A comprehensive machine learning framework for dynamic portfolio choice with transaction costs*

Luca Gaegauf, Simon Scheidegger, Fabio Trojani†

August 17, 2023

Abstract

We introduce a comprehensive computational framework for solving dynamic portfolio choice problems with many risky assets, transaction costs, and borrowing and short-selling constraints. Our approach leverages the synergy between Gaussian process regression and Bayesian active learning to efficiently approximate value and policy functions with a novel, formal way of characterizing the irregularly-shaped no-trade region; we then embed this into a discrete-time dynamic programming algorithm. This combination allows us to study dynamic portfolio choice problems with more risky assets than was previously possible. Our results indicate that giving the agent access to more assets may alleviate some illiquidity resulting from the presence of transaction costs.

JEL classification: C61, C63, C68, E21.

Keywords: Machine learning, computational finance, computational economics, Gaussian process regression, dynamic portfolio optimization, transaction costs, liquidity premia.

*We thank Felix Kübler, as well as seminar participants at the University of Geneva and Zürich, for their valuable comments. This work was generously supported by the Swiss National Science Foundation (SNF), under project IDs "Can Economic Policy Mitigate Climate Change?", "New methods for asset pricing with frictions", the Enterprise for Society (E4S), and the Geneva Institute for Wealth Management.

†Gaegauf at Department for Banking and Finance, University of Zurich; email: luca.gaegauf@bf.uzh.ch. Scheidegger at Department of Economics, University of Lausanne; Enterprise for Society (E4S); email: simon.scheidegger@unil.ch. Trojani at Geneva Finance Research Institute, University of Geneva , the Swiss Finance Institute, the University of Turin; email: fabio.trojani@unige.ch.

1 Introduction

In portfolio optimization, market frictions, such as transaction costs, are a focal area of investigation. Despite significant advancements, research predominantly addresses scenarios with a limited number of assets, leaving unanswered questions about the impact of transaction costs on portfolios comprising diverse assets. This gap becomes particularly noteworthy when considering the dynamic nature of transaction costs and liquidity premia. Given the broad application of transaction costs in finance, many questions remain unanswered: How do regulatory shifts pertaining to transaction costs influence market behaviors (Albuquerque et al., 2020)? How does investor behavior change when transaction cost rates change (Albuquerque et al., 2020)? How do high transaction costs in emerging markets impact international portfolio diversification strategies (Baule, 2008)? What interplay exists between investor behavioral biases and transaction costs in determining portfolio configurations? Additionally, with the rise of technological developments (*e.g.*, blockchain), there is potential for new intersections with transaction costs, prompting traditional portfolio theories to be updated (Liu, 2019). Although there have been advancements in computational methods addressing these questions, most approaches often limit the number of assets or employ oversimplified assumptions. Such simplifications may overlook important interactions, co-dependencies, or offsetting effects.

To address the complexities of multi-asset models, we present a generic computational framework designed to solve high-dimensional, finite-horizon, discrete-time dynamic stochastic portfolio optimization with proportional transaction costs and borrowing and shorting constraints. The novelty of our methodology is two-fold: first, we integrate Gaussian process regressions (GPRs) in a dynamic programming (DP) framework to precisely approximate value and policy functions; second, we innovate a technique for estimating the no-trade region (NTR) during each DP iteration.[1] Using GPRs, we can focus our computational efforts on the state space's non-hypercubic domain rather than waste computation outside, as is the case with grid-based methods. Additionally, our grid-free approach combined with an approximation of the NTR allows us to sample points strategically where they are needed. Consequently, we reach solutions using fewer points than competing methods and can scale our model to higher dimensions than was previously possible.

We test our method on a portfolio choice model with two to five risky assets, a risk-free bond, and per-period consumption. Given an initial endowment, an agent with CRRA utility manages a portfolio of said assets in discrete-time intervals over a

[1]The NTR—a key feature in portfolio choice settings with proportional transaction costs—is the region in the state space where the transaction costs offset the benefit of rebalancing the portfolio. See Section 2 for a more thorough explanation.

finite horizon to maximize her lifetime expected utility of consumption. We use two parameterizations to evaluate our method's efficacy and the economic drivers behind the agent's actions. Specifically, we use the parameterization from Schober et al. (2022) to draw a direct comparison to the current state-of-the-art. We additionally, use a systematic two-risky-asset parameterization to isolate the effect of individual parameters on the shape of the NTR. We conduct several numerical experiments using the former parameterization to evaluate our method's efficacy. First, we evaluate our method using value function fit to assess how well the GPR approximates the value function. Second, we study the Euler errors (EEs) to understand how well the intertemporal problem is solved. Third, we monitor the quality of the NTR approximation. Unlike prior evaluation attempts in existing literature, our approach allows us to pinpoint inaccuracies by decomposing the evaluation into these three components. This approach establishes a benchmark, facilitating a direct comparison with future methods. Then, using both of the previously mentioned parameterizations, we perform several analyses to uncover the economic drivers behind the agent's actions. We investigate the NTR and its sensitivity to varying economic parameters, which, for example, provide insights into how a risk-averse agent responds to changes in an asset's variance when faced with transaction costs. Finally, we conclude by conducting several Monte Carlo simulations, demonstrating the agent's responses to illiquidity. Given that existing methods for computing the liquidity premium proved inadequate in our context, we suggest a unique approach for the setting with CRRA utility and per-period consumption. Initial results suggest that a more diverse asset portfolio can mitigate the effects of illiquidity, especially when assets are correlated. Our results underscore the reliability and versatility of our method across different finance parameterizations.

The remainder of the paper is organized as follows: Section 2 briefly reviews the related literature. Section 3 describes the benchmark dynamic portfolio choice model with transaction costs. Section 4 introduces our solution method. In Section 5, we evaluate our method's performance using two parameterizations and in Section 6, we present our results and economic insights. Section 7 concludes. Furthermore, we elaborate more on the computational details in Appendix D, with Appendix D.3 focusing specifically on our parallelization scheme.

2 Literature review

Portfolio optimization has been a subject of extensive research since Merton (1971) first formulated and solved such a model in continuous time (*e.g.*, Gârleanu and Pedersen, 2013; Muhle-Karbe et al., 2023). Merton's seminal work proposes an optimal portfolio allocation that balances risk and return, suggesting that investors should reallocate their wealth if the allocation deviates from the optimum due to price changes. In practice, this rebalancing would require infinitely frequent updates with infinitely small adjustments, which is not observed in reality. To address this issue, researchers extended the model by Merton (1971) to include proportional transaction costs (see, *e.g.*, Constantinides, 1979; Kamin, 1975; Magill and Constantinides, 1976). A key feature when trading with transaction costs is that the portfolio reallocation policy is defined by an NTR: a region in the state space that contains the no-transaction-cost optimal allocation, where the benefit of reallocating wealth is offset by transaction costs. In studies with multiple risky assets, the literature so far has found that the NTR typically has the approximate shape of a square or parallelogram (see, *e.g.*, Lynch and Tan, 2010).[2]

The policy to rebalance a portfolio derived from settings with proportional transaction costs suggests that when the portfolio weights fall within the NTR, they can move freely,[3] whereas when outside of the NTR, portfolios are updated to a point on the NTR boundary (Constantinides, 1979). This means that as deviations from the no-transaction-cost optimum are allowed within the NTR, transaction cost payments accrued through the infinite-reallocation strategy are avoided. However, as Baccarin and Marazzina (2014) note, the optimal policy in a continuous-time setting still requires many infinitesimally small transactions when portfolio weights move outside the NTR. This issue can be solved mechanically by reformulating the portfolio choice problem in discrete time and only by considering investors who update their portfolios periodically (*e.g.*, quarterly or annually).

In the pursuit of more comprehensive models, Lynch and Tan (2010) underscore the importance of accounting for multiple assets by modeling two risky assets: a market index and a risky stock. They find that investors experience a 2%–25% utility loss when access to the risky stock is restricted, forcing them to hold only the market index. This result provides support for moving beyond models with a single risky asset to address the empirical demand for multi-asset solution methods.[4] However, scaling models to higher dimensions (*i.e.*, more assets) is challenging as the computational complexity increases exponentially; a phenomenon known as the curse of dimensionality (see, Bellman, 1961). Kamin (1975) and Akian et al. (1996) solved models with two

[2]This finding seems to be robust across various formulations of dynamic portfolio choice problems with transaction costs. Mei et al. (2016), for instance, derive a closed-form solution for a parallelogram-shaped NTR for an investor with quadratic utility; Muthuraman and Zha (2008) approximate a 2 to 7-dimensional parallelogram-shaped NTR in an infinite-horizon setting; Dybvig and Pezzo (2020) solve a continuous-time problem with a mean-variance investor.

[3]That is, small deviations of the relative asset weights within the NTR do not results in the agent rebalancing their portfolio.

[4]Empirical demand that stems from hedging requirements, optimizing risk-to-return ratios, and facilitating customized investment objectives.

risky assets, with and without a risk-free asset, respectively.[5] While many authors claim their methods can accommodate more assets, they struggle to overcome the curse of dimensionality in practice (see, *e.g.*, Magill and Constantinides, 1976; Abrams and Karmarkar, 1980; Leland, 2000; Zhang et al., 2019). To extend the number of assets, some researchers resort to simplifying assumptions. For example, Atkinson and Mokkhavesa (2004) compute the optimal policy for D uncorrelated assets; Mei et al. (2016) derives a closed-form solution for an D-risky-asset model for a mean-variance investor who optimally reallocates their wealth only once at the beginning of their lifetime; Brown and Smith (2011) employ a heuristic simulation-based method to compute policies for up to 10 risky assets but are unable to compute optimal policies; and Muthuraman and Zha (2008) determine the infinite-horizon NTR for up to 7 risky assets but, likewise, cannot compute optimal policies to rebalance the portfolio.

More recently, Schober et al. (2022)[6] have made progress in addressing the curse of dimensionality, solving a discrete portfolio choice problem with 5 risky assets. Their innovative use of adaptive sparse grids enabled them to sample fewer points than competing methods based on Cartesian grids. In contrast to dense grids, which consist of N^D data points for an D-asset problem with N grid points along each dimension, adaptive sparse grids have selectively placed points in regions of the hypercube that require higher local resolution, rather than increasing the resolution evenly across the computational domain (see, *e.g.*, Bungartz and Griebel, 2004; Brumm and Scheidegger, 2017; Scheidegger and Treccani, 2018). However, despite this innovation, the curse of dimensionality is only partially mitigated due to the rigid geometry inherent in grid-based approaches. Specifically, the computational domain of the said grid is still a hypercube, while, due to the no-borrowing and no-shorting constraints, the state space of the problem under consideration is an D-dimensional simplex (where D represents the number of risky assets). The volume of the feasible state space within a unit-hypercube shrinks at a rate proportional to the inverse factorial of the dimensionality of the problem, that is, $1/D!$. Consequently, a significant amount of computational effort is wasted on points that fall outside the feasible state space, ultimately limiting the number of assets that can be modeled if any grid-based method is used.

[5]Other research with 2 risky assets includes Oksendal and Sulem (2002); Janeček and Shreve (2004); Goodman and Ostrov (2010); Lynch and Tan (2010); Dybvig and Pezzo (2020).

[6]Their approach is based on the test case formulated by Cai et al. (2013).

3 A benchmark model of dynamic portfolio choice with transaction costs

In this section, we outline a canonical dynamic portfolio choice problem with transaction costs similar to the one previously studied by Schober et al. (2022). We have chosen this benchmark because this dynamic portfolio choice problem scales naturally with an increasing number of risky assets and, therefore, can serve us as an ideal benchmark to assess the performance of our novel solution algorithm (see Sec. 6).

3.1 The asset market

We consider a discrete-time dynamic portfolio choice model with a finite time horizon T in which an investor seeks to maximize their expected lifetime utility. The horizon $[0, T]$ is discretized into $T+1$ equally spaced periods at which the investor can rebalance her portfolio. There is a single investor that can allocate her wealth in the asset market among $D+1$ assets. D assets are risky (also referred to as "stocks") and are subject to proportional transaction costs. That is, when buying or selling a stock, the investor pays a transaction cost $\tau \in [0,1]$ proportional to the amount spent.[7] The stock's stochastic one-period gross returns are denoted by $\mathbf{R} = (R_1, ..., R_D)^\top$.

In addition, the agent can invest in a risk-free bond that is not subject to transaction costs and yields a gross one-period return $R_f = \exp(r)$, where r is the one period interest rate.

3.2 The investor's problem

There is a single consumption good (the numeraire) and an investor who maximizes her lifetime utility of consumption, that is,

$$\mathbb{E}_0\left[\sum_{t=0}^{T} \beta^t u(c_t W_t)\right], \tag{1}$$

where $\beta < 1$ is the agent's time discount factor, $u(\cdot)$ is the agent's utility function, c_t is the fraction of wealth spent on the consumption good in period t, and W_t is the investor's wealth. The investor tracks both wealth W_t and, due to the presence of transaction costs, the composition of wealth $\mathbf{x}_t$ (Zabel, 1973). In each period, the agent reallocates her wealth W_t among the $D+1$ assets and consumes the consumption good

[7]Note, however, that while the solution method introduced in this paper is illustrated in the context of proportional transaction costs for reasons of simplicity, it can be applied to alternative types of other transaction costs that may, for instance, be state-space dependent $\tau(\theta_t)$, asset dependent $\boldsymbol{\tau} = (\tau_1, ..., \tau_D)^\top$, or differ for purchases and sales $\{\tau_+, \tau_-\}$.

c_t. That is, given a t^{th}-period asset holding $\mathbf{x}_t$, where $\mathbf{x}_t = (x_{1,t}, ..., x_{D,t})^\top \in [0,1]^D$ denotes the fraction of wealth allocated to stocks, the agent decides how much of each stock to buy and sell. Specifically, the investor updates her portfolio by $\boldsymbol{\delta}_t = (\delta_{1,t}, ..., \delta_{D,t})$. $\delta_{i,t}$ denotes the fraction of wealth the investor uses to buy or sell the i^{th} stock, where $\delta_{i,t} > 0$ indicates a purchase and $\delta_{i,t} < 0$ indicates a sale of said asset. On each transaction, the agent pays $\tau|\delta_{i,t}W_t|$ for each asset $i \in [1, ..., D]$. After subtracting the transaction costs, the remaining wealth of the agent that is not invested in the risky assets or spent on the consumption good, is saved in the bond:

$$b_t W_t = \left(1 - \mathbf{1}^\top \cdot \mathbf{x}_t\right) W_t - \mathbf{1}^\top \cdot \boldsymbol{\delta}_t W_t - \mathbf{1}^\top \cdot (\tau|\boldsymbol{\delta}_t|W_t) - c_t W_t. \tag{2}$$

3.3 The dynamic portfolio choice problem

The dynamic portfolio choice problem that the investor faces is given by:

$$V_t\left(W_t, \mathbf{x}_t\right) = \max_{c_t, \boldsymbol{\delta}_t} \left\{ u\left(c_t W_t\right) + \beta \mathbb{E}_t\left[V_{t+1}\left(W_{t+1}, \mathbf{x}_{t+1}\right)\right] \right\} \quad \text{for } t < T,$$

for some initial wealth holding $W_0 > 0$. The respective bond holdings are defined by Equation (2) and $\mathbb{E}_t[\cdot]$ is the expectation operator. Both W_{t+1} and $\mathbf{x}_{t+1}$ are random variables as they are functions of the random variable $\mathbf{R}_t$. The dynamics of the $D+1$ state variables $\{W_{t+1}, \mathbf{x}_{t+1}\}$ are given by:

$$\begin{aligned} W_{t+1} &= b_t W_t R_f + \left(\left(\mathbf{x}_t + \boldsymbol{\delta}_t\right) W_t\right)^\top \cdot \mathbf{R}_t, \\ \mathbf{x}_{t+1} &= \frac{\left(\mathbf{x}_t W_t + \boldsymbol{\delta}_t W_t\right) \odot \mathbf{R}_t}{W_{t+1}}. \end{aligned}$$

The symbol $\odot$ represents the Hadamard product, given by $(\mathbf{a} \odot \mathbf{b})_i := a_i b_i$. The terminal value function is given by:

$$V_T\left(W_T, \mathbf{x}_T\right) = u\left((1 - \tau \mathbf{1}^\top \cdot \mathbf{x}_T) W_T\right).$$

That is, we assume that in the terminal period T, the investor must sell the remaining holdings on the market (*i.e.*, pay transaction costs) before consuming the full amount. Furthermore, the following constraints must hold:

$$\boldsymbol{\delta}_t W_t \geq -\mathbf{x}_t W_t, \tag{3}$$

$$b_t W_t \geq 0, \tag{4}$$

$$\mathbf{1}^\top \cdot \mathbf{x}_t \leq 1, \quad \text{for } t \in [0, T]. \tag{5}$$

Equations (3) and (4) constitute no-shorting constraints, and Equation (5) constitutes the no-borrowing constraint. Furthermore, we limit the range of $x_{i,t}$ to $[0, 1]$ for $i \in [1, ..., D]$. Finally, note that the $|\boldsymbol{\delta}_t|$ term in Equation (2) is non-differentiable. To counter this, we follow Cai et al. (2013) and use the common trick of decomposing $\boldsymbol{\delta}_t$ into its buying and selling components, $\boldsymbol{\delta}_t^+ = (\delta_{1,t}^+, ..., \delta_{D,t}^+)$ and $\boldsymbol{\delta}_t^- = (\delta_{1,t}^-, ..., \delta_{D,t}^-)$, respectively, with $\boldsymbol{\delta}_t^+, \boldsymbol{\delta}_t^- \geq 0$. Thus, we can write $\boldsymbol{\delta}_t := \boldsymbol{\delta}_t^+ - \boldsymbol{\delta}_t^-$, and Equation (2) can consequently be reformulated as:

$$b_t W_t = \left(1 - \mathbf{1}^\top \cdot \mathbf{x}_t\right) W_t - \mathbf{1}^\top \cdot (\boldsymbol{\delta}_t^+ - \boldsymbol{\delta}_t^-) W_t - \mathbf{1}^\top \cdot (\tau(\boldsymbol{\delta}_t^+ + \boldsymbol{\delta}_t^-) W_t) - c_t W_t.$$

3.3.1 The dynamic problem for an investor with CRRA utility

We simplify the problem by normalizing the value function $v(\mathbf{x}) = V(W, \mathbf{x})/W^{1-\gamma}$, allowing us to drop wealth as a state variable during optimization,[8] which allows us to solve the dynamic program globally. It, however, does not provide much in terms of reducing the curse of dimensionality, as we still have an D-dimensional state space.

We redefine the investor's normalized bond holdings b_t in period t as

$$b_t = 1 - \mathbf{1}^\top \cdot (\mathbf{x}_t - \boldsymbol{\delta}_t^+ + \boldsymbol{\delta}_t^- - \tau(\boldsymbol{\delta}_t^+ + \boldsymbol{\delta}_t^-)) - c_t, \tag{6}$$

and the state dynamics from t to $t + 1$ can be expressed free of the investor's wealth, that is,

$$\pi_{t+1} := b_t R_f + (\mathbf{x}_t + \boldsymbol{\delta}_t)^\top \cdot \mathbf{R}_t \tag{7}$$

$$W_{t+1} = W_t \pi_{t+1}, \tag{8}$$

$$\mathbf{x}_{t+1} = \frac{(\mathbf{x}_t + \boldsymbol{\delta}_t) \odot \mathbf{R}_t}{\pi_{t+1}}. \tag{9}$$

With this normalization, the investor solves the following dynamic program:

$$v_t(\mathbf{x}_t) = \max_{c_t, \boldsymbol{\delta}_t} \left\{ u(c_t) + \beta \mathbb{E}_t \left[\pi_{t+1}^{1-\gamma} v_{t+1}(\mathbf{x}_{t+1}) \right] \right\}, \quad t < T, \tag{10}$$

where the dynamics are given by (7)–(9), and the terminal value function by:

$$v_T(\mathbf{x}_T) = u\left(1 - \tau \mathbf{1}^\top \cdot \mathbf{x}_T\right), \tag{11}$$

The assumption that an agent must first sell asset holdings on the market before consuming total wealth remains unchanged. The above constraints hold for $t = 0, \ldots, T$

[8]This modeling choice does not imply that wealth is absent from an investor's lifetime utility, but merely that wealth levels do not influence an investor's choice.

and are reexpressed as:

$$\boldsymbol{\delta}_t \geq -\mathbf{x}_t, \tag{12}$$

$$b_t \geq 0, \tag{13}$$

$$\mathbf{1}^\top \cdot \mathbf{x}_{t+1} \leq 1, \tag{14}$$

where the Equations (12) and (13) are no-shorting and Equation (14) is the no-borrowing constraint. Now the investor's optimization problem no longer depends on W_t, and hence one state variable can be eliminated, reducing the problem to one of D dimensions. The non-normalized optimal choices can be obtained by multiplication with a given wealth W_t for any time t and state $\boldsymbol{x}_t$.

Finally, we define the NTR as

$$\Omega_t = \{\mathbf{x}_t : \boldsymbol{\delta}_t^* = 0\},$$

where $\boldsymbol{\delta}_t^*$ and c_t^* are the optimal policies for a given $\mathbf{x}_t$.

4 A scalable solution method for dynamic portfolio choice problems with transaction costs

In this section, we introduce a versatile, scalable, and flexible method based on supervised machine learning[9] for solving discrete-time dynamic portfolio optimization problems that incorporate (proportional) transaction costs. An efficient dynamic portfolio solution approach must accommodate several key characteristics. First, it needs to address irregular state space geometries, as Equations (12) and (14) constrain the feasible domain of the state space to a D-simplex (where D is the number of risky assets). Second, it should be able to handle moderate to high-dimensional state spaces,[10] as the dimensionality of the problem increases with the number of stocks being modeled. Providing the investor with a more diverse array of assets results in a more complex problem. Third, the method must handle non-differentiabilities in the value and policy functions, which stem from the non-differentiable absolute value function introduced by proportional transaction costs.

Existing methods that rely on Cartesian grids suffer from two main drawbacks: first, their high demand for points as dimensions scale,[11] and second, their inability to

[9]We follow the canonical machine learning language conventions: *supervised machine learning* refers to the training of a (regression or classification) model using a labeled dataset, that is, a dataset where each observation has a corresponding "answer" value. For more details, see, *e.g.*, Murphy (2012).

[10]That is, problems with $D \geq 3$ risky assets.

[11]An equidistant Cartesian grids yields N^D points, where N is the grid discretization, and D is the

sample points flexibly to achieve high local resolution around important features such as kinks. Schober et al. (2022) partially mitigate these issues by using adaptive sparse grids. However, their method still suffers from the curse of dimensionality, as they sample points on a hyper-cube rather than within the D-dimensional simplex induced by the model constraints. The volume of the D-dimensional simplex is $1/D!$, and hence it shrinks rapidly with increasing dimensionality. Consequently, with their approach, the amount of computation wasted on infeasible states grows with the dimension of the problem.

We propose using discrete-time DP where we approximate the value and policy functions with GPRs (see, *e.g.*, Williams and Rasmussen, 2006, for a textbook treatment).[12] In addition, we incorporate a novel approach for generating an approximation of the NTR within each DP iteration. This combination enables us to efficiently address the aforementioned challenges, allowing for more effective scaling to higher dimensions compared to existing methods.

The main strength of our solution algorithm lies in GPR's grid-free nature, making the solution method independent of the problem's geometry. This feature alone offers several advantages, but when combined with an approximation of the NTR, it significantly enhances the application. First, the grid-free GPRs enable us to focus computation on the hyper-simplex rather than a hypercube. While this does not entirely ameliorate the curse of dimensionality, it enables more efficient scaling compared to existing literature. Second, the grid-free nature of GPRs allows us to sample points around kinks to improve the approximation quality of these features. We utilize the previously mentioned NTR approximation to facilitate this step, as the NTR provides valuable information about the locations of kinks. This approach maintains a sample size that is small enough for tractability yet large enough for good performance, helping us avoid the computational overhead associated with scaling a problem. Third, the grid-free GPRs enable us to segment our state space and fit separate GPRs to different state-space regions arbitrarily. Using the NTR approximation, this strategy allows us to avoid approximating kinks around the NTR with the smooth GPR. Lastly, the NTR approximation provides additional computational benefits. For instance, we can predetermine the sign of the $\boldsymbol{\delta}$-policy, enabling us to efficiently constrain our optimization problem. Furthermore, the NTR offers a good initial guess for the solver, thereby reducing the time to solution. The remainder of this section presents a detailed overview of our algorithm.

dimension of the state space.

[12]GPRs are named as such, as they use Gaussian processes (GPs), a collection of random variables that have a joint Gaussian distribution, to predict continuous values. GPs and GPRs will be discussed in more detail in Section 4.2.

4.1 Abstract problem formulation & dynamic programming

The abstract class of problem we focus on is a discrete-time, finite-horizon stochastic optimal decision-making problem in the context of dynamic portfolio optimization. We provide a brief characterization, following the notation by Renner and Scheidegger (2018): let $\mathbf{x}_t \in \mathcal{B} \subset \mathbb{R}^D$ represent the state of the economy at time $t \in [0, 1, \ldots, T]$, where $D \in \mathbb{N}$ denotes the dimensionality of the state space. Controls (actions) are described by a period-specific policy function $p_t : \mathcal{B} \to \zeta$, with $\zeta \subset \mathbb{R}^{D+1}$ being the space of possible controls. The transition function for the economy between periods is given by a distribution π, which depends on the current state and policies, i.e., $\mathbf{x}_{t+1} \sim \pi(\cdot|\mathbf{x}_t, p_t(\mathbf{x}_t))$. The policy function p_t needs to be determined from optimality conditions, given the stochastic transition function π that maps a state-action pair to a successor state.

DP is the standard approach for finding the sequence of controls $\{\chi_t\}_{t=0}^{T}$ to maximize the value function $V(\mathbf{x}_0) := \mathbb{E}\left[\sum_{t=0}^{T} \beta^t u(\mathbf{x}_t, \chi_t)\right]$ for an initial state $\mathbf{x}_0 \in \mathcal{B}$. Here, $u(\cdot, \cdot)$ is the return function, and $\chi_t \in \Gamma(\mathbf{x}_t) \subset \mathcal{B}$, with $\Gamma(\mathbf{x}_t)$ denoting the set of feasible choices given $\mathbf{x}_t$. The discount factor $\beta \in (0, 1)$ weighs future returns. The DP approach aims to find period-specific policy functions p_t mapping the state $\mathbf{x}_t$ into the action χ_t, such that for all t, $\chi_t = p_t(\mathbf{x}_t) \in \Gamma(\mathbf{x}_t)$, and $\{\chi_t\}_{t=0}^{T}$ solves the original problem. The Bellman equation, which is consistent with the principle of optimality, can be expressed as $V_t(\mathbf{x}) = \max_\chi\{u(\mathbf{x}, \chi) + \beta\mathbb{E}[V_{t+1}(\tilde{\mathbf{x}})]\}$, where the successor state is distributed as $\tilde{\mathbf{x}} \sim \pi(\cdot|\mathbf{x}, \chi)$.

DP solves the problem recursively. Starting with a terminal value function for the terminal period $V_t(\cdot)$ for $t = T$, we solve the corresponding Bellman equation to obtain the previous period's value function values $\{v_{i,t-1}\}_{i=1}^{N}$, where $v_{i,t} = V_t(\mathbf{x}_{i,t})$, and controls $\{\chi_{i,t-1}\}_{i=1}^{N}$, for each point in a set of sampled state points $\{\mathbf{x}_{i,t-1}\}_{i=1}^{N}$. Iteratively, we compute the previous period's values v_{t-1} and controls χ_{t-1}, using a value function surrogate based on the state-value pairs from the previous iteration (details on how to compute the surrogate will be covered in Section 4.2). We continue this process until we have computed the value and policies for each period $\{\{v_{i,t}, \chi_{i,t-1}\}_{i=1}^{N}\}_{t=0}^{T-1}$.

Independent of the model specification, dynamic optimization problems are always accompanied by the same challenges when attempting to solve them numerically: efficiently approximating and interpolating high-dimensional value and policy functions on potentially irregularly shaped geometries. In the subsequent Sections 4.2 and 4.3, we describe how we tackle these challenges by combining GPRs with an approximation of the NTR.

4.2 Approximating functions with Gaussian process regressions

In the following, we briefly introduce GPR, a nonparametric regression method from supervised machine learning with universal approximation properties (see, *e.g.*, Micchelli et al., 2006) that we will use to approximate and interpolate multivariate policy and value functions (for more details, see, *e.g.*, Williams and Rasmussen, 2006; Murphy, 2012).

Consider a *training dataset* $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ that consists of N input states $\mathbf{x}_i \in \mathcal{B} \subset \mathbb{R}^d$ with corresponding observations $y_i \in \mathbb{R}$. The observations are assumed to be generated by an unknown function f, such that $y_i = f(\mathbf{x}_i) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. The matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ contains the *training inputs*, while $\mathbf{y} = [y_1, ..., y_N]^\top$ consists of the corresponding *targets* or *training inputs*. That training data can be generated using computer code or obtained from empirical data.[13] The goal is to infer a model of the unknown function f, enabling us to make predictions for y_* at a point $\mathbf{x}_*$ not present in the training set.[14] Recall from Section 4.1 that these predictions enable us to compute the future period's value of a given point in the state space.

In order to make predictions using the information in the dataset $\mathcal{D}$, we need to make assumptions about the characteristics of the underlying functions. A GP is a distribution over functions, defined by a mean function and a covariance function. The mean function, $\mu(\mathbf{x})$, represents the expected value of the function at input $\mathbf{x}$. The covariance function, also referred to as a covariance kernel $k(\mathbf{x}, \mathbf{x}')$, models the dependency between function values at different input points $\mathbf{x}$ and $\mathbf{x}'$ by computing a similarity (Williams and Rasmussen, 2006). A GP prior is assigned to the function f, representing our initial knowledge about it before observing any data:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{15}$$

Choosing an appropriate covariance function is crucial for GPRs and should be based on assumptions on smoothness and likely patterns in the data. A common assumption is that the correlation between two points decays with their distance. A popular choice of kernel that fulfills these assumptions is the Matern kernel,[15] defined as:

$$k_{\text{Matern}}(\mathbf{x}, \mathbf{x}'; \phi) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} d(\mathbf{x}, \mathbf{x}')\right)^\nu K_\nu \left(\sqrt{2\nu} d(\mathbf{x}, \mathbf{x}')\right), \tag{16}$$

[13]In our case, $\mathbf{x}_i$ is drawn from a simplex (*i.e.*, the state-space) and y_i is the solution to the Bellman equation.

[14]In the machine learning literature, *prediction* refers to interpolation.

[15]In our applications below (Secs. 5 and 6), we work with Matern 1.5 kernels. For more details on kernels, see, *e.g.*, Murphy (2023), Ch. 18.2, and `https://www.cs.toronto.edu/~duvenaud/cookbook`.

where $d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')(\mathbf{I}\boldsymbol{\ell})^{-2}(\mathbf{x} - \mathbf{x}')$ is the Euclidean distance scaled by the length scale $\boldsymbol{\ell}$, $K_\nu(\cdot)$ is a modified Bessel function, and $\Gamma(\cdot)$ is the gamma function. The kernel has *hyperparameters* $\phi = \{\nu, \boldsymbol{\ell}\}$, where ν is a smoothness parameter which takes the values $\{0.5, 1.5, 2.5\}$ and is chosen by the modeler and $\boldsymbol{\ell} = [\ell_1, ..., \ell_d]$ with $\ell_j > 0$ is the characteristic lengthscale of the j^{th} input dimension. The Matern kernel is particularly useful for modeling functions with abrupt changes and high non-linearity.

Once we have collected observations $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, by, for example, solving N Bellman equations at various locations within a set $\mathcal{B}$, our goal is to make predictions for new inputs stored in matrix $\mathbf{X}_*$ by drawing the corresponding $\mathbf{f}_*$ from the posterior distribution $p(f|\mathcal{D})$. The key GP assumption states that the observations $\mathbf{y}$ and the unobserved function values $\mathbf{f}_*$ jointly follow a multivariate normal distribution, which can be written as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) + \sigma_\varepsilon^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right), \tag{17}$$

where $k(\mathbf{A}, \mathbf{B})$ is the pairwise measure of the similarity between all points in matrix $\mathbf{A}$ and all points in matrix $\mathbf{B}$ (as given by the chosen kernel), $\mathbf{X}$ and $\mathbf{X}_*$ are the training inputs and out-of-sample observations, respectively, $\mathbf{I}$ is an identity matrix, and σ_ε^2 is the assumed noise level of observations (*i.e.*, the variance of ε). Standard results (see, *e.g.*, Williams and Rasmussen, 2006; Rasmussen and Nickisch, 2010) state that the conditional distribution of the unobserved component $p(\mathbf{f}_*|\mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ is a multivariate normal distribution, with mean function:

$$\tilde{\mu}(\mathbf{x}) = k(\mathbf{x}, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_\varepsilon^2 \mathbf{I}\right]^{-1} \mathbf{y}, \tag{18}$$

and a covariance:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_\varepsilon^2 \mathbf{I}\right]^{-1} k(\mathbf{X}, x'). \tag{19}$$

That is, to compute the predictive mean and covariance (Eqs. (18) and (19)), we must first calculate the four covariance matrices contained in Equation (17). Done naively, this step scales as $\mathcal{O}(N^3)$ with the number of observations N in the training set $\mathcal{D}$ (Williams and Rasmussen, 2006). In order to predict $\mathbf{f}_*$, we can simply use the mean function (Eq. (18)) evaluated it at the location of interest: $\tilde{\mu}(\mathbf{x})$. Eq. (19) additionally provides the *predictive variance* $\tilde{\sigma}^2(\mathbf{x}_*) := \tilde{k}(\mathbf{x}_*, \mathbf{x}_*)$, which can be used to derive point-wise predictive error bars and thus provide information about the model confidence at a point $\mathbf{x}_*$.

As it is crucial to minimize the training sample size, due to the poor algorithmic complexity of GPRs, the predictive variance (as computed by Eq. (19)), can be used to

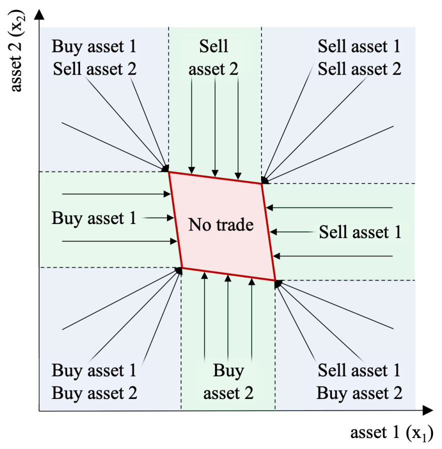


Figure 1: Schematic NTR (red parallelogram) in state space with corresponding $\boldsymbol{\delta}$-policy pattern (arrows). One (both) risky asset position(s) is (are) updated in the green (blue) regions, respectively.

sample points efficiently. Specifically, Bayesian Active Learning (BAL) (Houlsby et al., 2011) can be employed. BAL is a technique that strategically places observations in regions where they contribute the most to the approximator's quality. BAL utilizes the predictive variance to strategically place observations in regions where they contribute the most to the approximator's quality. By evaluating the predictive variance at different points in the state space, BAL identifies regions of higher uncertainty in the model's approximation, which indicates a need for additional information. Consequently, BAL focuses on acquiring new observations in these regions, leading to an optimized data selection process that balances the exploration of new areas in the state space with the exploitation of the current knowledge of the approximator. This efficient learning approach keeps the number of training samples small, mitigating the impact of the computational cost associated with GPR, while still providing an accurate approximation of the underlying function. For additional information on GPs, including their relevance in the era of deep learning and further discussion on scaling, please refer to Appendix D.1.

4.3 Characterizing the no-trade region

In the following, we briefly introduce the final missing component of our algorithm, that is, the approximation of the NTR. Consider Figure 1, which displays the optimal policy findings from previous literature for a portfolio optimization problem with two risky assets, as briefly discussed in Section 2. The parallelogram-shaped NTR (red region of the state space) signifies the area where wealth is not reallocated, should the portfolio weights fall within the said area. When portfolio weights fall outside of the NTR–indicated by the green and blue regions of the state space–they are updated to the boundary. The optimal policy of a point in a blue region of the state space is to update the portfolio weights to the nearest vertex of the NTR. The optimal policy of a point in a green region of the state space is to update the portfolio weights to the face of the NTR. These updates are done by buying and selling the respective assets. The optimal policies are represented by arrows in Figure 1. Concretely, consider a portfolio where asset 1's portfolio weights are low and asset 2's weights are high, that is, in the top left blue region in Figure 1. Should an agent hold this portfolio, the optimal behavior is to buy asset 1 and sell asset 2. After reallocation, the agent holds a portfolio whose weights lay on the top left vertex of the NTR in the state space.

We leverage the intuition presented in Figure 1 to design an efficient algorithm to approximate the NTR. Specifically, we make two assumptions to capture insights about the NTR's shape. For a dynamic portfolio optimization problem with D risky assets and proportional transaction costs, the assumptions read:

Assumption 1 *The NTR has 2^D vertices.*

Assumption 2 *The NTR is a D-dimensional convex polytope.*[16]

Assumption A1 states that, in settings with two risky assets, the NTR has $2^2 = 4$ vertices (as can be seen in Fig. 1).[17] By Assumption A2, the 2-dimensional NTR is then defined by connecting these vertices with straight lines. Note that these Assumptions do not enforce that the shape of the NTR is a strict parallelogram, but parallelogram-shaped NTRs satisfy both assumptions.[18] Furthermore, we use the previously discussed intuition regarding the optimal policy to approximate the NTR vertices, that is, as Figure 1 demonstrates, points in the blue regions of the state space update to the vertices of the NTR. Sampling a point in each of the blue regions (at least one point per region) allows us to approximate the NTR using A1 and A2.

In Section 4.3.1, we outline the exact steps that allow us to use Assumptions A1 and A2 to compute an approximation of the NTR in each DP iteration. Using this approximation, we then (1) determine the sign of the $\boldsymbol{\delta}$-policy and set the constraint

[16]A polytope is a geometric object with flat sides. More commonly, 2 and 3-dimensional polytopes are known as polygons and polyhedras, respectively. A convex polytope is a polytope where any linear combination of any points inside a polytope falls inside the polytope.

[17]Note that A1 suffers from the curse of dimensionality. For very high-dimensional state spaces, computing 2^D vertices is not feasible, however, moderately high dimensions (*e.g.*, $D = 10$ with $2^{10} = 1024$ vertices) do not pose significant challenges. See Appendix D.1 for more information.

[18]Similar assumptions have been used in existing research. Leland (2000) and Liu (2004) assume the NTR boundaries are straight lines (similar to A2). Muthuraman and Zha (2008) approximate a 2 to 7-dimensional hyper-polygonal NTR with intersecting hyperplanes, yielding similar results as if using A1 and A2. While much research does not make explicit assumptions, most still approximate NTRs that would satisfy our assumptions. For example, Lynch and Tan (2010) explicitly state that they have no *a priori* assumptions on the NTR but find, in line with the literature, that NTRs are "essentially" parallelograms or quadrilaterals. Similar findings have been made by Goodman and Ostrov (2010) and Muthuraman and Kumar (2006).

optimizer bounds dynamically, (2) sample state points efficiently, and (3) use multiple GPRs to map the value function in different regions of the state space. These steps, covering how we use the approximation of the NTR to facilitate the solving of the model, are outlined in Section 4.3.2.

4.3.1 Approximating the no-trade region

The intuition visualized in Figure 1 suggests that, in order to pin-point the vertices of an NTR, the optimization problem stated in Equation (10) must be solved for one state-space point in each blue region (in Fig. 1) to obtain the corresponding $\boldsymbol{\delta}$-policy. Therefore, we first sample a minimum of 2^D points $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_i\}_{i=1}^{2^D}$, one for each vertex, and compute each point's optimal policy $\{\hat{\hat{\boldsymbol{\delta}}}_i\}_{i=1}^{2^D}$. Then, we compute the NTR vertices $\{\hat{\boldsymbol{\omega}}_i\}_{i=1}^{2^D}$ where $\hat{\boldsymbol{\omega}}_i = \tilde{\mathbf{x}}_i + \hat{\hat{\boldsymbol{\delta}}}_i$. Using Assumption A2, we define the NTR as the convex hull around the vertices, that is, the NTR is defined as any linear combination of the vertices, $\hat{\boldsymbol{\Omega}}_t = \{\boldsymbol{\lambda}\hat{\boldsymbol{\omega}}_t \mid \boldsymbol{\lambda} \in (0,1)^N, \sum_{i=1}^N \boldsymbol{\lambda}_i = 1\}$. Algorithm 1 summarizes these steps.

Algorithm 1: Approximate the t^{th}-period NTR in the discrete-time finite-horizon portfolio choice model with proportional transaction costs.

```
  Input: t + 1st period's value function (surrogate) $\mathcal{V}_{t+1}$.
  Data: Set of $N = 2^D$ points $\tilde{\mathbf{X}}_t = \{\tilde{\mathbf{x}}_{i,t}\}_{i=1}^N \subset \mathcal{B}$.
  Result: Set of approximated NTR vertices: $\{\hat{\boldsymbol{\omega}}_{i,t}\}_{i=1}^N$; Approximated NTR: $\hat{\boldsymbol{\Omega}}_t$.
1 for $\tilde{\mathbf{x}}_{i,t} \in \tilde{\mathbf{X}}_t$ do
2     Obtain policy $\hat{\hat{\boldsymbol{\delta}}}_{i,t}$ for $\tilde{\mathbf{x}}_{i,t}$ by solving the optimization problem given by the
        Bellman equation (Eq. (10)) using $\mathcal{V}_{t+1}$ as next period's value function.
3     Compute the approximate NTR vertices $\hat{\boldsymbol{\omega}}_{i,t} = \tilde{\mathbf{x}}_{i,t} + \hat{\hat{\boldsymbol{\delta}}}_{i,t}$.
4 end
5 Compute the NTR approximation: $\hat{\boldsymbol{\Omega}}_t = \{\boldsymbol{\lambda}\hat{\boldsymbol{\omega}}_t \mid \boldsymbol{\lambda} \in (0,1)^N, \sum_{i=1}^N \boldsymbol{\lambda}_i = 1\}$.
```

For an unknown NTR, the location of the blue regions in Figure 1 are unknown *a priori*. Sampling points incorrectly leads to a high NTR approximation error. For example, a point selected from the green or red areas in Figure 1 would update to the face or inside of the NTR, respectively. A simple choice of points, where the points are likely to fall in the blue regions, are the vertices of the simplex, the origin, and the midpoints between each combination of simplex vertices. Figure 2 shows an illustrative

2-dimensional example. For a 2-dimensional state space, the $2^2 = 4$ green points are:[19]

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{bmatrix}.$$

The red points in Figure 2 are the corresponding NTR vertices computed using the approximated $\boldsymbol{\delta}$-policies, as represented by arrows. This choice of points allows for the correct detection of the NTR vertices in a large portion of the state space.[20] However, the sampling procedure can be updated should the modeler ascertain new information regarding the NTR's size and location in the state space.

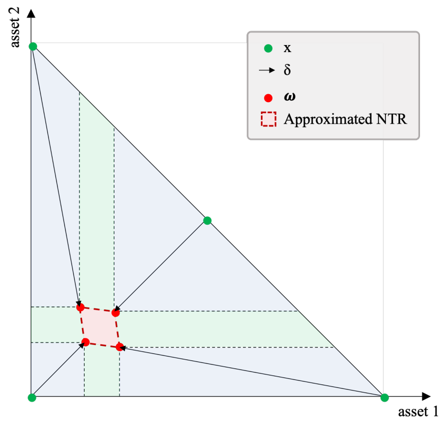


Figure 2: Using points sampled on the boundary of the simplex to determine the NTR vertices.

The feasibility of this approach hinges on our value function approximator being able to operate in non-hypercubic domains,[21] however, our general solution algorithm does not hinge on our NTR approximation method, as our method does not require the NTRs to be approximated using Assumptions A1 and A2. Our approach is motivated by our desire to solve high-dimensional portfolio choice problems where agents face proportional transaction costs. Our assumptions allow us to approximate the NTR in a way that presents a low computational burden. In settings with different transaction costs, or combinations of multiple transaction costs, and *a priori* unknown NTRs features, methods such as Gaussian mixture models (Murphy, 2012) may be better suited. They offer greater flexibility but come with additional computational overhead.

[19] Examples for higher-dimensional settings are provided in Appendix B.

[20] We discuss caveats to this sampling procedure in the Appendix C.

[21] To clarify, other methods (such as, *e.g.*, Schober et al., 2022; Cai et al., 2013) could use a similar approach to approximate the NTR; however, as their points are generated using a grid, they would be unable to draw similar benefits from the NTR approximation as we do (see Sec. 4.3.2).

4.3.2 Using the no-trade region approximation

There are a number of ways the NTR approximation facilitates the solving of our portfolio optimization problem. In the following, we discuss how to use the approximated NTR to determine the correct policy boundaries for the constraint optimizer, to sample points, and to discriminate between points inside and outside of the NTR. The latter allows us to train separate GPRs on different regions of the state space.

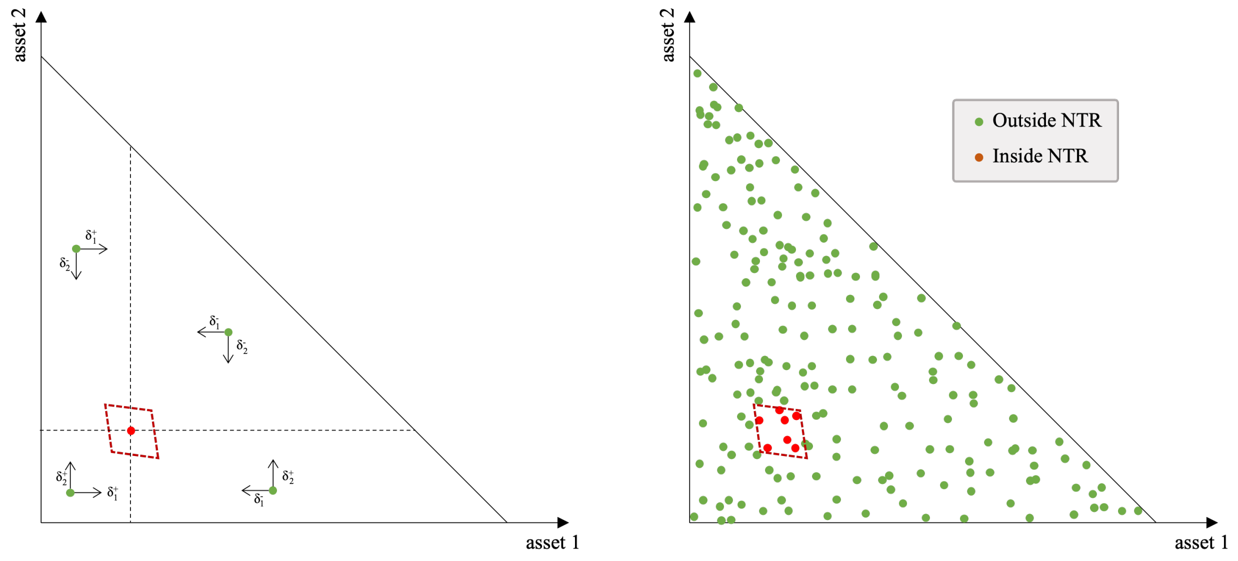
Determine $\boldsymbol{\delta}$-policy bounds: Recall from Section 3.3 that $\boldsymbol{\delta}$ was decomposed into its respective buying and selling components in order to better handle the kinks in the value functions induced by the proportional transaction costs, that is, we defined $\boldsymbol{\delta} = \boldsymbol{\delta}^{+} - \boldsymbol{\delta}^{-}$ with $\delta_i^+, \delta_i^- \geq 0$ for all $i \in [1, ..., D]$. A challenge that arises from this is how to enforce that the agent does not buy *and* sell the i^{th} asset simultaneously. A simple method is to guess which assets the agent buys and which they sell and to encode the problem so as to only allow this configuration. In the worst case, we must try every configuration of guesses (2^D combinations). The approximated NTR provides valuable information concerning the direction wealth that will be reallocated.

We first define a reference point using the approximated NTR vertices. If we assume A2, any linear combination of the NTR vertices lies inside the approximated NTR. Therefore, we compute the NTR mid-point by averaging the vertices. This ensures that the bounds for ambiguous points that have at least one asset which is not updated ($\delta_i = 0$) would also be correctly determined. Then, to determine the policy bounds, we simply compare the location of a state point to the said mid-point. If the point lies below the mid-point on a given dimension, we solve for δ_i^+. Otherwise, we solve for δ_i^-.[22] Figure 3a displays the intuition visually. The state space is segmented by the red mid-point (see dotted lines). The black arrows originating from a green state point indicate the respective δ_i directions for each segment.

Generating training points for the Gaussian processes: Standard joint-uniform sampling is a natural choice when computing global solutions to dynamic models (see, e.g., (Scheidegger and Bilionis, 2019), and references therein). However, in our context of dynamic portfolio choice problems with transaction costs, this strategy most likely undersamples points within the NTR as its volume is very small relative to the state space. This results in deteriorating the quality of the GPR approximation.[23] Furthermore, to counteract the curse of dimensionality as we scale the number of assets, having a minimal set of points that offers a reliable solution is crucial. As the

[22] We pass policy bounds $[0, 1]$ to the solver if we are solving for δ_i^+, and $[-1, 0]$ if we are solving for δ_i^-.

[23] The importance of data sampling is illustrated in Figure 19 (left column), which plots the cross-section of the value function predictions for low sample sizes (App. F). These results are discussed in Section 5.2.1.



(a) Use the NTR to determine the sign of the $\boldsymbol{\delta}$-policy. Each point in the same state-space segment (as defined by the mid point; dotted lines) are given the same policy bounds in the solver.

(b) GPRs fit the value function on different regions of the state space as defined by the approximated NTR. Two dataset are composed of red points inside (green points outside) of the NTR, respectively.

Figure 3: Using the approximated NTR to facilitate the process of solving the model. Panel (a) displays how to compute the sign of the $\boldsymbol{\delta}$-policy for state points in different regions of the state space; panel (b) discriminates points that fall inside and outside of the NTR, which are used as the respective training sets for different GPRs.

irregularities in the value and policy functions are a function of the NTR, we use the approximated NTR to sample points. There are several methods to do this, ranging from heuristic to algorithmic ones. Heuristic methods include oversampling the NTR and the kinks, while algorithm methods includes BAL (see Sec. 4.2) using the NTR to sample candidate points.

Multiple Gaussian process regressions: Finally, approximating the NTR allows us to use multiple GPRs to approximate different value function regions of the state space. As the NTR creates a kink in the value function around the boundary, we can use two GPRs to avoid having to approximate the kinks with smooth approximators. Instead, we fit one GPR with state points that lie inside the approximated NTR and one with points outside of the approximated NTR (see Fig. 3b). Formally, after solving the optimization problem for each point in the state space, we create two datasets:

$$\mathcal{D}_{\text{in},t} = \{(\mathbf{x}_{i,t}, \hat{v}_{i,t}) \mid (\mathbf{x}_{i,t}, \hat{v}_{i,t}) \text{ if } \mathbf{x}_{i,t} \in \hat{\boldsymbol{\Omega}}_t\}, \tag{20}$$

$$\mathcal{D}_{\text{out},t} = \{(\mathbf{x}_{i,t}, \hat{v}_{i,t}) \mid (\mathbf{x}_{i,t}, \hat{v}_{i,t}) \text{ if } \mathbf{x}_{i,t} \notin \hat{\boldsymbol{\Omega}}_t\}, \tag{21}$$

where $\mathbf{x}_{i,t} \in \mathbf{X}_t$ is the i^{th} state point, and $\hat{v}_{i,t}$ is the corresponding optimal value as computed by the solver. $\mathcal{D}_{\text{in},t}$ and $\mathcal{D}_{\text{out},t}$ are the datasets that consists of all points where the state is inside and outside of the approximated NTR $\hat{\boldsymbol{\Omega}}_t$, respectively. Then, in each period t, one GP is fit over each of the respective training sets. Then, we use the corresponding predictive means (*cf.* Eq. (18))

$$\tilde{\mu}(\tilde{\mathbf{x}}_t) = k(\tilde{\mathbf{x}}_t, \mathbf{X}_t) \left[k(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\varepsilon^2 \mathbf{I}\right]^{-1} \hat{\mathbf{v}}_t, \tag{22}$$

to approximate the value function for a $\tilde{\mathbf{x}}_t$ in the state space, where $\hat{\mathbf{v}}_t = [\hat{v}_{1,t}, ..., \hat{v}_{N,t}]^\top$. If $\tilde{\mathbf{x}}_t$ is inside of the approximated NTR ($\tilde{\mathbf{x}}_t \in \hat{\boldsymbol{\Omega}}_t$), then $\mathbf{X}_t$ and $\hat{\mathbf{v}}_t$ are composed of the points in $\mathcal{D}_{\text{in},t}$. Otherwise, if $\tilde{\mathbf{x}}_t$ is outside of the NTR ($\tilde{\mathbf{x}}_t \notin \hat{\boldsymbol{\Omega}}_t$), $\mathbf{X}_t$ and $\hat{\mathbf{v}}_t$ are composed of the points in $\mathcal{D}_{\text{out},t}$.

4.4 Dynamic programming with Gaussian process regressions and no-trade region approximations

The novel algorithm we propose for solving dynamic portfolio choice problems in discrete time combines DP, GPR's to approximate the value and policy functions, and an approximation of the NTR. Algorithm 2 summarizes the full solution method, which is outlined as follows. Starting from the terminal period T, we iterate backwards through time. For each DP iteration, we first define the value function surrogate used to complete the Bellman equation (Eq. (10)), that is, in period T, we use the terminal value function as specified by Equation (11). Then, we use Algorithm 1 to compute an approximation of the $t-1^{\text{st}}$ NTR $\hat{\boldsymbol{\Omega}}_{t-1}$. We do this by sampling 2^D points $\tilde{\mathbf{X}}_{t-1} = \{\tilde{\mathbf{x}}_{i,t-1}\}_{i=1}^{2^D}$. For each point in $\tilde{\mathbf{X}}_{t-1}$, we solve for the corresponding $\tilde{\boldsymbol{\delta}}_{t-1}$-policy, and use it to compute the vertices $\hat{\boldsymbol{\omega}}_{t-1}$. We use the approximated NTR $\hat{\boldsymbol{\Omega}}_{t-1}$ to sample N points from the state space $\mathbf{X}_{t-1}$ (as outlined in Sec. 4.3.2). For each point in $\mathbf{X}_{t-1}$, we solve for the optimal policy $\{\boldsymbol{\delta}_{i,t-1}, c_{i,t-1}\}_{i=1}^N$ and corresponding value $\{v_{i,t-1}\}_{i=1}^N$. Again, we do this by solving the recursive problem as defined by the Bellman equation (Eq. (10)) and laws of motion (Eqs. (7)–(9)). Next, we use the approximated NTR to discriminate between state-space points that fall inside and outside of the NTR, respectively, as defined by Equations (20) and (21). Finally, we train two GPRs to approximate the value functions by representing them with fully probabilistic GP models: one on the state points inside the NTR and one on those outside of the NTR.

This concludes the first iteration of our algorithm. We repeat these steps for the remaining iterations $t \in [T-1, ..., 1]$ using the previous GP approximations of the value function in the next iteration. Furthermore, as we train multiple GPs in each period (on points inside and outside of the NTR), we must determine whether the next period's state $\mathbf{x}_{i,t+1}$ (as given by Eq. (9)) falls inside or outside of the corresponding NTR approximation $\hat{\mathbf{\Omega}}_{t+1}$ and use the correct GP in the Bellman operator.

Algorithm 2: Dynamic programming with Gaussian process regressions and the NTR approximation for discrete-time finite-horizon portfolio optimization problems with proportional transaction costs.

```
   Input: Terminal value function $v_T$; time horizon $T$; sample size $N$.
   Result: Set of GP surrogates of the value functions $\{v_{t-1}\}_{t=0}^{T-1}$; set of approximated
           NTRs $\{\hat{\mathbf{\Omega}}_t\}_{t=0}^{T-1}$.
 1 Set $\mathcal{V}_T = v_T$ (Eq. (11)).
 2 for $t \in [T, ..., 1]$ do
 3     Approximate NTR $\hat{\mathbf{\Omega}}_{t-1}$ (Alg. 1) using $\mathcal{V}_t$ as the next period's value function.
 4     Sample $N$ points $\mathbf{X}_{t-1} = \{\mathbf{x}_{i,t-1}\}_{i=1}^{N} \subset \mathcal{B}$ from $D$-dimensional simplex.
 5     for $\mathbf{x}_{i,t-1} \in \mathbf{X}_{t-1}$ do
 6         Obtain value $\hat{v}_{i,t-1}$ and policy $\{\hat{\boldsymbol{\delta}}_{i,t-1}, \hat{c}_{i,t-1}\}$ for $\mathbf{x}_{i,t-1}$ by solving the
             optimization problem given by the Bellman equation (Eq. (10)) using $\mathcal{V}_t$ as
             the next period's value function.
 7     end
 8     Define the training sets:
 9         $\mathcal{D}_{\text{in},t-1} = \{(\mathbf{x}_{i,t-1}, \hat{v}_{i,t-1}) \ : \ \text{if } \mathbf{x}_{i,t-1} \in \hat{\mathbf{\Omega}}_{t-1}\}$,
10         $\mathcal{D}_{\text{out},t-1} = \{(\mathbf{x}_{i,t-1}, \hat{v}_{i,t-1}) \ : \ \text{if } \mathbf{x}_{i,t-1} \notin \hat{\mathbf{\Omega}}_{t-1}\}$.
11     Given $\mathcal{D}_{\text{in},t-1}$ and $\mathcal{D}_{\text{out},t-1}$, learn a surrogate of $v_{t-1}$ for inside and outside of the
         NTR $\{\mathcal{G}_{\text{in},t-1}, \mathcal{G}_{\text{out},t-1}\}$ (using the respective datasets) with GPs:
12     Set $\mathcal{V}_{t-1} = \{\mathcal{G}_{\text{in},t-1}, \mathcal{G}_{\text{out},t-1}\}$.
13 end
```

5 Verification of the solution algorithm

Our novel solution method allows us to perform a comprehensive evaluation of the correctness of the solution as well as to examine the implied behavior of an investor. To study the correctness, we follow the best practices of a sub-field of computational sciences called verification, validation, and uncertainty quantification (VVUQ; see, *e.g.*, Oberkampf and Roy, 2010, and references therein). VVUQ has to be an integral part of computational work as ours, as the models under consideration do not have analytical solutions, but instead have to be determined numerically, necessitating measures to assess the credibility and correctness of the derived results. While we aim to thoroughly assess the strengths of our solution method against existing research, the extent of VVUQ in previous studies is limited and does not provide the option for a rigorous comparison. Nevertheless, we provide an analysis in the hope that we may contribute to a standard for future research.

Parameter	Value
T	6
γ	3.5
β	0.97
R_f	4%
τ	1%

Table 1: Parameterization of the benchmart model by Schober et al. (2022).

To this end, we study two configurations of the benchmark model. First, we replicate the setting used by Schober et al. (2022), which enables us to draw a direct comparison between our work and the current state-of-the-art. Second, we introduce the configuration for a set of experiments that allow us to systematically study how economic factors like asset return, variance, and correlation influence the optimal policy and the NTR.

In what follows, we first outline these two calibrations in Section 5.1.[24] Then, in Section 5.2, we provide the results of the VVUQ experiments. We assess the quality of our solution using multiple metrics, including value function fit statistics, Euler errors, and an assessment of the quality of the approximation of the NTR.

5.1 Model calibration

To perform VVUQ on our solution method, we use two calibrations. The first anchors our research within established methodologies, while the second allows us to investigate the effect of various economic parameters on the optimal policy and the NTR. The initial calibration takes the calibration used by Schober et al. (2022), providing a direct comparison of the quality of our solution against existing benchmarks. Details of this calibration are provided in Section 5.1.1. Through our second calibration, we focus on understanding how economic factors such as asset return, variance, and correlation systematically shape policy outcomes and the NTR. To this end, we conduct a number of experiments relative to a simple baseline with identical. The specifics of parameterizations used in these experiments are elaborated in Section 5.1.2.

5.1.1 Baseline model parameterization

In this section, we briefly outline the baseline model parameterization, which allows us to draw a direct comparison with Schober et al. (2022). They solve a portfolio optimization problem with up to 5 risky assets where the investor (with a risk aversion of $\gamma = 3.5$) updates their portfolio once a year for $T = 6$ years, paying a transaction

[24]A technical calibration of our algorithm, including the parameterization of the GPs and solvers as well as an overview of the programming frameworks used, are outlined in Appendix D.

cost of $\tau = 1\%$ on each risky-asset transaction. The risk-free bond yields $R_f = 4\%$ annually. The entrire parameterization of the model is summarized in Table 1. The risky assets are described by their means $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$:

$$\boldsymbol{\mu} = \begin{pmatrix} 0.0572 \\ 0.0638 \\ 0.07 \\ 0.0764 \\ 0.0828 \end{pmatrix}, \quad \boldsymbol{\Sigma} = 10^{-2} \begin{pmatrix} 2.56 & 0.576 & 0.288 & 0.176 & 0.096 \\ 0.576 & 3.24 & 0.90432 & 1.0692 & 1.296 \\ 0.288 & 0.90432 & 4 & 1.32 & 1.68 \\ 0.176 & 1.0692 & 1.32 & 4.84 & 2.112 \\ 0.096 & 1.296 & 1.68 & 2.112 & 5.76 \end{pmatrix}.$$

When modeling fewer than five risky assets, we crop the dimensions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and use only the first D entries.

5.1.2 Stylized model parameterization

Next to the baseline setting discussed above, we also intend to study how the NTR changes with various parameters that have an economic impact, that is, relative to a parameterization with two identical risky assets with no correlation:

$$\boldsymbol{\mu} = \begin{pmatrix} 0.06 \\ 0.06 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 0.03 & 0.0 \\ 0.0 & 0.03 \end{pmatrix},$$

We run a series of experiments, each for $T = 10$ iterations, varying individual parameters. First, we keep both assets identical and vary one of the parameters for both assets simultaneously: $\mu_i \in [0.055, 0.065]$ and $\sigma_i \in [0.025, 0.035]$ for both $i = 1, 2$, $\rho \in [0.002, 0.005]$, and $\tau \in [0.005, 0.02]$. Then, we investigate how changing the discount factor, risk aversion, and the risk-free rate impacts the trajectory of the NTR over time: $\beta \in [0.95, 0.99]$, $\gamma \in [3.0, 4.0]$, and $R_f \in [0.03, 0.05]$. Finally, we run a number of experiments with non-identical assets where we create asymmetry in a single risky-asset parameter: $\mu_1 \neq \mu_2$, $\sigma_1 \neq \sigma_2$, or $\tau_1 \neq \tau_2$. We repeat these experiments for both zero ($\rho = 0.0$) and positive correlations ($\rho = 0.005$). The experiment parameters are summarized in Table 2. If otherwise unspecified, the remaining parameters are taken from Table 1.

5.2 Verification, validation, and uncertainty quantification results

Having the numerical experiments set up for our VVUQ exercise, we now assess three critical aspects of our solution: i) the fit of the value function, ii) the magnitude of the Euler errors, and iii) the accuracy of the NTR approximation. The fit of the value

Parameterization with identical assets			
1)	μ	0.055	0.065
2)	σ	0.025	0.035
3)	ρ	0.002	0.005
4)	τ	0.005	0.015
5)	β	0.95	0.99
6)	γ	3.0	4.0
7)	R_f	0.03	0.05
Parameterization with non-identical assets			
8)	$\boldsymbol{\mu}$		$\begin{pmatrix}0.065 & 0.06\end{pmatrix}^\top$
9)	$\boldsymbol{\sigma}$		$\begin{pmatrix}0.03 & 0.035\end{pmatrix}^\top$
10)	$\boldsymbol{\tau}$		$\begin{pmatrix}0.005 & 0.01\end{pmatrix}^\top$

Table 2: Parameter configuration for the stylized portfolio choice problems.

function and the Euler errors relate directly to how well the dynamic program is solved. Specifically, the Euler error provides a measure of how well the intertemporal problem was solved. However, since the Euler error is directly minimized by the constraint optimizer, evaluating the value function fit is necessary. A correctly solved problem is suggested by a combination of low out-of-sample Euler errors and high out-of-sample value function fit statistics. Similarly, an accurate NTR approximation is critical because inaccuracies can result in suboptimal behavior of the agent, hence limiting the algorithm's effectiveness.

We measure the quality of the fit for the value function using the relative absolute error (RAE) (Sec. 5.2.1) and evaluate the quality of the intertemporal solution with the relative Euler errors (REE) of consumption (Sec. 5.2.2). To evaluate the approximated NTR (Sec. 5.2.3), we first measure the squared error between the $\boldsymbol{\delta}$-policy computed by the constrained optimizer and $\boldsymbol{\delta}$-policy predicted using the approximated NTR. Then, we evaluate how the NTR approximation responds to changes in sample size and data-sampling protocol using the Hausdorff distance (Hausdorff, 1914). Importantly, all metrics were computed on the final DP iteration (*i.e.*, the 7th iteration), which enables us to gauge the overall quality of our solution, given that errors compound across iterations. The value function fit, as well as the Euler errors, are evaluated globally (*i.e.*, on the entire state space) and out-of-sample. To do this, we sample state points using a grid with points spaced 0.01 on each dimension to cover the feasible state space, resulting in a total of 5044 points. We use the value function surrogate computed during training to solve the dynamic program and obtain the optimal value and policies for each point on the grid. These values allow us to compare our GPR-based value function surrogate and to compute the Euler errors.[25] Furthermore, as we wish to

[25]It is worth noting that our evaluation process is computationally expensive given the need for a large

keep sample sizes small to manage the computational burden during the solving of the dynamic program, we sampled data joint-uniformly from the feasible state space and oversampled the NTR. Simple joint uniform sampling undersamples the NTR as the NTR is small relative to the state space, which is particularly problematic for small sample sizes.[26] We investigate whether additionally oversampling the approximated kinks induced by the NTR improves the various quality statistics. These metrics provide a comprehensive understanding of the solution's quality and identify potential areas of improvement.

In our VVUQ experiments, we use the model configuration presented in Section 5.1.1. This configuration allows us to effectively compare our results with existing state-of-the-art solution methods. We also run experiments to pinpoint the optimal training configuration for our algorithm by comparing various sample sizes ($N \in [100, 300, 500, 700]$) and data-sampling strategies for the 2-dimensional case. We find that the quality of the solution, as measured by the value function fit and Euler errors, generally improves as we increase N. However, there are diminishing returns of increasing the sample size. We compute the REEs for higher-dimensional settings with 3 and 5 risky assets, allowing us to draw a direct comparison to Schober et al. (2022). We demonstrate that the Assumptions A1 and A2 provide a good approximation of the NTR. Our ability to approximate the NTR is only moderately influenced by the sample size. Finally, regarding the data-sampling method, our results demonstrate that oversampling kinks provides a better fit and solution quality for small sample sizes and allows us to approximate the NTR more consistently across sample sizes.

5.2.1 Value function fit

We investigate now how well the GP approximates the value function. To evaluate the fit, we use the RAE, that is,

$$RAE(v, \hat{v}) = \frac{|v - \hat{v}|}{|v|},$$

where v is the true value function value for a given point in the state space as computed by the constraint optimizer and $\hat{v}$ is the corresponding prediction generated by the GP's predictive mean (Eq. (18)). Figure 4 plots the heatmaps of the out-of-sample RAE values in the state space in the final iteration of the experiment (iteration 7) for $N = [100, 300, 500, 700]$. Table 3 presents the corresponding summary statistics.[27]

number of points to approximate the RAE and REE statistics accurately. However, it is crucial to avoid over-assessing the results' accuracy by using fewer points in the post-processing step.

[26]Evidence supporting these claims can be found in Appendix F (Fig. 19).

[27]Note, all plots and statistics are based on a single draw of data points and are not averaged over multiple draws.

Figure 4 shows low RAE values across the state space with areas of higher errors accruing locally. These areas are seen to concentrate where there are few training points (*e.g.*, second panel of Fig. 4 near the simplex boundary) and along kinks (*e.g.*, horizontally and vertically organized error distributions in the fourth panel of Fig. 4). The results show that increasing the sample size improves the value function fit; however, the summary statistics show that satisfactory errors are achieved at all sample sizes of $N > 100$. Furthermore, results in Appendix F demonstrate that as approximation errors accrue near kinks, oversampling the approximated kinks outperforms just oversampling the NTR for small sample sizes (see Tab. 12 and Fig. 20). However, once N is sufficiently large, oversampling the kinks becomes redundant. Hence, oversampling kinks can be used to keep the sample size low, as long as this does not come at the cost of the overall quality of the approximation. Note that direct comparison with Schober et al. (2022) is not possible as they do not directly quantify their value function approximation errors. A full comparison of the RAE values for all sample sizes and RAE summary statistics across sample sizes for different data-sampling protocols can be found in Appendix F (*cf.* Fig. 21 and 22, respectively).

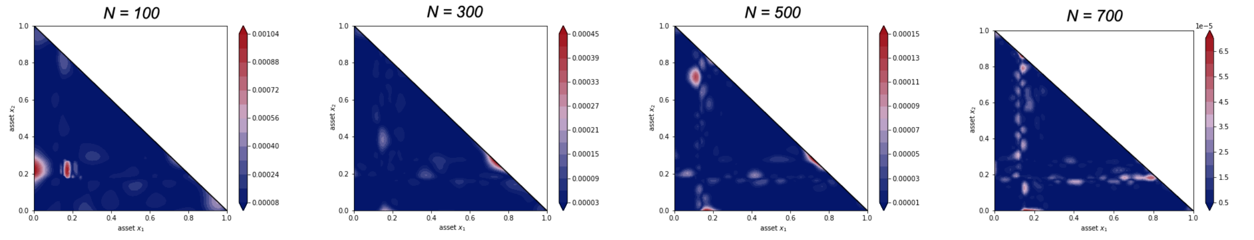


Figure 4: Out-of-sample RAE value function fit heatmap. The RAE values were computed using solutions from a model where the NTR was oversampled.

N	Mean (%)	99.9th percentile (%)	Max (%)
100	0.0070	0.101	0.110
300	0.0022	0.0372	0.0456
500	0.0007	0.0137	0.0159
700	0.0004	0.0061	0.0070

Table 3: Out-of-sample RAE value function fit summary statistics (in percent) of the final iteration (iteration 7) for various sample sizes. The summary statistics were computed using solutions from a model where the NTR was oversampled.

5.2.2 Euler errors

Euler errors provide a measure of how well the intertemporal problem is solved. As mentioned above, the gradient of the Bellman is used by the solver during optimization. Hence the Euler errors should only be large if floating-point precision is low (*i.e.*, single precision as opposed to double precision) or if the value function fit is poor (however, even here, we can have erroneously low errors). In the previous section, we demonstrated our method's ability to approximate the value function with low out-of-sample RAEs. In this section, we will investigate whether the out-of-sample Euler errors display similar performance.

Our portfolio optimization problem has two sets of Euler errors that need to be optimized: the consumption Euler error, $\varepsilon_{c_t}(\mathbf{x}_t)$, and the portfolio-reallocation Euler error, $\varepsilon_{\delta_{i,t}}(\mathbf{x}_t)$:[28]

$$\varepsilon_{c_t}(\mathbf{x}_t) = -c_t^{-\gamma} + \beta\mathbb{E}_t\left[R_f\pi_{t+1}^{-\gamma}\left((1-\gamma)v_{t+1}(\mathbf{x}_{t+1}) - \nabla_{\mathbf{x}_{t+1}}v_{t+1}(\mathbf{x}_{t+1})\mathbf{x}_{t+1}\right)\right] = 0,$$
$$\varepsilon_{\delta_{i,t}}(\mathbf{x}_t) = c_t^{-\gamma}\frac{\partial c_t}{\partial\delta_{i,t}} + \beta\mathbb{E}_t\left[R_{i,t}\pi_{t+1}^{-\gamma}\left((1-\gamma)v_{t+1}(\mathbf{x}_{t+1}) - \nabla_{\mathbf{x}_{t+1}}v_{t+1}(\mathbf{x}_{t+1})(\mathbf{e}_i - \mathbf{x}_{t+1})\right)\right] = 0,$$

where

$$\frac{\partial c_t}{\partial\delta_{i,t}} = \begin{cases} -(1+\tau) & \text{if } \delta_{i,t} > 0 \\ -(1-\tau) & \text{if } \delta_{i,t} < 0 \end{cases}, \tag{23}$$

and $\mathbf{e}_i$ is a i-dimensional vector of zeros with a 1 in the i^{th} position.[29]

The Euler errors relative to consumption ($\tilde{\varepsilon}(\mathbf{x}_t)$) are defined as:

$$\tilde{\varepsilon}_{c_t}(\mathbf{x}_t) = \frac{\beta\mathbb{E}_t\left[R_f\pi_{t+1}^{-\gamma}\left((1-\gamma)v_{t+1}(\mathbf{x}_{t+1}) - \nabla_{\mathbf{x}_{t+1}}v_{t+1}(\mathbf{x}_{t+1})\mathbf{x}_{t+1}\right)\right]^{-\frac{1}{\gamma}}}{c_t} - 1 = 0,$$
$$\tilde{\varepsilon}_{\delta_{i,t}}(\mathbf{x}_t) = \frac{\left[\frac{\beta\mathbb{E}_t\left[R_{i,t}\pi_{t+1}^{-\gamma}\left((1-\gamma)v_{t+1}(\mathbf{x}_{t+1}) - \nabla_{\mathbf{x}_{t+1}}v_{t+1}(\mathbf{x}_{t+1})(\mathbf{e}_i - \mathbf{x}_{t+1})\right)\right]}{-\frac{\partial c_t}{\partial\delta_{i,t}}}\right]^{-\frac{1}{\gamma}}}{c_t} - 1 = 0,$$

and refer to these as REEs.

We compute the out-of-sample REEs for $N = 500$ and the final iteration with the same data-sampling protocol as previously, that is, the data was sampled joint-uniformly from the feasible state space, and the approximated NTR was oversampled. We summarize the REE statistics in Table 4.[30] Recall that the gradient with respect

[28]Derivations can be found in Appendix E.

[29]Note that the gradient given by Equation (23) is not defined when $\delta_{i,t} = 0$. In the implementation, we compute the Euler errors for non-updates $\delta_{i,t} = 0$ by using the gradient corresponding to the decomposed portfolio reallocation, $\delta_{i,t}^{+}$ and $\delta_{i,t}^{-}$, suggested by the position of the state point in relation to the approximated NTR (see Section 4.3.1). $\partial c_t/\partial\delta_{i,t} = -(1+\tau)$ if $\delta_{i,t}^{+} \geq 0$ and $\delta_{i,t}^{-} = 0$, and $\partial c_t/\partial\delta_{i,t} = -(1-\tau)$ if $\delta_{i,t}^{+} = 0$ and $\delta_{i,t}^{-} \geq 0$.

[30]See Appendix F, Table 13, for the summary statistics for sample sizes $N = [300, 500, 700]$ and for results where the model was trained on data that also oversampled the approximated kinks. These were omitted from the main text as the sample size had a negligible effect on the REE results. The REE results are visualized in Figures 23 and 24, which plot the REE by policy and sample size, respectively. Note that these

to δ_i is not defined at $\delta_i = 0$ and hence the Euler errors are not defined. Therefore, in Table 4, the cells reporting the REEs with respect to $\boldsymbol{\delta}$ contain two values: the global REE statistics and, in brackets, the REE statistics that were computed by omitting observations where $\delta_i = 0$ for $i \in [1, 2]$.

The results in Table 13 demonstrate that the intertemporal problem is solved well. The $\boldsymbol{\delta}$ REEs are low and indicate that the investor misallocates around 0.02% of her wealth on average and approximately 0.3% in the max. If we omit the regions where the Euler errors are not defined, the statistics drop significantly, implying an investor misallocates only approximately 5.0×10^{-6}% of her wealth on average. The max errors are also low, suggesting an approximate 0.001% misallocation. These values are not reported by other studies and hence cannot be compared to the literature. The REEs with respect to c are low. On average and in the max, the investor only misallocates 1.4×10^{-8}% and 5.3×10^{-7}% of her wealth, respectively. These REE values with respect to c are an improvement on the Euler errors reported in contemporary research.[31]

	Mean	99.9th percentile	Max
δ_1 (%)	0.0222 (4.6×10^{-6})	0.278 (0.00079)	0.334 (0.00099)
δ_2 (%)	0.0179 (4.7×10^{-6})	0.274 (0.00054)	0.332 (0.00111)
c (%)	1.4×10^{-8}	4.7×10^{-7}	5.3×10^{-7}

Note: Values in brackets are computed by omitting observations where $\delta_i = 0$ for $i \in [1, 2]$. As the gradient w.r.t. δ is not defined at $\delta_i = 0$, the Euler error is not defined.

Table 4: Out-of-sample REE summary statistics (in percent) of the final iteration (iteration 7) for $N = 500$. The summary statistics were computed using solutions from a model where the NTR was oversampled.

Table 5 extends this analysis to higher-dimensional settings and presents the REEs for the 2, 3, and 5 risky-asset settings for the final iteration (iteration 7). We intend for this calibration to offer a proof-of-concept and for it to allow us to compare our results to the state-of-the-art.[32] The 2, 3, and 5 risky-asset cases were solved using 500, 700, and 2000 points, respectively. Again, the models were trained on points sampled from a joint-uniform distribution with an oversampled NTR approximation. The number of points needed to solve the dynamic portfolio optimization problem with the proposed method is significantly lower than competing literature.[33]

The values in Table 5 demonstrate that the REE levels are maintained as the problem is scaled to higher dimensions. The mean (max) consumption errors remain

figures were plotted to enable a direct comparison to Schober et al. (2022).

[31]Additional results contained in Appendix F demonstrate that, unlike the value function fit statistics, the REE results are robust to the sample size (see Tab. 13). However, oversampling the approximated kinks, in addition to the approximated NTR, does yield a slight improvement, again identifying the kinks as a main driver of the errors (see Tab. 13).

[32]In both the three and the five risky-asset cases, the new assets the agent can trade have the following properties: $\mu_5 > \mu_4 > \mu_3 > \mu_2 > \mu_1$, and $\sigma_5 > \sigma_4 > \sigma_3 > \sigma_2 > \sigma_1$, with $\rho_{1,4}, \rho_{1,5}, \rho_{2,4}, \rho_{2,5}, \rho_{3,4}, \rho_{3,5}, \rho_{4,5} \neq 0$. Therefore, we are unable to perform a systematic comparison of the effect of increasing the number of assets an agent has access to.

[33]For a note on performance, see Appendix D.4.

of the order of $10^{-6}\%$ ($10^{-5}\%$) or below throughout all experiments. The $\boldsymbol{\delta}$-specific REEs are slightly higher, remaining of the order of $10^{-2}\%$ ($10^{-1}\%$) or below. Again, the reported REEs are global errors, that is, the values are inflated by including the regions where the Euler error is not defined at $\delta_i = 0$.

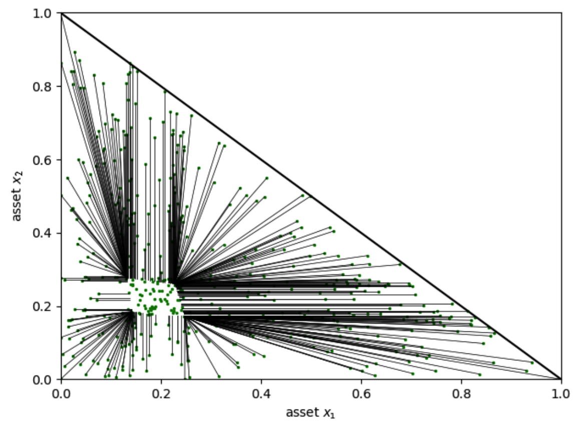
	2 risky assets		3 risky assets		5 risky assets	
N	500		700		2000	
REE (%)	mean	max	mean	max	mean	max
δ_1	0.0222	0.334	0.0341	0.2832	0.0377	0.2847
δ_2	0.0179	0.332	0.0308	0.2817	0.0306	0.3239
δ_3			0.0270	0.2833	0.0299	0.3831
δ_4					0.0294	0.2858
δ_5					0.0334	0.4485
c	1.0×10^{-8}	5.3×10^{-7}	1.0×10^{-8}	3.4×10^{-8}	4.4×10^{-6}	8.6×10^{-5}

Table 5: REE statistics for a high-dimensional portfolio optimization problem for final iteration.

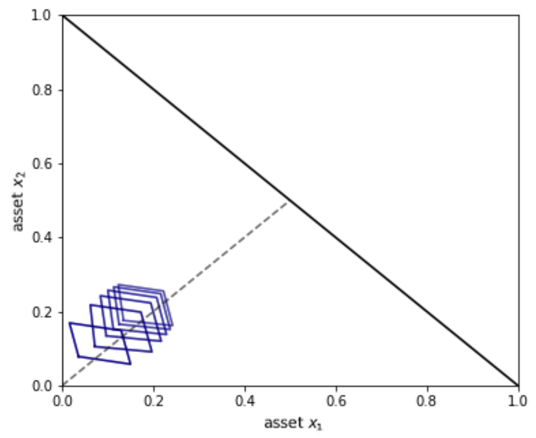
5.2.3 No-trade region approximation quality

The key difference in a setting with proportional transaction costs versus one without proportional transaction costs is the emergence of an NTR. Figure 5 visualizes the agent's optimal behavior for the final iteration (Panel 5a) and across time (Panel 5b). The former displays, for any (green) point in the state space, an agent's optimal $\boldsymbol{\delta}$-policy represented by the black line. The area where the portfolio is not updated (*i.e.*, points without a corresponding black line) makes up the NTR. The figure demonstrates that the optimal policy for state space points outside of the NTR is for the agent to reallocate her wealth to a point on the boundary of the NTR. Figure 5b plots the approximated NTRs for all periods ($t \in [0, ..., T-1]$) and shows that over time, the NTR moves towards the origin (down and left). Furthermore, the plot visualizes that the NTRs are approximately parallelogram-shaped in each period. The up-most right NTR in Figure 5b corresponds to the implied NTR of Figure 5a. In this section, we perform two experiments: first, we determine whether the NTR as approximated by Assumptions A1 and A2 offers a good approximation; second, we investigate how the choice of sample size and data-sampling protocol affects the approximation of the NTR.

First, we evaluate Assumptions A1 and A2. We do this by computing the mean squared error between the $\boldsymbol{\delta}$-policy predictions implied by the approximated NTR to the $\boldsymbol{\delta}$-policy generated using the constraint optimizer, generating predictions using the approximated NTR by updating the state points to the NTR boundary (see Sec. 4.3.1). We plot the errors in Figure 6a and present the results for all iterations in Table 6. The results display low mean and max squared errors for all periods, suggesting the polygon approximation based on the approximated vertices is sufficient. This provides



(a) NTR forms from $\boldsymbol{\delta}$-policy (black line) as computed by constraint optimizer for each (green) point in the state space (iteration 7).



(b) Trajectory of approximated NTRs over time. NTRs move towards the origin as time approaches the terminal time.

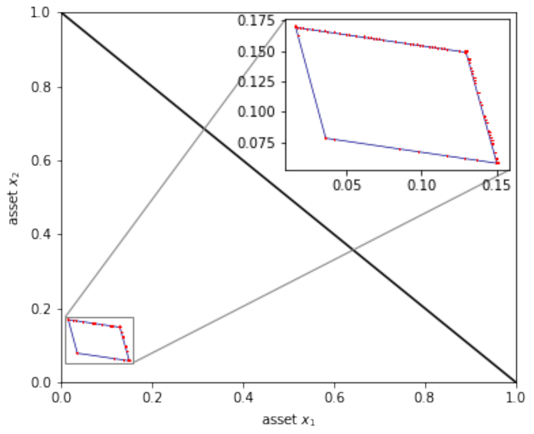
Figure 5: $\boldsymbol{\delta}$-policy and approximated NTR results using the parameterization by Schober et al. (2022) (cf. Sec. 5.1.1 of this paper).

legitimacy for our approach of discriminating the state space into regions within and outside of the NTR and allows us to conduct an ex-post analysis on the approximation of the NTR without loosing generality.

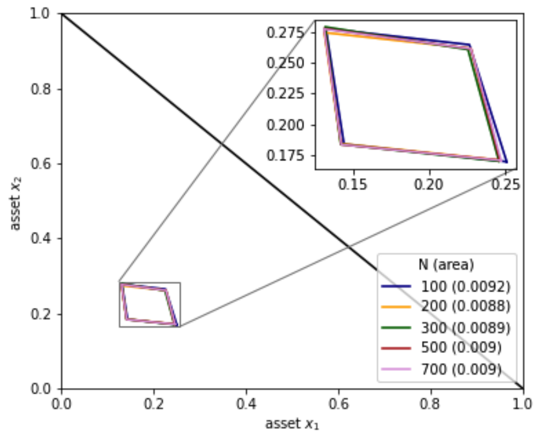
Second, we evaluate how sample size impacts the quality of the NTR approximation. Figure 6b plots how the NTR approximation changes as we increase the sample size $N = [100, 200, 300, 500, 700]$ for the final iteration (iteration 7). Visually, the difference in the approximated NTR for large sample sizes of $N \geq 300$ is small. The $N = 100$ NTR visually deviates from the NTRs approximated with larger sample sizes. This finding is supported by the values in Table 7 where we compare two NTRs using the Hausdorff distance:

$$d_H(\hat{\boldsymbol{\Omega}}_N, \hat{\boldsymbol{\Omega}}_M) = \max\left\{ \sup_{x \in \hat{\boldsymbol{\Omega}}_N} d(x, \hat{\boldsymbol{\Omega}}_M), \sup_{y \in \hat{\boldsymbol{\Omega}}_M} d(\hat{\boldsymbol{\Omega}}_N, y) \right\}, \tag{24}$$

where sup represents the supremum, inf represents the infimum, $d(a, B) = \inf_{b \in B} d(a, b)$ measures the distance from a point $a \in A$ to the set B, and M and N denote the sample sizes used to approximate the respective NTR. These values demonstrate that approximated NTRs using $N \geq 100$ all yield similar results. Furthermore, results in Appendix F demonstrate that oversampling the approximated kinks results in more consistent NTR approximations across sample sizes (see Fig. 26 and Tab. 14).



(a) NTR approximation error between computed policy and policy implied by approximated NTR.



(b) Comparing the final iteration's NTR computed using various sample sizes.

Figure 6: Evaluation of the quality of the NTR approximation.

t	Mean squared error (%)	Max squared error (%)
0	0.00015	0.00075
1	0.00015	0.00068
2	0.00012	0.00060
3	0.00017	0.00079
4	0.00011	0.00055
5	0.00009	0.00046
6	0.00005	0.00022

Table 6: Mean and max squared errors between optimal $\boldsymbol{\delta}$-policy as computed by the constraint optimizer and the predicted $\boldsymbol{\delta}$-policy as implied by approximated NTR.

6 Results

Our method enables the solution of high-dimensional portfolio optimization problems that were previously challenging or infeasible to solve. Our objective is to explore the economic implications of the solutions provided by our proposed model. To this end, we conduct two sets of experiments. First, we analyze how the shape of the NTR is determined by the various economic parameters present in the model specification. Following the existing literature, we focus on two risky assets, as this configuration facilitates visualizing changes to the NTR. While corroborating existing findings, for exam, from Dybvig and Pezzo (2020), we further investigate how previously unstudied economic model parameters influence the NTR and the trajectory of the NTR over time. Lastly, our analysis of the NTR concludes with an examination of the sizes of higher-dimensional NTRs relative to the size of the state space. Second, we employ Monte Carlo simulations to provide insights into the agent's behavior when trading in the presence of proportional transaction costs. Here, we offer a novel approach to

N	100	200	300	500	700
100	0.0				
200	0.0052	0.0			
300	0.0051	0.0027	0.0		
500	0.0047	0.0020	0.0019	0.0	
700	0.0038	0.0022	0.0027	0.0018	0.0

Table 7: Hausdorff distance between NTRs across sample size computed using solutions from a model where the NTR was oversampled.

quantify the liquidity premium—a metric that measures how much the agent is willing to pay to avoid paying transaction costs. Furthermore, as transaction costs deter potentially optimal trades, we compute the decline in expected lifetime utility. Finally, we evaluate the potential utility loss from utilizing a suboptimal trading strategy that is easier to compute, demonstrating our method's flexibility.These experiments hold significant importance as they unveil economically meaningful patterns and behaviors. Despite being largely unable to make systematic statements about the effect of illiquidity as the baseline parameterization was not designed to conduct a systemaic analysis, we do find trends that suggest the effects may be reduced when agents are given access to more risky assets. To detail our findings, we proceed in two steps: Section 6.1 outlines our study of the NTR, while Section 6.2 explores the Monte Carlo simulation results.

6.1 A Systematic investigation of the no-trade region

As discussed in Section 5.2.3, the NTR is approximately a parallelogram NTR when modeling two risky assets. Figure 5b shows that, over time, the NTR moves towards the origin of the state space. The agent holds less of her wealth in risky assets as the terminal period approaches. The optimal policy dictates that the agent updates her portfolio holdings to the boundary of the NTR if her holding lies outside the NTR and does not update if the holdings fall inside the NTR (see Fig. 5a). Furthermore, we demonstrated that the polygon approximation based on Assumptions A1 and A2 provides a good approximation. In this section, we extend the analysis of the NTR and investigate how various optimization problem components affect the NTR's shape. While many of the results confirm the work of previous studies (*e.g.*, Dybvig and Pezzo (2020)), we provide more depth concerning the trajectory of the NTR across time and how the shape of the NTR changes relative to non-asset economic parameters, such as the agent's risk aversion γ or the risk-free rate R_f. The experiments in this section are based on the stylized model parameterization (see Sec. 5.1.2 and Tab. 2) to facilitate a systematic comparison. We use the baseline calibration to investigate the NTR in higher dimensions (see Sec. 5.1.1).

Figures 7–10 plot the NTRs for Experiments 1–4 (cf. Tab. 2), respectively. Each figure is composed of 3 panels, plotting the NTR for $t = 0, 4$, and 9 from left to right, respectively. These periods correspond to the DP iterations $1, 5,$, and 10, respectively. Figures 11 and 12 plot the full set of NTRs for $t = [0, ...9]$ for Experiments 8 and 9 (Tab. 2) with and without correlation, that is, the left panel of both figures shows the NTRs with identical risky assets. The middle and right panels plot the NTRs for Experiments 8 and 9, respectively. Figure 13 plots the NTRs for Experiment 10 (Tab. 2). The NTRs in Figures 7, 8, 10, 11, and 13 are approximated using risky assets with no correlation ($\rho = 0$) and hence yield square NTRs. Figures 9 and 11 are approximated using risky assets with correlation ($\rho > 0$) yielding NTRs with the shape of a parallelogram.

Experiments 1–3 (Tab. 2) allow us to investigate the effect of changing asset-specific parameters on the shape of the NTR. We begin with identical assets without correlation and simultaneously change individual parameters for both assets. Figure 7 depicts the NTRs when we increase and decrease the return of both assets (μ_1, μ_2) in tandem (Tab. 2, Ex. 1). The NTRs are square as the agent is indifferent between the two identical assets. Relative to the blue baseline NTR, an increase (decrease) in the return shifts the NTR upwards towards a higher relative allocation (downwards, lower relative allocation) along the diagonal, respectively. Holding volatility constant, high-return assets are more attractive to the agent. The shift along the diagonal relative to the baseline is more significant for smaller t (*i.e.*, earlier years) because the agent has a higher ability to absorb risk as the majority of their life remains ahead. Figure 8 plots the results when we increase and decrease the variance of both assets (σ_1, σ_2) in tandem (Tab. 2, Ex. 2). Relative to the blue baseline NTR, an increase (decrease) in the variance decreases (increases) the area of the NTR, that is, more risk decreases the area where the agent is inactive as they are reluctant to hold high allocations of the risky assets. The size difference persists over time. Furthermore, the difference between the three NTRs is most pronounced when t is small. In early periods, the agent favors low risky asset allocations as the risk-averse agent is less willing to hold risky assets. Figure 9 plots the results when we increase the correlation of the assets (ρ). Relative to the blue baseline NTR, a correlation increase changes the shape of the NTR from a rectangle to a parallelogram as the stocks become substitutes. That is, a trade is less likely to occur if an agent were to over-weight one asset and underweight the other. Correlation is responsible for the NTR's parallelogram shape observed in the baseline calibration (*cf.* Fig. 5b). This shape persists over time.

Figure 10 corresponds to Experiment 4 (Tab. 2) and plots the results when we increase and decrease the transaction cost of both assets (τ) in tandem. Decreasing the transaction cost results in a smaller NTR that shrinks around the no-transaction-cost optimum. The agent becomes less willing to hold allocations that deviate more strongly from the no-transaction-cost optimum. For earlier periods (small t), increasing the transaction cost has a smaller effect on the NTR area than in later periods (large t), as the agent is more capable of coping with the effect of paying transaction costs in earlier periods.

Figure 11 depicts, from left to right, NTRs across time for the baseline identical-asset parameterization, assets with different returns and equal variances ($\mu_1 < \mu_2$; Ex. 8), and assets with equal returns and different variances ($\sigma_1 < \sigma_2$; Ex. 9), respectively. The primary trend in the left panel shows that as t increases, the NTR moves towards lower relative asset holdings (down and left). The trajectory through the state space for identical assets follows the 45° line as the agent is indifferent between the two risky assets. The middle panel plots NTRs shifted off the 45° line in the direction that favors asset 1; risky asset with a higher return. The right panel plots NTRs with a reduced size along the axis of the asset with a higher variance relative to the baseline calibration. The agent is more likely to trade when exposed to more risk through a higher holding of asset 2. Moving backward through time to small t, the NTR path deviates from the 45° line, favoring asset 1; the asset with lower risk. Figure 12 performs a similar exercise with assets with a positive correlation. The findings are identical to those in Figure 11, except that the NTR is a parallelogram rather than rectangular. Finally, Figure 13 plots the NTRs for assets with differing transaction costs, $\tau_1 < \tau_2$ (Ex. 10). Only the shape of the NTR is affected as the agent is more willing to trade the lower-cost asset (*i.e.*, asset 1). The trajectory of the top-right vertex follows the 45° degree line.

Finally, we investigate how the NTR's trajectory is affected when non-asset economic parameters are changed. Figure 14 plots, from left to right, the NTR across time for the identical-asset calibration with no correlation when the discount factor β (Ex. 5), risk aversion γ (Ex. 6), or the risk-free rate R_f (Ex. 7) is changed, respectively. Changing the discount factor β does not have a visible effect on an agent's optimal portfolio choice. Increasing (decreasing) the risk aversion γ (middle panel) increases (decreases) an agent's preferences for risk, and hence shifts the NTR upwards (downwards) along the 45° line, that is, an agent who has a higher tolerance for risk is more willing to hold higher allocations of risky assets. Furthermore, increasing the risk aversion also increases the area the NTRs span across time, reflecting that the agent's tolerance for risk accumulates over time. Finally, decreasing the risk-free rate R_f makes risky assets relatively more appealing to a risk-averse agent and hence, shifts the NTRs towards higher allocations along the 45° degree line.

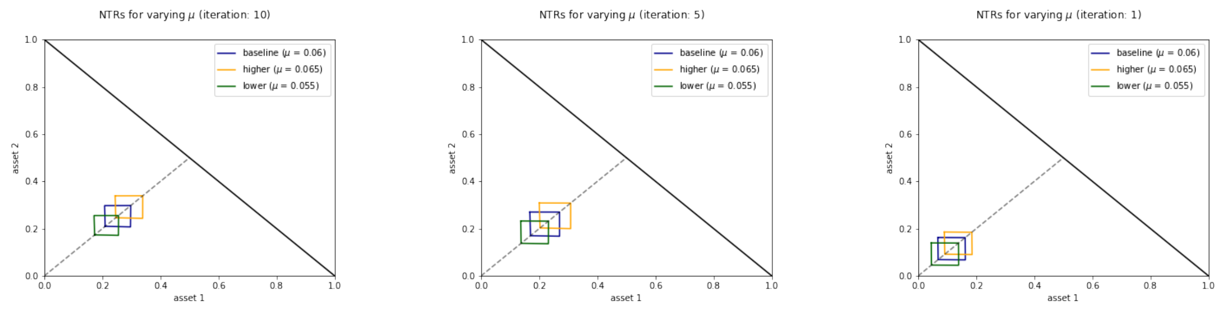


Figure 7: NTR over time for identical assets and varying risky asset mean values (μ).

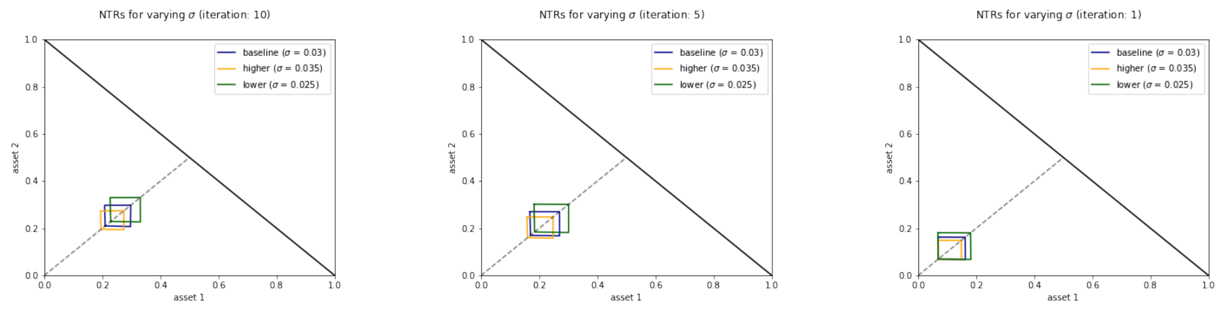


Figure 8: NTR over time for identical assets and varying risky asset variance values (σ).

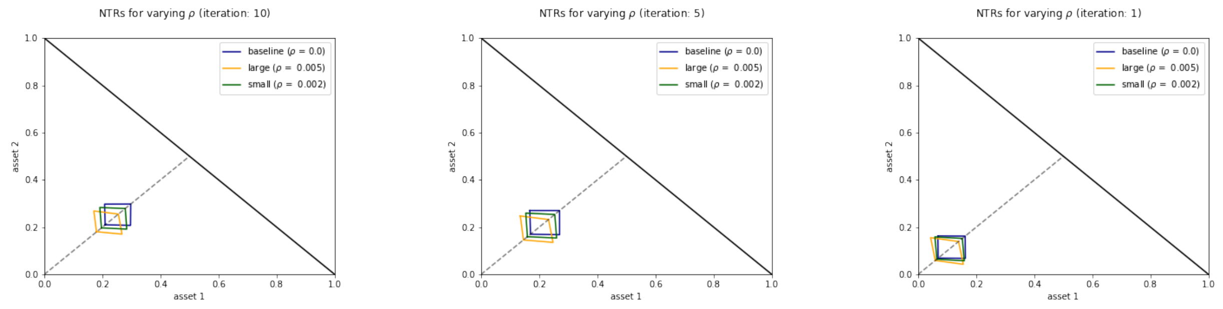


Figure 9: NTR over time for identical assets and varying risky asset correlation values (ρ).

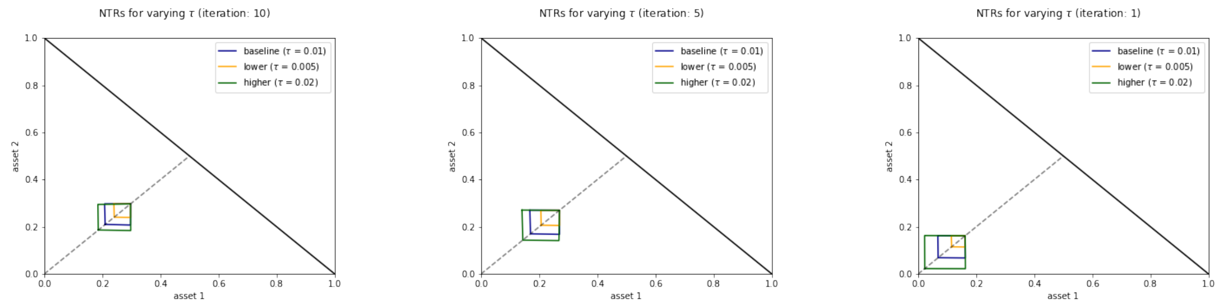


Figure 10: NTR over time for identical assets and varying risky asset transaction costs (τ).

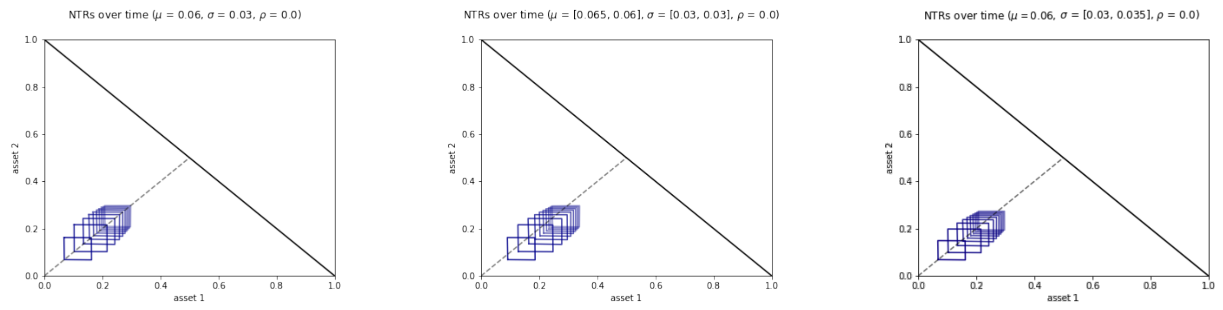


Figure 11: NTRs over time for various asset configurations without correlation ($\rho = 0.0$). Left: identical assets; middle: non-identical assets with $\mu_1 \neq \mu_2$; right: non-identical assets with $\sigma_1 \neq \sigma_2$.

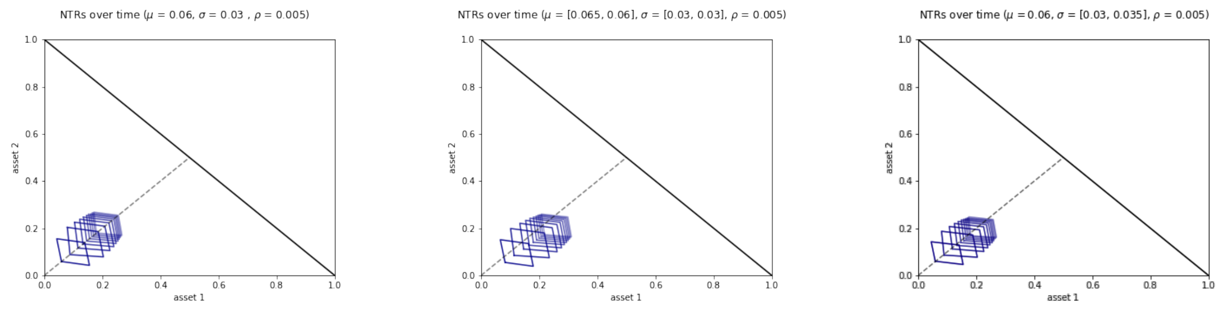


Figure 12: NTRs over time for various asset configurations with positive correlation ($\rho = 0.005$). Left: identical assets; middle: non-identical assets with $\mu_1 \neq \mu_2$; right: non-identical assets with $\sigma_1 \neq \sigma_2$.

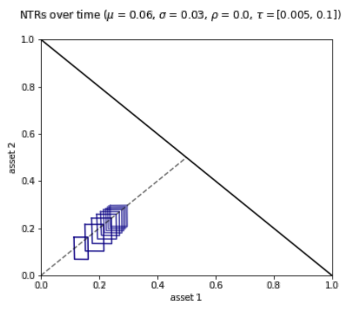


Figure 13: NTRs over time for identical assets with differing transaction costs with $\tau_1 \neq \tau_2$ and no correlation ($\rho = 0.0$).

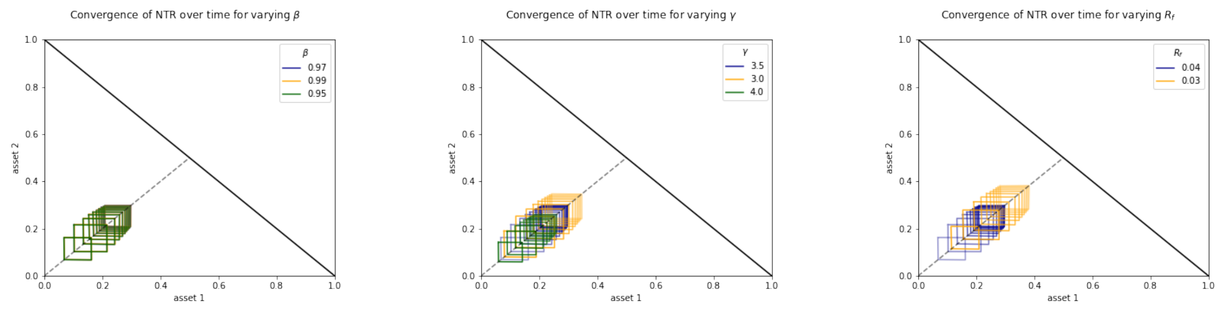


Figure 14: NTRs over time for various non-asset economic parameters. Left: varying discount factor (β); middle: varying risk aversion (γ); right: varying risk-free rate (R_f).

Next, we investigate the properties of high-dimensional NTRs using the baseline calibration (Sec. 5.1.1). Table 8 presents the relative volume that the NTR occupies in the state space for each period for the 2, 3, and 5 risky-asset cases. The relative volume is computed by taking the ratio of the NTR's volume to the volume of simplex given by $1/D!$. The results in Table 8 demonstrate that the volume of the NTR shrinks as we increase the number of assets. In the 2 risky-asset case, the NTR occupies approximately 2% of the simplex, while in the 5 risky-asset case, the NTR occupies between 0.003% and 0.01% of the simplex. This finding can be justified with a geometry argument and with an economic argument. First, holding edge length constant, the volume of a parallelotope[34] decreases as the dimensionality increases. Furthermore, more assets allow for a richer correlation structure and hence a tighter parallelotope. However, we are unable to determine the extent of this as the correlation used in this experiment is not structured. Second, as the agent has access to more assets, finding a weighting scheme that offsets the transaction costs closer to the current weighting scheme is easier. When the agent only has access to a few assets, fewer options in terms of weights exist, and hence, the agent may need to heavily rebalance one specific asset to offset the transaction costs.

	risky assets		
t	2	3	5
0	1.80	0.371	0.0082
1	1.93	0.377	0.0091
2	2.08	0.378	0.0064
3	2.24	0.370	0.0047
4	2.40	0.366	0.0034
5	2.53	0.333	0.0081
6	1.98	0.200	0.0030

Table 8: Relative NTR volume to N-simplex (%) for varying number of risky assets. The volume of the simplex for 2, 3, and 5 risky assets: 0.5, $0.1\bar{6}$, and $0.008\bar{3}$, respectively.

[34] A parallelotope is the generalization of a parallelogram to arbitrary dimensions.

6.2 Monte Carlo portfolio simulations

We now perform a battery of Monte Carlo experiments to investigate how an agent in our model behaves when trading in the presence of transaction costs. To do this, we simulate 10,000 paths of random risky-asset returns over the full lifetime of an agent. We use the baseline model parameterization (see Sec. 5.1.1) and solve a model where the agent is active for $T = 15$ years. Then, for a starting state $\mathbf{x}_{t=0}$,[35] we compute the optimal policy $\{\boldsymbol{\delta}_0, c_0\}$ for the setting with and without transaction costs.[36] Using the Monte Carlo returns and Equations (6), (7), and (9), we compute the next period's state variable $\mathbf{x}_{t+1}$ and repeat this process. We assume the agent begins with an initial wealth $W_0 = 1$ and use Equations (7) and (8) to compute their wealth for subsequent periods and compute the value of the state and policy variables: $\mathbf{X}_t = \mathbf{x}_t W_t$, $\boldsymbol{\Delta}_t = \boldsymbol{\delta}_t W_t$, and $C_t = c_t W_t$.

We first provide some general intuition with Figure 15, which plots an overview of the Monte Carlo simulations for the setting with transaction costs.[37] The top three rows plot the risky asset-specific variables, where the first (second) two columns plot the risky asset 1 (2) variables, respectively. The first row plots the random returns $R_{i,t}$; the second row plots the relative asset holdings $x_{i,t}$ (left) and the value of the holdings $X_{i,t}$ (right); and the third row plots the relative portfolio updates $\delta_{i,t}$ (left) and their value $\Delta_{i,t}$ (right). The bottom row plots the consumption c_t, the value of consumption C_t, and wealth W_t, from left to right, respectively. The main findings from this plot support the expected economic intuition. The value of consumption is smoothed over time. Wealth is gradually consumed over time. In comparison to the setting without transaction costs, the agent consumes less on average as part of their wealth is lost to the cost of trading and the inefficiency introduced by the transaction costs. The average asset holdings of both assets in earlier periods are lower in the case of transaction costs. This is driven by the fact that the agent is more likely to find themselves in a state where they would like to increase their asset holdings but where the benefit is offset by the transaction costs. However, the mean of the asset holdings in the settings with and without transaction costs converge over time.

Next, we investigate individual Monte Carlo paths to highlight specific agent behavior in Figures 16 and 17. The former offers an intuition of how the asset holdings interact with the NTR. The latter offers two examples that compare the settings with and without transaction costs. The top row in both figures plots the random Monte

[35] We sample starting states joint uniformly from the entire simplex that constitutes state space. Although this sampling strategy is slightly unrealistic, as a new agent entering the financial market would not be randomly allocated a portfolio, the results do not differ much should the agent acts optimally in the $t = 0$.

[36] To compute the optimal policies, we train a machine learning model on the optimal policy values obtained while solving the model ($y \in \{\boldsymbol{\delta}_t, c_t\}$) for each corresponding state value $\mathbf{x}_t$ (see Sec. 4.2).

[37] A similar plot for the setting without transaction costs is included in Appendix F, Figure 27.

Carlo asset returns $R_{i,t}$. The second row plots the relative asset holdings $x_{i,t}$ across time. Figure 16 additionally plots the asset holding's optimal update $\delta_{i,t}$ in the third row. The left (right) column depicts the values for risky asset 1 (and 2, respectively).

In Figure 16, we show a selected Monte Carlo path and its interaction with the NTR. In the second row, the red ribbon represents the corresponding cross-section of the NTR, with the red line providing additional information on how the asset holding is updated to the boundary of the NTR when outside. In the third row, the red bands highlight when the portfolio $\{x_{1,t}, x_{2,t}\}$ falls inside the NTR and is not traded. Beyond providing a general intuition of the behavior uncovered by the model, this figure serves to highlight an additional intuition. Namely, in $t = 4$ ($t = 8$), asset 1 (2, respectively) is not traded while the other asset is. Each asset, and each combination of assets in higher dimensions, has its own respective region where it is not traded while the other assets are traded.[38] These partial-asset NTRs create illiquidity for a subset of the tradable assets and often cover much larger areas of the state space.

Finally, Figure 17 compares the agent's optimal behavior in the setting with and without transactions costs, plotting two selected examples where the asset holdings $x_{i,t}$ are compared (second row). These plots demonstrate that, in the setting with two risky assets with the baseline parameterization, the agent anticipates the transaction costs in their future decisions as they approach the terminal period (yellow and blue lines converge). However, the right panel demonstrates that the agent is not always able to consistently anticipate returns, and the speed of convergence to no-transaction-cost levels may differ.[39]

[38]These regions can be identified in Figure 5a by the regions in the state space where the corresponding $\boldsymbol{\delta}$-policy are horizontal or vertical lines. These indicate that only one of the two risky assets is rebalanced for a given point in the state space.

[39]Figure 28 in Appendix F plots the difference between asset holdings with and without transaction costs ($x_{i,t}|\tau = 0.01 - x_{i,t}|\tau = 0.0$) across all Monte Carlo simulation paths. Over time, the agent attempts to anticipate the transaction costs, however, depending on the magnitude of the returns, the agent is more or less successful.

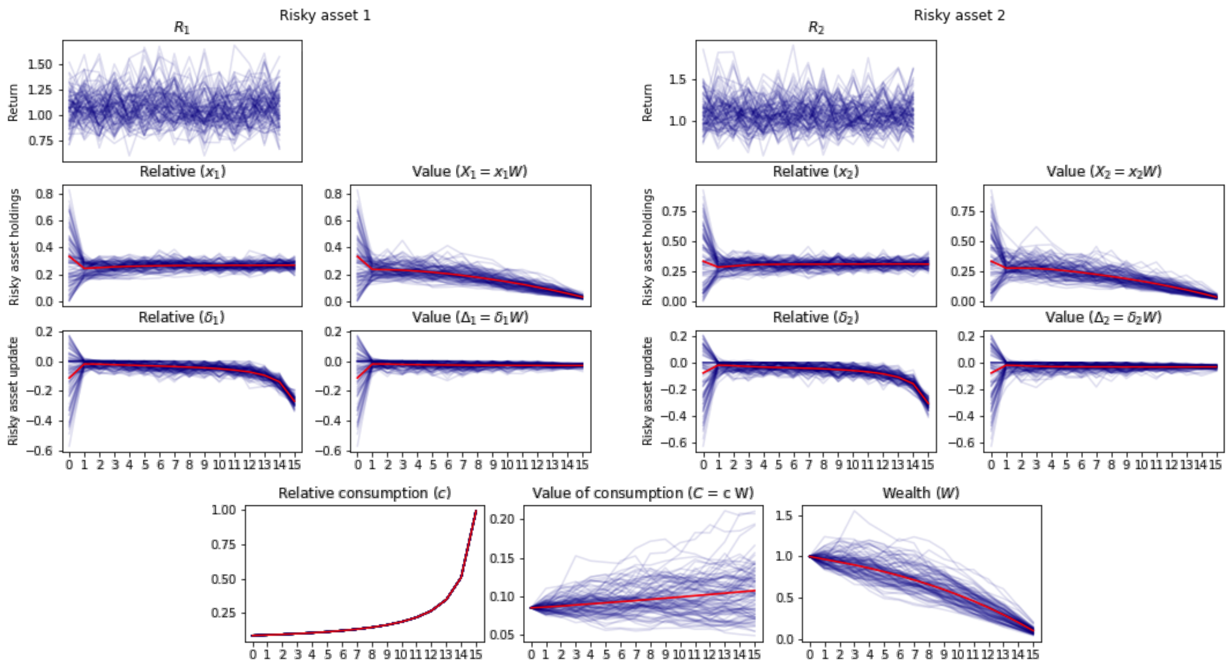


Figure 15: Monte Carlo simulation summary with transaction costs ($\tau = 0.01$).

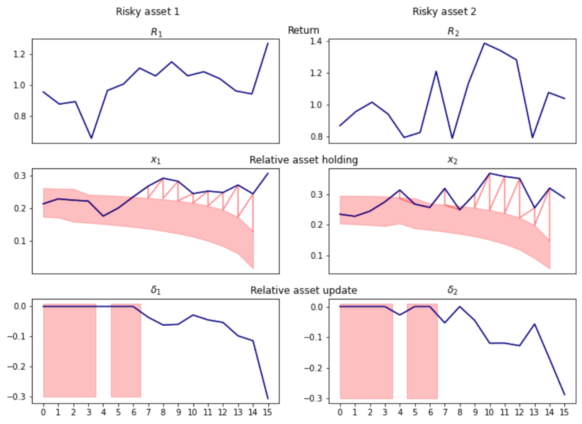


Figure 16: A selected Monte Carlo path interacting with the NTR.

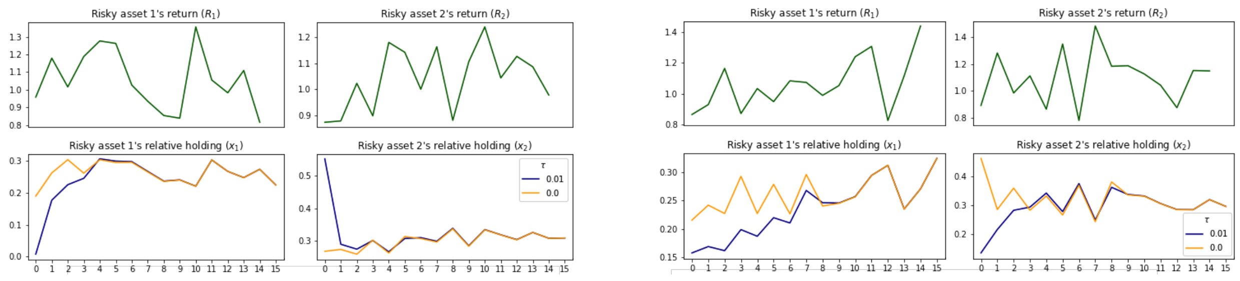


Figure 17: Comparison of Monte Carlo paths with and without transaction costs ($\tau = 0.01$ and $\tau = 0.0$).

6.2.1 Liquidity premium

The liquidity premium is defined as the expected return an agent would be willing to give up in order to not have to pay transaction costs (Constantinides, 1986). This definition is theoretically clear, however, when consumption is introduced into the model, implementation is not. In contrast to a portfolio optimization problem without consumption, consumption introduces a channel for wealth to leak in a utility-increasing way. To address this, Constantinides (1986) added an additional constraint and only considered policies where consumption was a fixed fraction of wealth $c = \alpha W$ (for some α), thereby computing an upper bound. This assumption may be satisfactory for infinite horizon models. However, in finite horizon models, the fraction of consumption changes as we approach the terminal period (*cf.* relative consumption (c) plot in Fig. 15). Hence, making similar assumptions is not possible.

We approximate the liquidity premium as follows. We compute the solution for the portfolio optimization problem as described in Section 3 with and without transaction costs and compare the respective expected lifetime utilities at time $t = 0$. In the setting without transaction costs, we reduce the risky-asset returns by the liquidity premium; a scalar scaling factor, λ, that is, we solve

$$\mathbb{E}_0\left[\sum_{t=0}^{T}\beta^t u(c_t W_t) \;\middle|\; \lambda\boldsymbol{\mu},\ \boldsymbol{\Sigma}\right],$$

with relative bond holdings

$$b_t = \left(1 - \mathbf{1}^\top \cdot \mathbf{x}_t\right) - \mathbf{1}^\top \cdot \boldsymbol{\delta}_t - c_t,$$

and terminal value

$$v_T(x_T) = u(1).$$

The remaining state dynamics and the Bellman equation remain unchanged (see Sec. 3.3.1). Note that, for simplicity and feasibility, the liquidity premium term λ is a scalar and has neither a time index nor an asset index. We use bisection to compute the value of λ such that the lifetime utilities of an agent trading with and an agent trading without transaction costs are equal. We approximate the expected lifetime utility using Monte Carlo simulations.

To solve for λ using bisection, we first set the upper and lower guesses for λ: the lower guess $\underline{\lambda}$ is set such that all the risky asset's expected returns are weakly less than the risk-free rate;[40] the upper bound $\bar{\lambda}$ is set such that the risky returns are

[40] We use $\underline{\lambda} = r/\max(\boldsymbol{\mu} + \mathrm{diag}(\boldsymbol{\Sigma})/2)$.

unchanged $\bar{\lambda} = 1.0$. Then, we compute the agent's expected lifetime utility via Monte Carlo simulations for each $\lambda \in [\underline{\lambda}, \bar{\lambda}]$.[41] Next, we compute the expected lifetime utility for a λ halfway between the upper and lower guesses: $\lambda_j = (\underline{\lambda} + \bar{\lambda})/2$. We compare these utilities to the expected lifetime utility in the case with transaction costs, also computed through a Monte Carlo simulation, to determine the new upper and lower guesses. That is, if $U_0(C|\tau \neq 0) > U_0(C|\tau = 0, \lambda_j)$, we update our lower guess $\underline{\lambda} = \lambda_j$, and if $U_0(C|\tau \neq 0) < U_0(C|\tau = 0, \lambda_j)$, we update our upper guess $\bar{\lambda} = \lambda_j$. This is repeated for a predetermined number of iterations J or until a convergence criterion is reached.

This approach is computationally very expensive as it involves solving a dynamic program and conducting a Monte Carlo simulation for every $\lambda \in [\bar{\lambda}, \underline{\lambda}, \lambda_1, ..., \lambda_J]$. Therefore, we only perform $J = 5$ bisection iterations. While this allows us to approximate bounds on the liquidity premium, it does not allow us to comment on the quality of the bounds. We, therefore, also compute a final best guess λ^* in an attempt to gain higher precision on the point estimate.[42] We perform this exercise to get an initial understanding of how the liquidity premium behaves as we scale dimensions, not to provide a feasible solution to approximating the liquidity premium.

The bisection results for each iteration and the best guess for the 2, 3, and 5 risky-asset cases are presented in Table 9. The 5-iteration bisection results give us liquidity premium bounds for the 2, 3, and 5 risky-asset cases, respectively: [0.0%, 3.13%], [0.0%, 3.47%], [0.0%, 4.01%]. That is, an agent would be willing to sacrifice up to approximately 4% of the risky-asset drift to avoid paying 1% transaction cost. However, as all the approximated bounds include 0.0%, these are not very informative. The best guesses offer more precision, suggesting that the agent would be willing to sacrifice 3.08%, 1.15%, and 1.60% of risky-asset drift, respectively.[43] The extent of the economic interpretation we can do at this point is unfortunately limited as the baseline parameterization does not allow for systematic comparison. However, the trend seems to indicate that agents are able to avoid some of the effects of illiquidity when given a richer set of assets to optimize over. At this point, we offer a proof-of-concept to compute the liquidity premium for high-dimensional discrete-time dynamic portfolio optimization problems with consumption and transaction costs.

[41]The Monte Carlo simulations are conducted as described in Section 6.2. That is, baseline model parameterization (see Sec. 5.1.1).

[42]The best guess is determined by fitting a univariate spline of degree 3 (default settings) on the bisection results, $U_0(C|\tau = 0) \sim \lambda$, and determining the root $U_0(C|\tau \neq 0) = U_0(C|\tau = 0)$.

[43]These values correspond to the following returns sacrificed per risky asset by the agent: 2 risky assets: [3.77%, 3.86%]; 3 risky assets: [1.40%, 1.44%, 1.47%]; 5 risky assets: [1.96%, 2.00%, 2.05%, 2.10%, 2.15%].

2 risky assets		3 risky assets		5 risky assets	
λ	$U_{\tau=0}/U_{\tau\neq 0}$	λ	$U_{\tau=0}/U_{\tau\neq 0}$	λ	$U_{\tau=0}/U_{\tau\neq 0}$
0.5	115.65	0.4444	118.37	0.3584	125.82
0.75	111.36	0.7222	114.38	0.6792	120.98
0.875	105.63	0.8611	107.89	0.8396	111.08
0.9375	102.00	0.93055	103.76	0.9198	105.09
0.96875	100.02	0.965275	101.54	0.9599	101.93
0.96918562*	100.00	0.98854459*	100.00	0.98402378*	99.98
1.00	97.95	1.00	99.23	1.00	98.71

Note: * indicates the best guess; Lifetime utilities for with and without transaction cost (best guess): 2 risky assets: [-2242.7005, -2242.6088]; 3 risky assets: [-251,789,580.012, -251,789,112.587]; 5 risky assets: [-17,380,459,596,785.88, -17,377,577,421,688.197].

Table 9: Bisection results for liquidity premium approximation.

6.2.2 Lifetime utility loss from transaction costs

Finally, we investigate now the percentage of expected lifetime utility an agent loses due to transaction costs and due to adopting an alternative strategy that is easier to compute. These experiments were also conducted using the baseline parameterization (see Sec. 5.1.1) with Monte Carlo simulations (see Sec. 6.2). The comparisons are made between agents that are both exposed to the same sequences of Monte Carlo shocks. First, we compare the expected lifetime utility of an agent active in a setting with transaction costs versus an agent active in a setting without transaction costs. The results are presented in Table 10. The trend we find with lifetime utility loss mirrors that of liquidity premium. The agent loses up to 2% of their lifetime utility when trading two risky assets with transaction costs. This number falls when the agent is given more assets to trade, suggesting that a richer set of assets offers avenues for the agents to offset the inefficiencies of the transaction costs. Second, we compute the expected lifetime utility of an agent trading that rebalances her portfolio using an alternative strategy, comparing the optimal strategy to the strategy from the setting without transaction costs and the strategy where an agent approximates the proportional transaction costs with a quadratic transaction cost.[44] In both of these alternative scenarios, the agent still pays proportional transaction costs, but acts as if the transaction costs take a different form.[45] The results are presented in Table 11 and demonstrate the importance of using the correct strategy. Agents who base their decisions on an approximation where they ignore transaction costs lose an increasing amount of their lifetime utility when increasing the number of assets at their availability. While this utility loss can be reduced by using a quadratic approximation, the use of an incorrect strategy remains costly. This serves to demonstrate the importance of a method that is able

[44]The approximation was computed by minimizing $\min_a \int_{-1}^{1}(\tau|x| - ax^2)^2 dx$, for proportional transaction cost $\tau = 0.01$. The result is $a \approx 0.0124$.

[45]This analysis further illustrates the generality of the proposed solution method as it is agnostic to the form of the transaction costs.

to approximate a solution for a high-dimensional portfolio optimization problem with proportional transaction costs.

Risky assets	% of lifetime utility lost
2	2.05
3	0.77
5	1.29

Note: Lifetime utilities for with and without transaction cost: 2 risky assets: [-2242.7005, -2196.8066]; 3 risky assets: [-251,789,580.012, -249,857,885.799]; 5 risky assets: [-17,380,459,596,785.88, -17,156,148,727,970.414].

Table 10: Percentage of expected lifetime utility lost due to transaction costs.

Risky	% of lifetime utility lost	
assets	No TC	Quadratic TC
2	0.20	0.15
3	12.54	9.49
5	29.70	23.08

Note: These results were computed using the original baseline calibration horizon of $T = 7$. Lifetime utilities for proportional transaction costs, without transaction costs, and a quadratic approximation of proportional transaction costs: 2 risky assets: [-348.85, -349.56, -349.37]; 3 risky assets: [-19,437.05, -21,874.74, -21,282.29]; 5 risky assets: [-2,080,796.05, -2,698,782.50, -2,561,073.88].

Table 11: Percentage of expected lifetime utility lost due to agent trading with an incorrect approximation of the proportional transaction costs. The agent trades with a strategy that is optimal given no transaction costs or given quadratic transaction costs in a setting where they pay proportional transaction costs.

7 Conclusion

In this paper, we address the numerical challenges of multi-asset portfolio choice models. We propose a generic, scalable, and flexible computational framework for high-dimensional, finite-horizon, discrete-time dynamic stochastic portfolio optimization with proportional transaction costs. Specifically, we propose a solution method based on GPRs in a DP framework combined with smart sampling. Using GPs offers a grid-free approach that allows us to avoid the downsides grid-based methods encounter. We augment our method by using an approximation of the NTR that enables us to efficiently oversample data inside the NTR and along kinks. This combination addresses challenges posed by high-dimensional state spaces and irregular geometries, and enable us to alleviate the curse of dimensionality.

We apply the proposed method to a portfolio choice model with two to five risky assets, a risk-free bond, and per-period consumption. A CRRA agent maximizes her expected lifetime utility by managing a portfolio of assets over a specified finite time horizon. She rebalances her portfolio at discrete intervals and gains utility each period by consuming parts of her wealth. The method's solution and the economic insights were evaluated on two parameterizations: one based on Schober et al. (2022) for benchmarking our method against the existing framework and a systematic two-risky-asset configuration to isolate the effect of individual parameters on the NTR. Our analysis demonstrates that our framework can offer high-quality solutions with fewer points than existing methods. Our framework fits within the broader area of portfolio optimization, as proportional transaction costs are a key feature in financial models. It provides a new opportunity to address questions, such as: how regulatory shifts that affect transaction costs might influence market behaviors; how emerging markets' high transaction costs affect international portfolio diversification strategies; or how portfolios with technologies with various transaction costs schemes, like cryptocurrency, fit into an investor's portfolios, without sacrificing asset diversity or other severe simplifying assumptions.

References

Abrams, R. A. and Karmarkar, U. S. (1980). Optimal multiperiod investment-consumption policies. *Econometrica: Journal of the Econometric Society*, pages 333–353.

Akian, M., Menaldi, J. L., and Sulem, A. (1996). On an investment-consumption model with transaction costs. *SIAM Journal on control and Optimization*, 34(1):329–364.

Albuquerque, R., Song, S., and Yao, C. (2020). The price effects of liquidity shocks: A study of the sec's tick size experiment. *Journal of Financial Economics*, 138(3):700–724.

Atkinson, C. and Mokkhavesa, S. (2004). Multi-asset portfolio optimization with transaction cost. *Applied Mathematical Finance*, 11(2):95–123.

Baccarin, S. and Marazzina, D. (2014). Optimal impulse control of a portfolio with a fixed transaction cost. *Central European Journal of Operations Research*, 22(2):355–372.

Baule, R. (2008). Optimal portfolio selection for the small investor considering risk and transaction costs. *OR Spectrum*, 32:61–76.

Bellman, R. (1961). *Adoptive control processes: A guided tour*. University Press.

Brown, D. B. and Smith, J. E. (2011). Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Science*, 57(10):1752–1770.

Brumm, J. and Scheidegger, S. (2017). Using adaptive sparse grids to solve high-dimensional dynamic models. *Econometrica*, 85(5):1575–1612.

Bungartz, H.-J. and Griebel, M. (2004). Sparse grids. *Acta Numerica*, 13:1–123.

Cai, Y., Judd, K. L., and Xu, R. (2013). Numerical solution of dynamic portfolio optimization with transaction costs. Technical report, National Bureau of Economic Research.

Constantine, P. G. (2015). *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM.

Constantinides, G. M. (1979). Multiperiod consumption and investment behavior with convex transactions costs. *Management Science*, 25(11):1127–1137.

Constantinides, G. M. (1986). Capital market equilibrium with transaction costs. In *Theory Of Valuation*, pages 207–227. World Scientific.

Domingos, P. (2012). A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87.

Dybvig, P. H. and Pezzo, L. (2020). Mean-variance portfolio rebalancing with transaction costs. *Available at SSRN 3373329*.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31.

Gârleanu, N. and Pedersen, L. H. (2013). Dynamic trading with predictable returns and transaction costs. *The Journal of Finance*, 68(6):2309–2340.

Goodman, J. and Ostrov, D. N. (2010). Balancing small transaction costs with loss of optimal allocation in dynamic stock trading strategies. *Siam journal on applied mathematics*, 70(6):1977–1998.

Gropp, W., Lusk, E., and Skjellum, A. (1999). *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press.

Hausdorff, F. (1914). *Grundzüge der mengenlehre*. Göschens Lehrbücherei/Gruppe I: Reine und Angewandte Mathematik Series. Von Veit.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Janeček, K. and Shreve, S. E. (2004). Asymptotic analysis for optimal investment and consumption with transaction costs. *Finance and Stochastics*, 8(2):181–206.

Kamin, J. H. (1975). Optimal portfolio revision with a proportional transaction cost. *Management Science*, 21(11):1263–1271.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leland, H. (2000). Optimal portfolio implementation with transactions costs and capital gains taxes. *Haas School of Business Technical Report*.

Liu, H. (2004). Optimal consumption and investment with transaction costs and multiple risky assets. *The Journal of Finance*, 59(1):289–338.

Liu, W. (2019). Portfolio diversification across cryptocurrencies. *Finance Research Letters*, 29:200–205.

Lynch, A. W. and Tan, S. (2010). Multiple risky assets, transaction costs, and return predictability: Allocation rules and implications for us investors. *Journal of Financial and Quantitative Analysis*, 45(4):1015–1053.

Magill, M. J. and Constantinides, G. M. (1976). Portfolio selection with transactions costs. *Journal of economic theory*, 13(2):245–263.

Mei, X., DeMiguel, V., and Nogales, F. J. (2016). Multiperiod portfolio optimization with multiple risky assets and general transaction costs. *Journal of Banking & Finance*, 69:108–120.

Merton, R. C. (1971). Optimum consumption and portfolio rules in a continuous-time model. In *Stochastic optimization models in finance*, pages 621–661. Elsevier.

Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, 7(12).

Muhle-Karbe, J., Sefton, J. A., and Shi, X. (2023). Dynamic portfolio choice with intertemporal hedging and transaction costs. *Available at SSRN 4522752*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Murphy, K. P. (2023). *Probabilistic machine learning: Advanced topics*. MIT Press.

Muthuraman, K. and Kumar, S. (2006). Multidimensional portfolio optimization with proportional transaction costs. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 16(2):301–335.

Muthuraman, K. and Zha, H. (2008). Simulation-based portfolio optimization for large portfolios with transaction costs. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 18(1):115–134.

Oberkampf, W. L. and Roy, C. J. (2010). *Verification and validation in scientific computing*. Cambridge University Press.

Oksendal, B. and Sulem, A. (2002). Optimal consumption and portfolio with both fixed and proportional transaction costs. *SIAM Journal on control and optimization*, 40(6):1765–1790.

Rasmussen, C. E. and Nickisch, H. (2010). Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015.

Renner, P. and Scheidegger, S. (2018). Machine learning for dynamic incentive problems. *Available at SSRN 3282487*.

Scheidegger, S. and Bilionis, I. (2019). Machine learning for high-dimensional dynamic stochastic economies. *Journal of Computational Science*, 33:68–82.

Scheidegger, S. and Treccani, A. (2018). Pricing American Options under High-Dimensional Models with Recursive Adaptive Sparse Expectations. *Journal of Financial Econometrics*, 19(2):258–290.

Schober, P., Valentin, J., and Pflüger, D. (2022). Solving high-dimensional dynamic portfolio choice models with hierarchical b-splines on sparse grids. *Computational Economics*, 59(1):185–224.

Wächter, A. and Biegler, L. T. (2006). *On the implementation of an interior-point algorithm for non-linear optimization with some add-ons*. Argonne National Laboratory.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR.

Zabel, E. (1973). Consumer choice, portfolio decisions, and transaction costs. *Econometrica: Journal of the Econometric Society*, pages 321–335.

Zhang, R., Langrené, N., Tian, Y., Zhu, Z., Klebaner, F., and Hamza, K. (2019). Dynamic portfolio optimization with liquidity cost and market impact: a simulation-and-regression approach. *Quantitative Finance*, 19(3):519–532.

A Dropping wealth as state variable with CRRA utility

Using

$$u(C) = u(cW) = \frac{c^{1-\gamma}W^{1-\gamma}}{1-\gamma} = W^{1-\gamma}u(c)$$

and assuming that $V(W, \mathbf{x}) = W^{1-\gamma}v(\mathbf{x})$:

$$\begin{aligned} V_t(W_t, \mathbf{x}_t) = W_t^{1-\gamma}v_t(\mathbf{x}_t) &= \max_{c_t,\delta_t} W_t^{1-\gamma}u(c_t) + \beta\mathbb{E}\{W_{t+1}^{1-\gamma}v_{t+1}(\mathbf{x}_{t+1})\} \\ \Rightarrow v_t(\mathbf{x}_t) &= \max_{c_t,\delta_t} u(c_t) + \beta\mathbb{E}\{\pi_{t+1}^{1-\gamma}v_{t+1}(\mathbf{x}_{t+1})\} \qquad |\pi_{t+1} = \frac{W_{t+1}}{W_t} \end{aligned}$$

B Additional examples of state-space points to approximate higher-dimensional NTRs

Recall from Section 4.3.1 that the location and size of a NTR, and hence its vertices, are unknown *a priori*. To approximate the vertices, points must be sampled from the state space such that they are likely to update to the vertices of the NTR, *i.e.*, from the blue regions in Figure 1 (and the higher-dimensional equivalences). Without any additional information about the NTR, points that are on the vertices of the simplex, the origin, and the midpoints between each combination of simplex vertices, are likely to fall in said blue regions. For a 3-dimensional state space, the $2^3 = 8$ points are:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

while, for a 4-dimensional state space, the $2^4 = 16$ points are:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0.0 & 0.5 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

C The unfortunate edge case when identifying the NTR

There is an unfortunate edge-case that must be contended with. The method we describe to approximate the NTR (see Sec. 4.3.1) fails when the true NTR is not fully contained in the feasible state space (see Figure 18). This edge-case seems to be more common when modeling higher-dimensional sets of risky assets. However, there are multiple ways to handle this:

- Allow costless shorting and/or borrowing only for NTR approximation step (Mei et al., 2016),
- Solve for additional state-space points that update to the NTR face to infer the intersection of multiple faces,
- Approximate the NTR incorrectly and omit components of the algorithm that require a correct NTR approximation (*i.e.*, NTR assisted data sampling does not necessarily require correct NTR approximation while fitting multiple GPs on various regions of the state space may).
- Use more flexible machine learning methods to approximate NTR (*e.g.*, Gaussian mixture models (Murphy, 2012)).

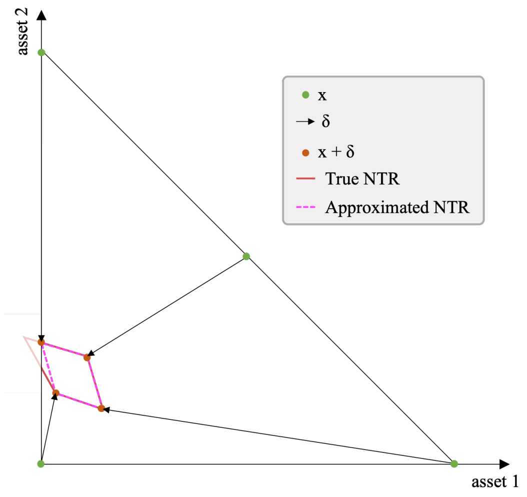


Figure 18: NTR approximation edge case. If the full NTR is not contained in the feasible space, *i.e.*, the NTR overlaps an axis (or simplex boundary), the NTR may be approximated incorrectly.

Note that, each of these methods comes with its own difficulties and implications. For example, Gaussian mixture models have high computational overhead, approximating the faces of high-dimensional NTR faces (which tend to be smaller geometrical objects) may be difficult to implement, and omitting algorithm components may reduce the precision of the value function approximation in critical areas of the state space. However, depending on the requirements of the modeler, a satisfactory solution can be found.

D Computational details

The algorithm as outlined in Section 4.4 omits all implementation and technical details. However, the performance of a numerical implementation of our method relies on a number of components that influence the runtime and error measurements. In this section, we outline the computational decisions made in our implementation. Section D.1 details computational decisions regarding GPs, including the choice of mean and kernel function, boundaries on the noise, and the training specification; Section D.2 outlines the choice of constraint optimizer and its calibration; Section D.3 discusses the parallelization of the algorithm focusing on distribution the solving of the Bellman equations across multiple nodes; and Section D.4 outlines the programming language and libraries used to implement the various components discussed in the previous sections.

D.1 Details on Gaussian processes

As discussed in Section 4.2, GPRs require the modeler to select a mean function and a kernel function. We chose the mean function to be a constant, which is a standard choice that allows the mean to be non-zero but poses no other assumptions *a priori*. We use a scaled Matern kernel $\theta_{\text{scale}}\mathcal{K}(x, x'|l)$, where $\mathcal{K}$ is the Matern kernel defined by Equation (16). The scale parameter θ_{scale} is fit using the data. We set the smoothness parameter to $\nu = 1.5$ and allow for automatic relevance detection (ARD). ARD allows us to calibrate a length-scale parameter ℓ_i (see Eq. (16)) for each state dimension independently rather than having a single length scale parameter for all dimensions. Although there are more formal ways of tuning these hyperparameters, ours were determined through unstructured trial and error.

The only sources of noise in our training data (Eqs. (20) and (21)) are approximation errors, solver tolerance, and machine precision rounding errors. Other than these, our training labels are precise, and hence noise can be minimized. Therefore, the upper noise bound was kept very low at $\sigma_\epsilon^2 \leq 10^{-8}$. Allowing for too much noise detriments our approximation quality, most significantly at kinks. The noise bound could be set lower; however, computational precision hinders the inverting of the matrix.

To fit the GP parameters to the data, we use the Adam optimizer (Kingma and Ba, 2014), which is a variant of stochastic gradient descent that uses momentum and a decayed adaptive learning rate. Momentum computes a weighted average of past gradients, promoting convergence along consistent gradients while reducing the influence of noisy gradients. Adam optimizes the learning rate by adapting it for each parameter through normalization with a weighted average of the historical squared gradients. The learning rate is another hyperparameter chosen by the modeler, which determines the step size taken during gradient descent. We ran each experiment with two learning rate schedules. First, we used a larger learning rate of 1.0, which allowed us to approach the optimum quickly, followed by a lower learning rate of 0.1, allowing us to solve for the optimum with higher precision. When running the experiments, we ran each schedule for between 25–1000 iterations using Adam-based stochastic gradient descent. For experiments with a relatively small sample size, overfitting the GPR is more common and can be counteracted by reducing the number of Adam iterations. Both GPRs (inside and outside the NTR) were trained using the same configurations on their respective training datasets (Eqs. (20) and (21)).

Finally, it is worth noting that GPRs are not often considered a suitable tool for high-dimensional analysis, as the kernel's distance measure becomes uninformative in settings with more than approximately 20 dimensions (Domingos, 2012). However, we highlight two points that provide additional context. First, high-dimensional problems pose computational challenges irrespective of the chosen approximation method. Consider a 10 versus 20-dimensional hypercube in a state space. To solve for only the vertex values of these hypercubes, we must solve approximately 1,000 versus 1,000,000 Bellman equations, respectively. Our method will first be limited by the computational requirements of solving the N Bellman equations before we reach the bounds of GPRs' informativeness. Second, contemporary dimensionality reduction techniques, such as active subspaces (Scheidegger and Bilionis, 2019) and deep kernel learning (Wilson et al., 2016), may enable the handling of larger input dimensions.

D.2 Constraint optimizer for solving the Bellman equation

Recall from Section 4.1 that in order to obtain the previous period's value function values and controls for each point in a set of sampled state points, we solve the Bellman equation. To do this, we use a constrained optimizer, specifically, the interior point optimizer (IPOPT; Wächter and Biegler, 2006, `http://www.coin-or.org/Ipopt/`). The optimizer's configuration is straightforward. That is, all hyperparameters are set as low as possible such that the algorithm still converges reliably. Again, we tune these hyperparameters using unstructured trial and error. We use a tolerance of 10^{-6} across all experiments. If computation does not converge after 300 seconds, optimization on a given state point with a given initial guess is stopped. Up to five initial guesses are used per point in the state space, and up to 20% of the observations can be dropped per iteration if not converged. For all experiments conducted, this bound was never violated.

D.3 Parallelization

To efficiently solve larger problems within a reasonable timeframe, we use parallel computing. Particularly, there is a part of the DP algorithm (see Sec. 4.1) that allows for straightforward parallelization: the evaluation of the Bellman operator. Each iteration in the DP procedure involves solving N separate optimization problems, where N is the number of observations. These problems are independent and can be solved simultaneously, making the generation of training data for GP highly suited to parallelization. We utilized the Message Passing Interface (MPI; Gropp et al., 1999) to implement parallelization in our algorithm. Assume that we have n_{cpu} computational cores at our disposal, also referred to as workers or processes. In each iteration of the DP algorithm, we distribute (or *broadcast*) the current value function across all processes, allowing each to evaluate the Bellman operator independently. This distribution has minimal communication costs. Then, the N training inputs can be processed in parallel. Each worker takes a fraction of the test points, reducing their individual workload where the bulk of the computational time is spent. Instead of a single process solving N optimization problems, n_{cpu} now solve N/n_{cpu} problems. Once completed, the workers *gather* the distributed data, a process that also requires minimal communication. Furthermore, we have also parallelized the fitting of the GP hyperparameters across MPI workers, offering further acceleration.

D.4 Technological framework

The code was written in Python 3 using the `PyTorch` (`https://pytorch.org`) auto-differentiation package, which allows the computation of exact gradients. These gradients are used by the constraint optimizer (Sec. D.2), gradient descent solver (Sec. D.1), and to compute the Euler errors (Sec. 5.2.2) helping to avoid human error when computing the gradients manually and to gain more precision in our gradients than finite differences offers. Furthermore, `PyTorch` offers an integrated distribution backend that simplifies the implementation of any MPI communication (Sec. D.3).

The `GPyTorch` package (`https://gpytorch.ai`) is `PyTorch`'s corresponding GP extension. These packages allow us to accelerate the fitting and inference of the GPR through two avenues: first, using a gradient descent method (as discussed in Section D.1) rather than the commonly used BFGS solver, thereby benefitting from the most recent algorithmic advancements in stochastic gradient descent methodologies, and second, using black box matrix-matrix multiplication (Gardner et al., 2018), which reduces the computational complexity of GPR inference from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$, where N is the sample size. Furthermore, it provides access to the tools developed for machine learning that, for example, stabilize convergence (*e.g.*, the Adam optimizer).

A strength of this setup is that many additional technical tricks can be implemented to accelerate training. First, `Pytorch` has integrated GPU distribution, which can accelerate training for large datasets. Second, the auto-differentiation feature allows for dimension reduction of various forms. For example, our framework allows for the integration of active subspaces (Constantine, 2015) or neural network preprocessing (known as deep kernel networks; Wilson et al., 2016). These additions allow for more economically complex models to be addressed, however, they were not necessary for the model investigated in this paper. Finally, the `cyipopt` package (`https://cyipopt.readthedocs.io/en/stable/`) is a Python wrapper for the IPOPT optimization package that enables C-like performance.

With this technical stack and very limited parallelization, we ran all experiments to completion in less than one week, while the experiments with two risky assets were significantly faster. Note, however, that at this time, optimization was not a priority. A number of straightforward methods to reduce the computation time are readily available. For example, further scaling our compute resources (*i.e.*, more MPI workers), efficient sampling of data points to minimize our training data, implementing stochastic gradient descent rather than gradient descent to accelerate the fitting of the GPs, and using sparse linear solvers in the constraint optimizer.

E Derivation of the Euler equation error

This section provides the derivation for the EEs: a key metric used to assess the quality of our intertemporal solutions.

$$
\begin{aligned}
\varepsilon_{c_t}(\mathbf{x}_t) &= \frac{\partial u(c_t) + \beta \mathbb{E}_t\left[\pi_{t+1}^{1-\gamma} v_{t+1}(\mathbf{x}_{t+1})\right]}{\partial b_t} \\
&= \frac{\partial u(c_t)}{\partial b_t} + \beta \mathbb{E}_t \frac{\partial \pi_{t+1}^{1-\gamma} v_{t+1}(\mathbf{x}_{t+1})}{\partial b_t} \\
&= \frac{\partial u(c_t)}{\partial c_t}\frac{\partial c_t}{\partial b_t} + \beta \mathbb{E}_t\left[\frac{\partial \pi_{t+1}^{1-\gamma}}{\partial b_t} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\frac{\partial v_{t+1}(\mathbf{x}_{t+1})}{\partial b_t}\right] \\
&= -c_t^{-\gamma} + \beta \mathbb{E}_t\left[\frac{\partial \pi_{t+1}^{1-\gamma}}{\partial \pi_{t+1}}\frac{\partial \pi_{t+1}}{\partial b_t} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\frac{\partial v_{t+1}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}}\frac{\partial \mathbf{x}_{t+1}}{\partial \pi_{t+1}}\frac{\partial \pi_{t+1}}{\partial b_t}\right] \\
&= -c_t^{-\gamma} + \beta \mathbb{E}_t\left[\frac{\partial \pi_{t+1}}{\partial b_t}\left(\frac{\partial \pi_{t+1}^{1-\gamma}}{\partial \pi_{t+1}} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\frac{\partial v_{t+1}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}}\frac{\partial \mathbf{x}_{t+1}}{\partial \pi_{t+1}}\right)\right] \\
&= -c_t^{-\gamma} + \beta \mathbb{E}_t\left[R_f\left((1-\gamma)\pi^{-\gamma} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\nabla_{\mathbf{x}_{t+1}} v_{t+1}(\mathbf{x}_{t+1})\left(-\frac{\mathbf{x}_{t+1}}{\pi_{t+1}}\right)\right)\right] \\
&= -c_t^{-\gamma} + \beta \mathbb{E}_t\left[R_f \pi_{t+1}^{-\gamma}\left((1-\gamma) v_{t+1}(\mathbf{x}_{t+1}) - \nabla_{x_{t+1}} v_{t+1}(\mathbf{x}_{t+1})\mathbf{x}_{t+1}\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
\varepsilon_{\delta_{i,t}}(\mathbf{x}_t) &= \frac{\partial u(c_t) + \beta \mathbb{E}_t\left[\pi_{t+1}^{1-\gamma} v_{t+1}(\mathbf{x}_{t+1})\right]}{\partial \delta_{i,t}} \\
&= \frac{\partial u(c_t)}{\partial \delta_{i,t}} + \beta \mathbb{E}_t \frac{\partial \pi_{t+1}^{1-\gamma} v_{t+1}(\mathbf{x}_{t+1})}{\partial \delta_{i,t}} \\
&= \frac{\partial u(c_t)}{\partial c_t}\frac{\partial c_t}{\partial \delta_{i,t}} + \beta \mathbb{E}_t\left[\frac{\partial \pi_{t+1}^{1-\gamma}}{\partial \delta_{i,t}} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\frac{\partial v_{t+1}(\mathbf{x}_{t+1})}{\partial \delta_{i,t}}\right] \\
&= \frac{\partial u(c_t)}{\partial c_t}\frac{\partial c_t}{\partial \delta_{i,t}} + \beta \mathbb{E}_t\left[\frac{\partial \pi_{t+1}^{1-\gamma}}{\partial \pi_{t+1}}\frac{\partial \pi_{t+1}}{\partial \delta_{i,t}} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\frac{\partial v_{t+1}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}}\frac{\partial \mathbf{x}_{t+1}}{\partial \delta_{i,t}}\right] \\
&= c_t^{-\gamma}\frac{\partial c_t}{\partial \delta_{i,t}} + \beta \mathbb{E}_t\left[(1-\gamma)\pi_{t+1}^{-\gamma} R_{i,t} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\nabla_{\mathbf{x}_{t+1}} v_{t+1}(\mathbf{x}_{t+1})\left(\frac{1}{\pi_{t+1}}(\mathbf{e}_i * \mathbf{R}_t - \mathbf{x}_{t+1} R_{i,t})\right)\right] \\
&= c_t^{-\gamma}\frac{\partial c_t}{\partial \delta_{i,t}} + \beta \mathbb{E}_t\left[(1-\gamma)\pi_{t+1}^{-\gamma} R_{i,t} v_{t+1}(\mathbf{x}_{t+1}) + \pi_{t+1}^{1-\gamma}\nabla_{\mathbf{x}_{t+1}} v_{t+1}(\mathbf{x}_{t+1})\frac{r_{i,t}}{\pi_{t+1}}(\mathbf{e}_i - \mathbf{x}_{t+1})\right] \\
&= c_t^{-\gamma}\frac{\partial c_t}{\partial \delta_{i,t}} + \beta \mathbb{E}_t\left[R_{i,t}\pi_{t+1}^{-\gamma}\left((1-\gamma) v_{t+1}(\mathbf{x}_{t+1}) - \nabla_{\mathbf{x}_{t+1}} v_{t+1}(\mathbf{x}_{t+1})(\mathbf{e}_i - \mathbf{x}_{t+1})\right)\right]
\end{aligned}
$$

F Supplementary figures and tables

Figure 19 depicts the cross section of the predicted value function for each of the sample sizes and three data-sampling protocols. The columns plot the results from a model solved by sampling data using a joint-uniform distribution (left), a joint-uniform distribution with oversampling the approximated NTR (middle), and a joint-uniform distribution with oversampling the approximated NTR and the approximated kinks (right). The rows plot various sample sizes from $N = [100, 200, 300, 500, 700]$, from top to bottom, respectively.

Visually, we observe that both of the strategies that oversample the NTR achieve a better value function fit than standard joint-uniform sampling. First, with joint-uniform sampling, we were only able to solve the model using $N \geq 500$ points, where we still had difficulties resolving the NTR accurately due to the low number of points sampled in the relatively small NTR. This is visible in the large fluctuations of the bottom two panels of the first column. Second, large fluctuations are also observable in the top right panel. That is, when oversampling the approximated NTR and kinks, the algorithm was unable to correctly approximate the value function. The remaining panels all portray similar value function approximations without regions of large fluctuations. Through careful inspection, we are able to see that increasing the sample size improves the predictions, as there are fewer small fluctuations in the lines. For example, in the top middle panel where we use a model trained with data that oversampled the approximated NTR and $N = 100$, we can observed slight kinks in the peaks. These are indicative of poor approximation quality and are not present when the sample size is increased.

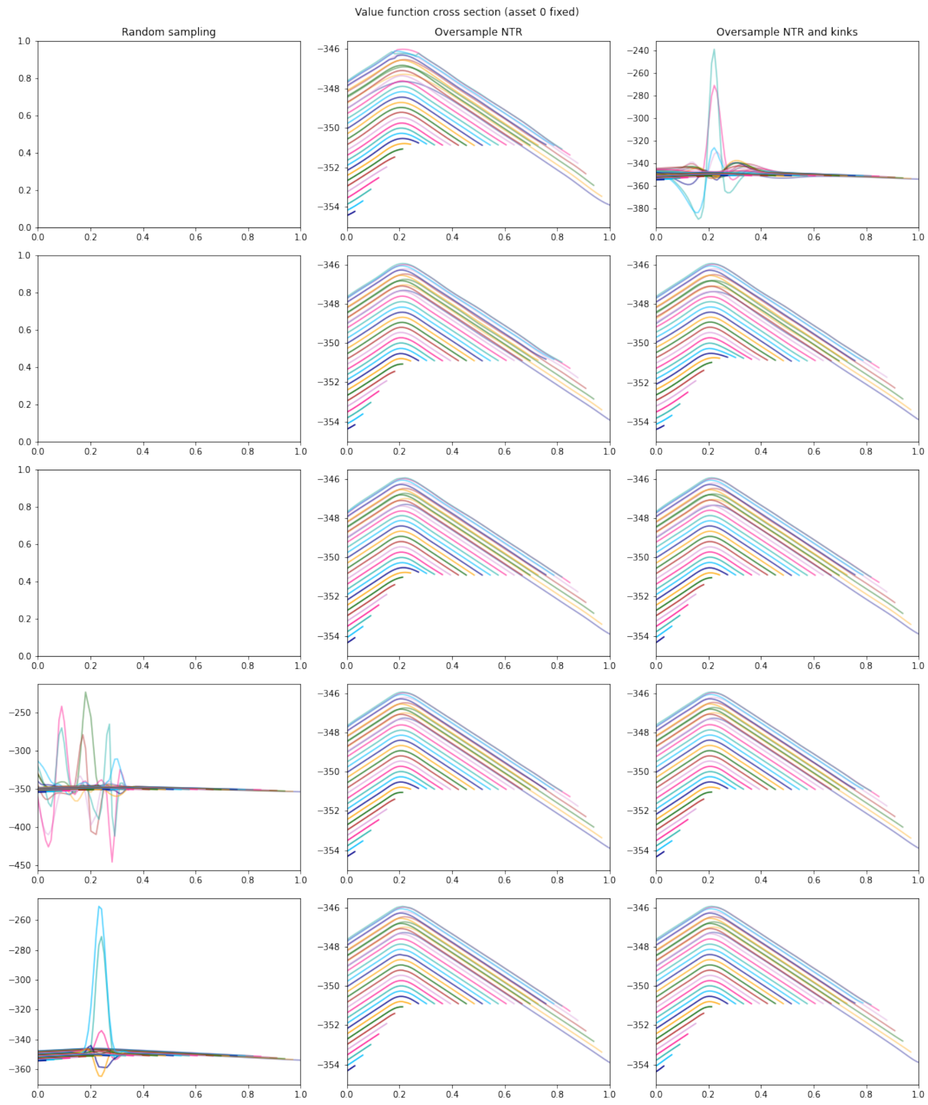


Figure 19: Cross section of the final iteration value function prediction using Schober et al. (2022) parameterization where asset 1 values are fixed. Columns (left to right): model solved with joint-uniform sampled data, joint-uniform sampled data and approximated NTR oversampled, and joint-uniform sampled data and approximated NTR and approximated kinks oversampled, respectively. Rows (top to bottom): $N = 100, 200, 300, 500$, and 700, respectively.

Figure 20 plots heatmaps of the out-of-sample RAE statistics to assess the value function fit for various sample sizes. We plot $N = [100, 300, 500, 700]$ from left to right, respectively. The plots were generated using a model solved with data that was sampled joint-uniformly with the approximated NTR and approximated kinks oversampled. The summary statistics are presented in Table 12. These results correspond to the results from the model without the kinks oversampled, which are presented in Figure 4 and Table 3, respectively. Again, we observe that the errors across the majority of the state space are very low and regions of high error only accrue locally. The second panel demonstrates that the errors accrue around kinks, while the third panel demonstrates that they also accrue where few points are sampled (near simplex). In contrast to Figure 4, the kinks do not seem to be as pronounced for $N = 500$, suggesting that kinks can be sampled strategically to increase the global fit.

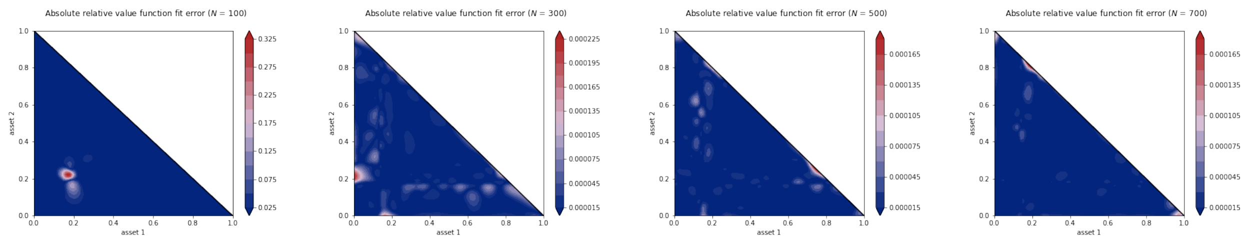


Figure 20: Out-of-sample RAE value function fit heatmap. The plots were generated using solutions from a model where the NTR and kinks were oversampled.

N	Mean (%)	99.9th percentile (%)	Max (%)
100	0.568	26.78	33.1
300	0.0014	0.0191	0.0238
500	0.0007	0.0129	0.0182
700	0.0006	0.0129	0.0195

Table 12: Out-of-sample RAE value function fit summary statistics (in percent) of the final iteration (iteration 7) for various sample sizes. The summary statistics were computed using solutions from a model where the NTR and kinks were oversampled.

Figure 21 plots a full comparison of the RAE statistics for each of the sample sizes and three data-sampling protocols. The columns plot the results from a model solved by sampling data using a joint-uniform distribution (left), a joint-uniform distribution with oversampling of the approximated NTR (middle), and a joint-uniform distribution with oversampling of the approximated NTR and the approximated kinks (right). The rows plot various sample sizes from $N = [100, 200, 300, 500, 700]$ from top to bottom, respectively. The main results are discussed above and in Section 5.2.1. Additionally, we see relatively higher local errors when the NTR is not oversampled (left column).

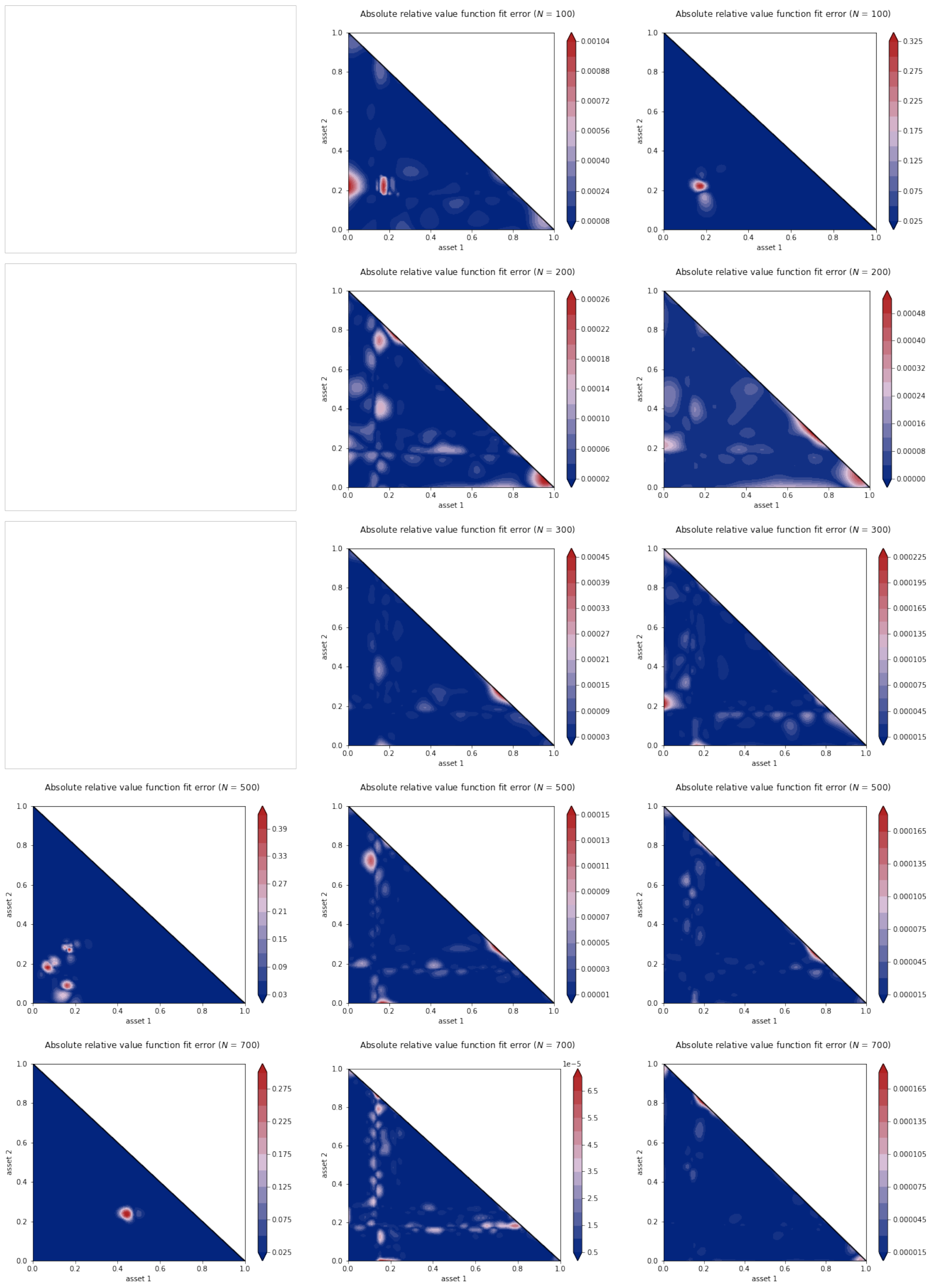


Figure 21: Out-of-sample value function fit heatmaps measured by RAE. Columns (left to right): model solved with joint-uniform sampled data, joint-uniform sampled data and approximated NTR oversampled, and joint-uniform sampled data and approximated NTR and approximated kinks oversampled, respectively. Rows (top to bottom): $N = 100, 200, 300, 500$, and 700, respectively.

Figure 22 plots the out-of-sample RAE summary statistics over sample size. Again, the sampling method that does not oversample the NTR underperforms, only achieving mean-RAE parity with the other methods for large sample sizes. The other two sampling methods (oversampling NTR and oversampling NTR and kinks) perform similarly for sample sizes of $N \geq 100$.

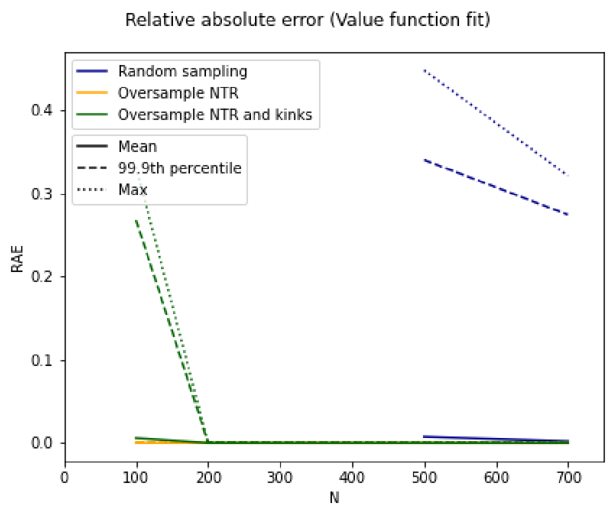
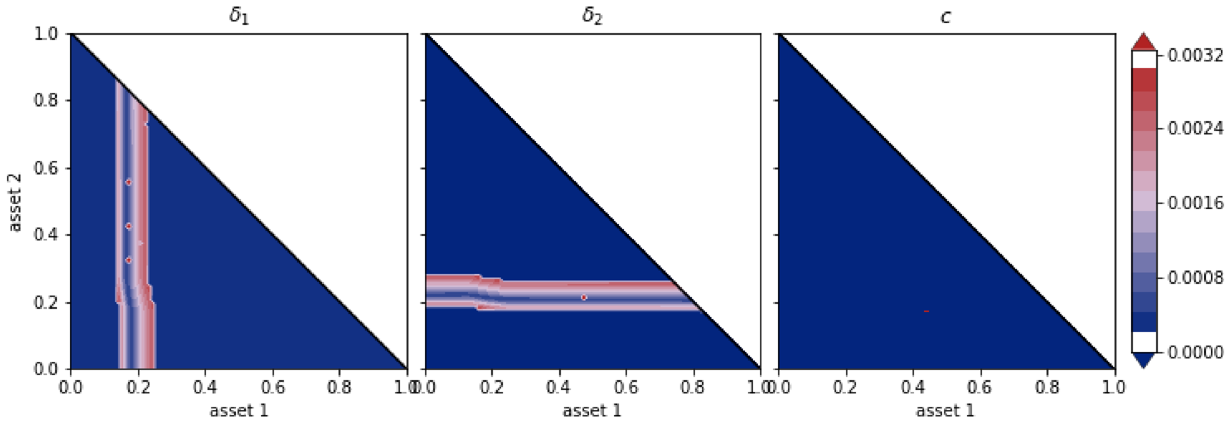


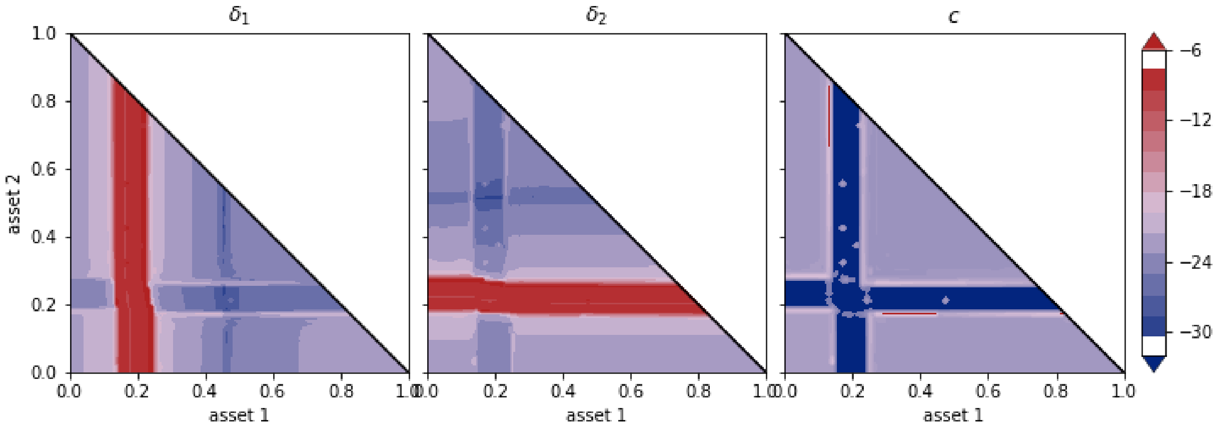
Figure 22: Out-of-sample value function fit statistics as measured by RAE over sample size for different data-sampling methods.

Figures 23 and 24 plot additional heatmaps to those in Section 5.2.2. Both figures were generated using models that were solved using joint-uniformly sampled data with the approximated NTRs oversampled. Figure 23 was generated using $N = 500$. Figure 23 plots the heatmaps by policy while Figure 24 plots the REE with respect to consumption for various sample sizes. Both figures have 2 panels. The top depicts the raw values, while the bottom depicts the log values in order to provide more visual distinction. Table 13 presents the summary statistics corresponding to Figures 23 and 24 in Panel A. Panel B provides similar summary statistics of the out-of-sample REE for a model that was solved using joint-uniformly sampled data with the approximated NTRs and kinks oversampled.

Figure 23 shows low REEs in the full state space except where $\delta_i = 0$. In this area, the gradient with respect to δ_i is undefined due to the kink induced by the transaction cost. This is again confirmed by the statistics in Table 13 and justifies why we additionally offer the REE statistics where we omit the regions where $\delta_i = 0$. Again, these values were not reported by others, and hence we cannot offer a meaningful comparison. The REEs with respect to c show low errors globally. Figure 24 shows that these results hold for lower sample sizes where the max values do not exceed 2×10^{-5} and 10^{-7}, respectively. The results in Table 13 demonstrate that the sample size and data-sampling methods do not have a large effect on the REE statistics. However, Table 13 demonstrates that oversampling the kinks can also yield small improvements in the 99.9th-percentile and max $\boldsymbol{\delta}$-errors, suggesting again that the kinks are a main driver of the errors. Finally, Figure 25 plots the change in EEs for the various data-sampling methods as sample size increases. Again, oversampling the NTR leads to large improvements in performance.

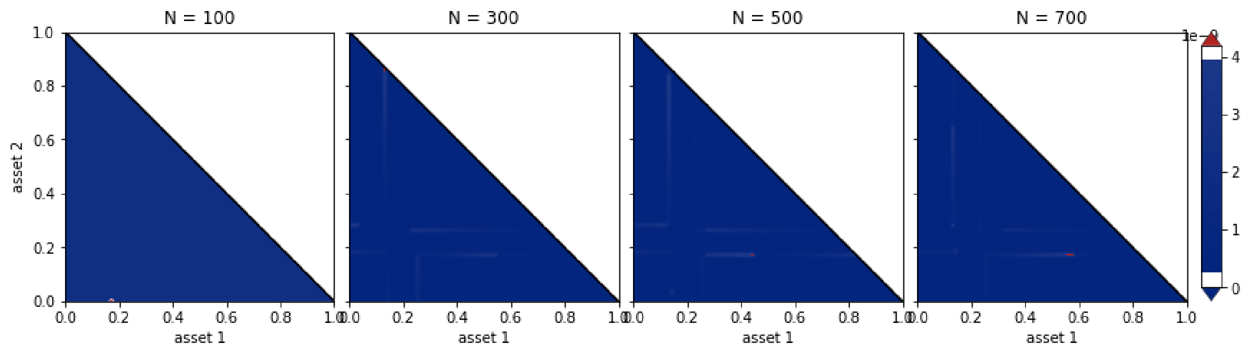


(a) Absolute relative Euler errors.

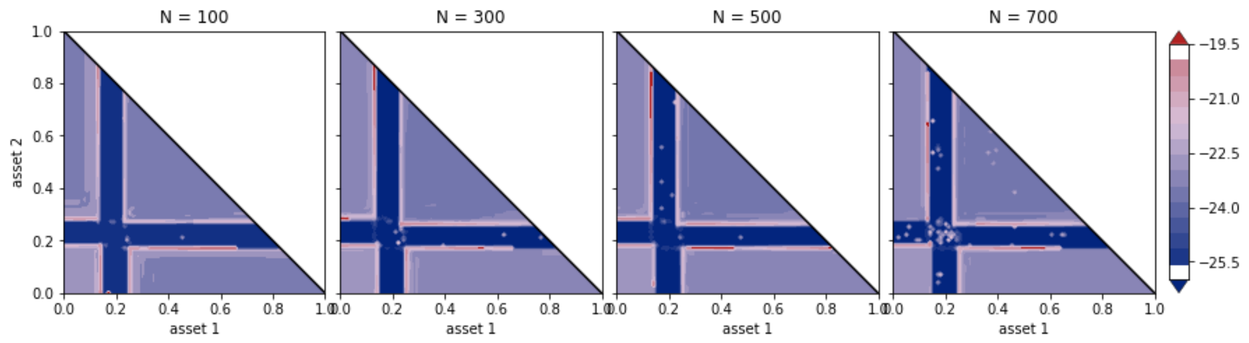


(b) Log absolute relative Euler errors.

Figure 23: Final iteration (iteration 7) out-of-sample REEs by policy variable. The plots were generated using solutions from a model where the NTR was oversampled with a sample size of $N = 500$.



(a) Absolute relative Euler errors.



(b) Log absolute relative Euler errors.

Figure 24: Final iteration (iteration 7) out-of-sample REEs by sample size. The plots were generated using solutions from a model where the NTR was oversampled.

Panel A: Approximated NTR oversampled

N		Mean	99.9th percentile	Max
300	δ_1	0.0222 (2.9×10^{-6})	0.273 (5.3×10^{-4})	0.283 (6.8×10^{-4})
	δ_2	0.0189 (3.9×10^{-6})	0.284 (4.8×10^{-4})	0.392 (5.5×10^{-4})
	c	1.3×10^{-8}	4.0×10^{-7}	4.3×10^{-7}
500	δ_1	0.0222 (4.6×10^{-6})	0.278 (7.9×10^{-4})	0.334 (9.9×10^{-4})
	δ_2	0.0179 (4.7×10^{-6})	0.274 (5.4×10^{-4})	0.332 (1.1×10^{-3})
	c	1.4×10^{-8}	4.7×10^{-7}	5.3×10^{-7}
700	δ_1	0.0223 (2.7×10^{-6})	0.304 (3.9×10^{-4})	0.393 (4.8×10^{-4})
	δ_2	0.0185 (4.1×10^{-6})	0.284 (6.0×10^{-4})	0.395 (8.2×10^{-4})
	c	1.1×10^{-8}	3.9×10^{-7}	4.4×10^{-7}

Panel B: Approximated NTR and kinks oversampled

N		Mean	99.9th percentile	Max
300	δ_1	0.0221 (5.4×10^{-6})	0.272 (9.9×10^{-4})	0.284 (1.2×10^{-3})
	δ_2	0.0190 (8.4×10^{-6})	0.283 (1.5×10^{-3})	0.396 (2.0×10^{-3})
	c	1.5×10^{-8}	4.8×10^{-7}	5.0×10^{-7}
500	δ_1	0.0282 (8.2×10^{-6})	0.269 (9.8×10^{-4})	0.285 (1.5×10^{-3})
	δ_2	0.0227 (1.7×10^{-6})	0.270 (9.2×10^{-5})	0.282 (1.5×10^{-4})
	c	8.3×10^{-8}	2.5×10^{-6}	2.6×10^{-6}
700	δ_1	0.0225 (2.9×10^{-6})	0.282 (6.1×10^{-4})	0.395 (8.6×10^{-4})
	δ_2	0.0184 (7.8×10^{-6})	0.282 (1.5×10^{-3})	0.396 (2.0×10^{-3})
	c	1.3×10^{-8}	4.2×10^{-7}	4.2×10^{-7}

Note: Values in brackets are computed by omitting observations where $\delta_i = 0$ for $i \in [1, 2]$. As the gradient w.r.t. δ is not defined at $\delta_i = 0$, the EE is not defined.

Table 13: Final iteration (iteration 7) out-of-sample REE summary statistics (in percent) for two data-sampling protocols and various sample sizes.

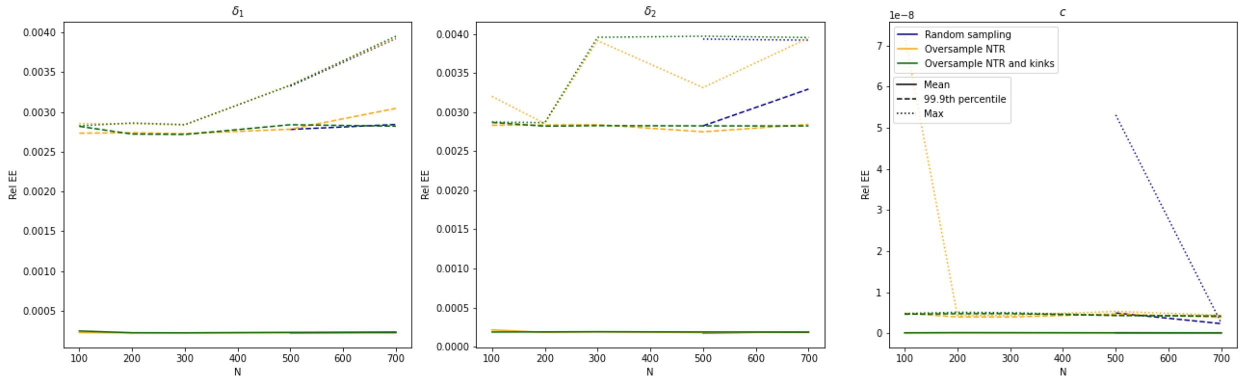


Figure 25: Absolute REE over sample size for various data-sampling methods.

Figure 26 plots the final iteration's approximated NTR for various sample sizes. This figure was generated using a model that was solved on data that was joint-uniformly sampled with oversampled NTR and kinks. This figure can be compared to Figure 6b in Section 5.2.3. Table 14 are the corresponding Hausdorff distances (Eq. (24)) for each pair of NTRs. Both Figure 26 and Table 14 demonstrate that the approximated NTR is robust to sample size when we oversample the kinks in addition to the NTR. All NTRs are highly similar, with only the $N = 100$ NTR yielding larger errors.

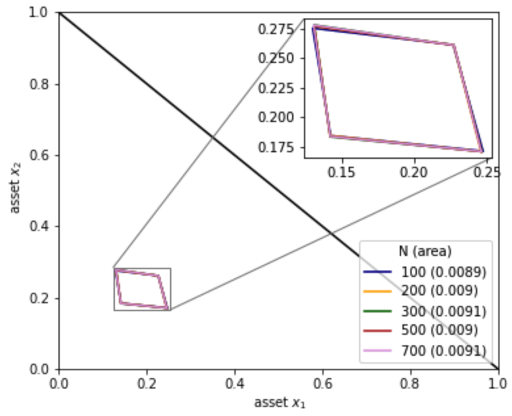


Figure 26: Comparing the final iteration's NTR computed using various sample sizes. The NTR approximations were computed using solutions from a model where the NTR and kinks were oversampled.

N	100	200	300	500	700
100	0.0				
200	0.0027	0.0			
300	0.0029	0.0018	0.0		
500	0.0022	0.0019	0.0019	0.0	
700	0.0022	0.0021	0.0019	0.0020	0.0

Table 14: Hausdorff distance between NTRs across sample size computed using solutions from a model where the NTR and kinks were oversampled.

Table 15 presents the results of the percentage of overlapping area between two consecutive NTRs over time. The values were generated using a parameterization of identical assets and no correlation. The results show that as we progress through the DP iterations (backward through time), the NTRs converge. That is, the distance between each consecutive NTR shrinks. Ultimately, the NTRs will converge to their infinite-horizon values suggesting that, given the consumption policy converges as well, a T_1-period finite-horizon solution can approximate a T_2-period finite-horizon solution, for $T_1 < T_2$ and sufficiently large T_1.

t	Overlap (%)
0	91.8
1	90.7
2	88.5
3	88.0
4	84.0
5	79.0
6	71.4
7	56.3
8	27.2

Table 15: Percentage of overlapping NTR area between the t^{th} and $t+1^{\text{st}}$ iterations for identical assets without correlation (corresponds to the left panel of Figure 11).

Figure 27 plots an overview of the Monte Carlo simulation for the setting without transaction costs. That is, an identical dynamic program as outlined in Section 3 with $\tau = 0$. This figure corresponds to Figure 15 in the main text. The top three rows plot the risky asset-specific variables, whereas the first (second) two columns plot the risky asset 1 (2) variables, respectively. The first row plots the random returns $R_{i,t}$; the second row plots the relative asset holdings $x_{i,t}$ (left) and the value of the holdings $X_{i,t}$ (right); and the third row plots the relative portfolio updates $\delta_{i,t}$ (left) and their value $\Delta_{i,t}$ (right). The bottom row plots the consumption c_t, the value of consumption C_t, and wealth W_t, from left to right, respectively. The simulated risky-asset returns are identical to those used for Figure 15. The findings from Figure 27 are similar to those displayed in Figure 15. That is, consumption is smoothed over time, and wealth is gradually consumed over time. As highlighted in the main text: the agent that is exposed to transaction costs consumes less on average as part of their wealth is lost to the cost of trading. The means of the asset holdings in both settings (with and without transaction costs) converge over time.

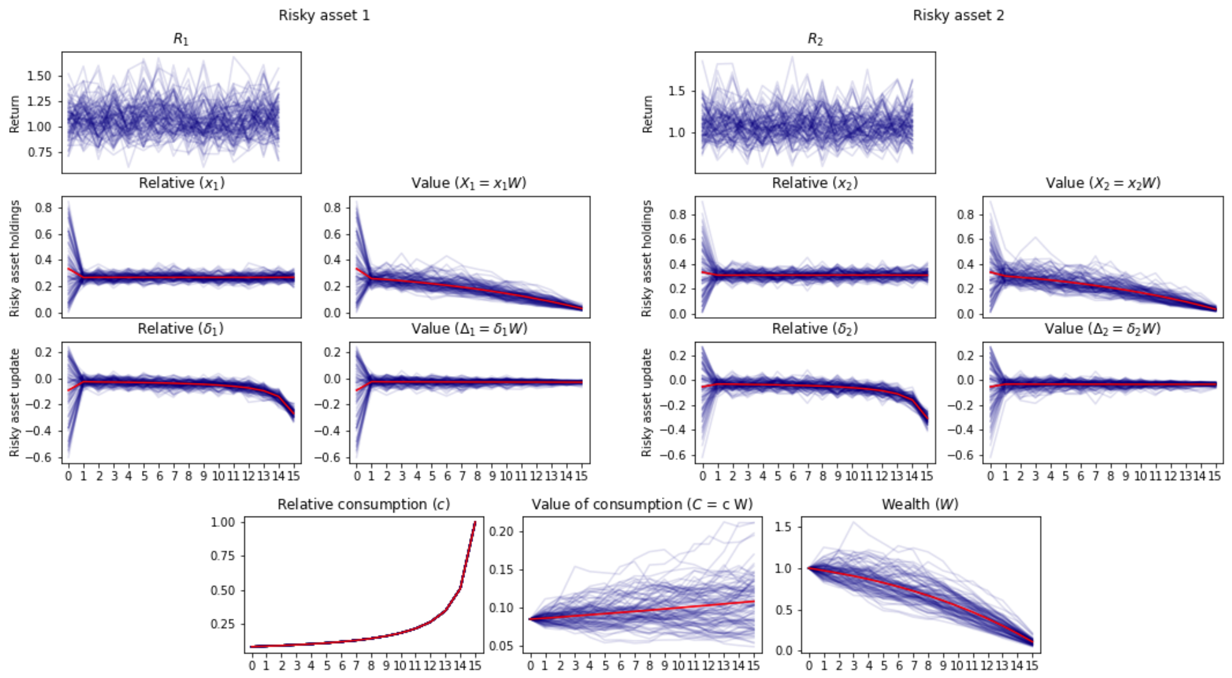


Figure 27: Monte Carlo simulation summary without transaction costs ($\tau = 0$).

That the asset holdings in the setting with and without transaction costs converge over time is largely due to the intuition discussed relative to Figure 17. That is, in the two risky-asset setting with the Schober et al. (2022) calibration, the agent is able to anticipate the transaction costs in a large portion of the state space. The optimal policy of an agent who pays transaction costs is similar to an agent who does not pay transaction costs. Figure 28 demonstrates this by plotting the difference in risky-asset holdings for asset 1 (2) between the setting with and without transaction costs in the left (right) panels, respectively. For many Monte Carlo simulation paths, the agent is able to anticipate the transaction costs and hence plays a strategy similar to the optimal strategy given trading is free. This is signified by the difference being close to 0 in the plot. Note, however, that this is not a general finding, but is a result of the parameterization used. We do not find similar results for the higher-dimensional cases.

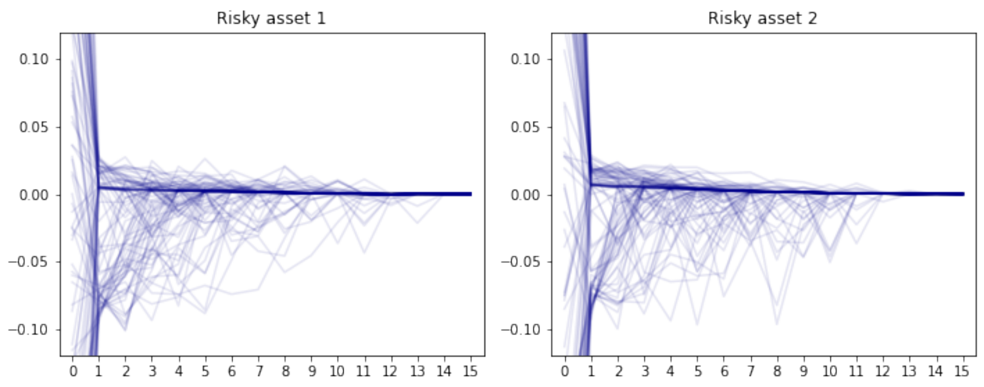


Figure 28: Difference between risky-asset holdings with and without transaction costs over time.