check{v}_t, {}^{j}\check{\boldsymbol{u}}_t)$
13:    end for
14:    Scale inputs ${}^{j}\boldsymbol{y}$ and outputs ${}^{j}\check{v}_t, {}^{j}\check{\boldsymbol{u}}_t$ using standard and min-max scalers
15:    Train a feedforward neural network $\mathring{V}^{\hat{\theta}_t}$ to map scaled ${}^{j}\boldsymbol{y} \mapsto ({}^{j}\check{v}_t, {}^{j}\check{\boldsymbol{u}}_t)$
16:    Store trained model to V_hat_theta[t]. Store scalers for time step $t$
17: end for
18: Return: Trained neural networks V_hat_theta, input/output scalers
   =0
```

The construction of the above neural network is documented in (https://github.com/yyunpeng/NNsurrogate2). It adopts the idea of [Tao Chen, Mike Ludkovski, and Moritz Voß (19 Dec 2023): On parametric optimal execution and machine learning surrogates, Quantitative Finance, DOI: 10.1080/14697688.2023.2282657]. However, here we do not have the approximation of the optimal control as a function of state variables, and the value function will be directly used to find the value of optimal control for each sample path. That is, to find the optimal control of any sample path with state variables y, we utilise the definition of the value function and thus the optimal control is given by $\arg\max_{\forall \mathbf{u}_t \in \mathcal{U}_t}\{\frac{1}{\gamma} \cdot (u_t^{(0)} \cdot C)^{\gamma} + \beta \cdot \boldsymbol{l}^{\intercal} \cdot \mathring{V}^{\hat{\theta}_t}\big(g(\boldsymbol{y}, \mathbf{u}_t|\tilde{\boldsymbol{w}}_t)\big)\}$. In this way, there is one less round of approximation, and for multiple output neurons, the MSE tends to be high during the training than that of the case with a single output neuron for the value function.

In fact, $\mathring{V}^{\hat{\theta}_t}$ is an asymptotically unbiased estimation of $\mathring{V}_t$, under the assumption of finite deep network, perfect optimization, and finite set of admissible strategy. Details can be referred to the Appendix.

5. APPENDIX

APPENDIX: ASYMPTOTIC UNBIASEDNESS OF THE VALUE FUNCTION ESTIMATOR

Lemma: Under the following assumptions:

(A1) Finite Deep Network: The neural network is deep enough with finite parameters, implying a finite VC dimension, such that the hypothesis space includes the true value function $\mathring{V}_t(\boldsymbol{y})$;

(A2) Perfect Optimization: The training process achieves the global minimizer of the empirical risk.

(A3) Finite Strategy Set: $\mathcal{U}_t$ is a finite set (i.e. decimal places are finite) for all t.

we can obtain $\mathbb{E}\Big[\big(\mathring{V}_t(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_t}(\boldsymbol{Y})\big)^2\Big] \to 0$ and,

$$\forall t \le T-2, \mathbb{E}\Big[\big(\mathring{V}_{t+1}(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_{t+1}}(\boldsymbol{Y})\big)^2\Big] \to 0 \Rightarrow \mathbb{E}\Big[\big(\mathring{V}_t(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_t}(\boldsymbol{Y})\big)^2\Big] \to 0$$

Proof:
Denote

$$ER(\hat{\theta}_{T-1}) = \frac{1}{n}\sum_{i=1}^{n}\big(\mathring{V}_{T-1}({}^{\boldsymbol{i}}\boldsymbol{y}) - \mathring{V}^{\hat{\theta}_{T-1}}({}^{\boldsymbol{i}}\boldsymbol{y})\big)^2$$

$$PR(\hat{\theta}_{T-1}) = \mathbb{E}\Big[\big(\mathring{V}_{T-1}(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_{T-1}}(\boldsymbol{Y})\big)^2\Big].$$

where ${}^{\boldsymbol{i}}\boldsymbol{y}$ is one training sample, $\mathring{V}^{\hat{\theta}_t}$ is the trained NN surrogate of the approximated value function with the expectation computed using a quantization method, and $\hat{\theta}_t$ is the trained optimal parameters of $\mathring{V}^{\hat{\theta}_t}$.

Under A2, we have $ER(\hat{\theta}_{T-1}) \to 0$ as $n \to \infty$. By the Vapnik–Chervonenkis theory, a finite network with finite parameters (A1) has a finite VC dimension. This implies the uniform convergence of the empirical risk (ER) to the population risk (PR) holds. Thus, for the empirical risk minimizer $\mathring{V}^{\hat{\theta}_t}$, we have

$$\sup_{\forall \hat{\theta}_{T-1}} |ER(\hat{\theta}_{T-1}) - PR(\hat{\theta}_{T-1})| \to 0 \text{ as } n \to \infty$$

Combining what we have,

$$\sup_{\forall \hat{\theta}_{T-1}} |ER(\hat{\theta}_{T-1}) - PR(\hat{\theta}_{T-1})| \ge |ER(\hat{\theta}_{T-1}) - PR(\hat{\theta}_{T-1})| \Rightarrow PR(\hat{\theta}_{T-1}) = \mathbb{E}\Big[\big(\mathring{V}_{T-1}(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_{T-1}}(\boldsymbol{Y})\big)^2\Big] \to 0.$$

Let us prove the statement by induction with the asymptotic condition. For the base case, using a similar argument as before, we have

$$\frac{1}{n}\sum_{i=1}^{n}\Big(\sup_{\forall \mathbf{u}\in\mathcal{U}_{T-2}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot {}^{\boldsymbol{i}}\boldsymbol{y}_{[-1]})^{\gamma} + \beta\cdot\boldsymbol{w}^{\intercal}\cdot\mathring{V}^{\hat{\theta}_{T-1}}\big(g({}^{\boldsymbol{i}}\boldsymbol{y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}({}^{\boldsymbol{i}}\boldsymbol{y})\Big)^2$$

$$\to \mathbb{E}\Big[\Big(\sup_{\forall \mathbf{u}\in\mathcal{U}_{T-2}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot \boldsymbol{Y}_{[-1]})^{\gamma} + \beta\cdot\boldsymbol{w}^{\intercal}\cdot\mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y})\Big)^2\Big] \to 0$$

Here we want to prove the base case by showing

$$\Big|\Big|\mathring{V}_{T-2}(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y})\Big|\Big|_2 \to 0$$

where $||Z||_2 = \mathbb{E}(Z^2)^{1/2}$. Proving so implies $\mathbb{E}\Big[\big(\mathring{V}_{T-2}(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y})\big)^2\Big] \to 0$. In the following we abuse the $\forall \mathbf{u} \in \mathcal{U}_{T-2}$ to be $\forall \mathbf{u}$ for simplicity.

$$\Big|\Big|\mathring{V}_{T-2}(\boldsymbol{Y}) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y})\Big|\Big|_2$$
$$= \Big|\Big|\sup_{\forall \mathbf{u}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot \boldsymbol{Y}_{[-1]})^{\gamma} + \beta\cdot\boldsymbol{w}^{\intercal}\cdot\mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y})\Big|\Big|_2$$

$$
\begin{aligned}
&= \Big|\Big|\sup_{\forall \mathbf{u}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot \boldsymbol{Y}_{[-1]})^{\gamma} + \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y}) + \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big)\Big|\Big|_2\\
&= \Big|\Big|\sup_{\forall \mathbf{u}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot \boldsymbol{Y}_{[-1]})^{\gamma} + \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y}) + \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big)\Big|\Big|_2\\
&\le \Big|\Big|\sup_{\forall \mathbf{u}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot \boldsymbol{Y}_{[-1]})^{\gamma} + \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y}) + \beta\cdot \sup_{\forall \mathbf{u}} \boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big)\Big|\Big|_2\\
&\le \Big|\Big|\sup_{\forall \mathbf{u}} \frac{1}{\gamma}\cdot(\mathbf{u}_{[1]}\cdot \boldsymbol{Y}_{[-1]})^{\gamma} + \beta\cdot \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \mathring{V}^{\hat{\theta}_{T-2}}(\boldsymbol{Y})\Big|\Big|_2 + \beta\cdot \Big|\Big|\sup_{\forall \mathbf{u}} \boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big)\Big|\Big|_2
\end{aligned}
$$

where the last inequality comes from the Triangle Inequality in L^2. For the first term in the above, it approaches 0 because of (A1) and (A2). For the second term, under a finite $\mathcal{U}_{T-2}$ with $||\mathcal{U}_{T-2}|| = \kappa$, we have

$$
\mathbb{E}\Big[\Big(\sup_{\forall \mathbf{u}\in\mathcal{U}_{T-2}} \boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big) - \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}|\boldsymbol{\epsilon}_t)\big)\Big)^2\Big] = \mathbb{E}\Big[\Big(\max_{\forall 1\le j\le \kappa} \boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{\epsilon}_t\big) - \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{\epsilon}_t)\Big)^2\Big].
$$

Since for any $\mathbf{u}_j$, we have

$$
\begin{aligned}
&\mathbb{E}\Big[\Big(\boldsymbol{w}^{\intercal}\cdot \mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{\epsilon}_t\big) - \boldsymbol{w}^{\intercal}\cdot \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{\epsilon}_t)\Big)^2\Big]\\
&= \mathbb{E}\Big[\sum_{s=1}^{l}\sum_{h=1}^{l} w_s\cdot \Big(\mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(s)}\big) - \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(s)})\big)\Big)\cdot w_h\cdot \Big(\mathring{V}_{T-1}(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(h)}\big) - \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(h)})\big)\Big)\Big]\\
&= \sum_{s=1}^{l}\sum_{h=1}^{l} w_s\cdot w_h\cdot \mathbb{E}\Big[\Big(\mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(s)}\big) - \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(s)})\big)\Big)\cdot \Big(\mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(h)}\big) - \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(h)})\big)\Big)\Big]
\end{aligned}
$$

Denote $\mathring{V}_{T-1}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(s)}\big) - \mathring{V}^{\hat{\theta}_{T-1}}\big(g(\boldsymbol{Y},\mathbf{u}_j|\boldsymbol{e}_t^{(s)}\big)$ as $Z^{(s)}$, by Cauchy-Schwarz Inequality and A2, we have

$$
|\mathbb{E}(Z^{(s)}\cdot Z^{(h)})| \le \big[\mathbb{E}\big({Z^{(s)}}^2\big)\cdot \mathbb{E}\big({Z^{(h)}}^2\big)\big]^{\frac{1}{2}} \to 0
$$

and hence the summation goes to 0. Thus, the base case is true.