Multi-asset financial markets: mathematical modelling and data-driven approaches

Milena Vuletić
Christ Church
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity Term 2025

To my parents.

Acknowledgements

I am deeply grateful to my supervisors, Prof. Rama Cont and Prof. Mihai Cucuringu, for supporting me and encouraging intellectual freedom throughout this journey. Rama put me in the spotlight early, offering guidance, rigorous training, and thoughtful mentorship. Mihai believed in me from the beginning, and without him, I might never have ventured into generative modelling.

This DPhil would not have been possible without the financial support of BNP Paribas, through the EPSRC Centre for Doctoral Training in Mathematics of Random Systems (EPSRC Grant EP/S023925/1). I am also grateful to my examiners (past and present), and to all those who contributed to my work through feedback and questions at seminars and conferences.

I am especially thankful to my family for their constant presence and encouragement throughout all the highs and lows. This journey would not have been possible without the emotional support of those around me. I first thank Abbas, without whom I may have never applied to the programme.

I remain grateful to my lifelong friends for their unwavering support: Mina, who always reminded me she was there; Vuk, who helped me transition into DPhil life; Mia, who was present for many key moments; and Bogdan, who I could count on, regardless of distance.

I would also like to acknowledge those who joined my life during the course of this DPhil and who have made a lasting impact: Domenico, whose patience, presence, and unwavering support carried me through more than he probably knows; Debbie, for sharing both the good and the difficult while exploring the world together; Linnea, for warmly welcoming me into the Christ Church family; and Anna, for the many unique and memorable experiences.

Statement of Originality

Chapters 2, 3 and 4 are based on joint work with Prof. Rama Cont. Chapter 2 is based on our paper 'Simulation of arbitrage-free implied volatility surfaces' [1], and Chapter 3 is based on 'VolGAN: a generative model for arbitrage-free implied volatility surfaces' [2]. Chapter 4 is based on 'Data-driven hedging with generative models' [3]. Chapter 5 is based on joint work with Dr. Felix Prenzel and Prof. Mihai Cucuringu, more precisely, on 'Fin-GAN: forecasting and classifying financial time series via generative adversarial networks' [4], which extends my Part C dissertation. Chapter 6 is based on joint work with Prof. Mihai Cucuringu, 'GraFiN-Gen: graph-based ensemble generative modelling for multi-asset forecasting' [5].

In all cases, I was the principal lead of the research project and involved in all aspects of the research - theory, numerics, data analysis, and implementation.

References:

[1] Cont, R. and Vuletić, M. (2023). 'Simulation of Arbitrage-Free Implied Volatility Surfaces', *Applied Mathematical Finance*, 30(2), pp. 94–121. doi: 10.1080/1350486X.2023.2277960.

[2] Vuletić, M. and Cont, R. (2025). 'VolGAN: A Generative Model for Arbitrage-Free Implied Volatility Surfaces', *Applied Mathematical Finance*, pp. 1–36. doi: 10.1080/1350486X.2025.2471317.

[3] Cont, R. and Vuletić, M. (2025). 'Data-driven hedging with generative models'. https://ssrn.com/abstract=5282525.

[4] Vuletić, M., Prenzel, F. and Cucuringu, M. (2024). 'Fin-GAN: forecasting and classifying financial time series via generative adversarial networks', *Quantitative Finance*, 24(2), pp. 175–199. doi: 10.1080/14697688.2023.2299466.

[5] Vuletić, M. and Cucuringu, M. (2025). 'GraFiN-Gen: graph-based ensemble generative modelling for multi-asset forecasting'. https://ssrn.com/abstract=5317725.

Abstract

This thesis develops statistical models and data-driven algorithms for modelling, simulation, and forecasting asset price dynamics in financial markets with many instruments and risk factors, focusing on equity and option markets. In such multi-asset settings, modelling of co-movements is crucial in order to implement correct hedging strategies, and generate realistic portfolio dynamics and loss distributions. Models also need to respect arbitrage relations linking prices of various instruments, which often imposes nonlinear constraints on state variables.

Chapter 1 introduces notation, outlines the thesis structure, and compares various types of generative models, establishing shared themes and contributions.

Chapter 2 presents a computationally tractable method for simulating arbitrage-free implied volatility surfaces. Our approach conciliates static arbitrage constraints with a realistic representation of statistical properties of implied volatility co-movements.

Chapter 3 introduces VolGAN, a generative model for arbitrage-free implied volatility surfaces. The model is trained on time series of implied volatility surfaces and underlying prices, and is capable of generating realistic scenarios for the joint dynamics of the implied volatility surface and the underlying asset.

Chapter 4 proposes a non-parametric data-driven methodology for hedging using generative models. In contrast with conventional model-based hedging approaches relying on sensitivity analysis of model pricing functions, our approach uses (conditional) generative models to simulate realistic market scenarios given current market conditions, and computes hedging strategies which minimise risk across these scenarios. The approach incorporates trading costs, leads to an optimal selection of hedging instruments, and adapts to market conditions. We illustrate the effectiveness of this

methodology for hedging option portfolios using VolGAN, and compare its performance with delta and delta-vega hedging.

Chapter 5 investigates the use of Generative Adversarial Networks (GANs) for probabilistic forecasting of financial time series. To this end, we introduce a novel economics-driven loss function for the generator, rendering GANs more suitable for a classification task. Our approach, named Fin-GAN, moves beyond pointwise forecasts and allows for uncertainty estimates. Numerical experiments on equity data showcase the effectiveness of our proposed methodology, which achieves higher Sharpe Ratios compared to commonly used supervised learning models, such as LSTM and ARIMA.

Chapter 6 explores the construction of conditional generative models in a multi-asset setting by leveraging cross-asset relationships through a graph-based probabilistic ensemble framework. Rather than combining point forecasts, our method ensembles full conditional return distributions. The graph captures the transferability of predictive information across assets, with edge weights learned via a profit-maximisation objective reformulated as a LASSO regression. This computationally efficient approach induces sparse and interpretable weights. We apply our method to Fin-GAN, and demonstrate that the LASSO-induced graph outperforms benchmarks, including asset-specific models, return correlation-based graphs, and graph structures based on historical PnL or Sharpe Ratio attained by single-asset generators.

Contents

List of Figures

Chapter 1

Introduction

While the bulk of mathematical models in finance have focused on detailed analytical modelling of single-asset markets, a large number of practical applications involve multi-asset markets, often with a large number of assets. This leads to many theoretical and computational challenges related to scalability, model sparsity, and arbitrage constraints. This thesis aims to develop efficient analytical and data-driven modelling approaches and algorithms for multi-asset financial markets which are scalable to high-dimensional settings.

Our focus is on custom conditional generative models tailored to specific financial tasks, such as simulating implied volatility surfaces and forecasting asset returns. We show how these models can be directly applied to hedging, risk management, and trading strategy design. The methodology in this thesis bridges traditional financial modelling and modern machine learning.

Although generative artificial intelligence (GenAI) models perform exceptionally well in domains like image and language generation, their application to financial data presents distinct challenges, some of which we list below:

- high dimensionality of market data,
- limited availability of historical samples,
- complex and nonstationary temporal structure,
- time-varying and nonlinear dependencies,
- lack of standardised validation protocols,
- low interpretability and trustworthiness of outputs.

These issues often prohibit the use of large, generic architectures from other domains. Instead, we advocate for carefully tailored models that are suitable for financial purposes. Moreover, even if a model is appropriately trained, evaluating the quality of simulated scenarios is not straightforward. This is particularly true when training directly on market data. Unlike domains where realism can be judged visually or semantically, financial generative models must be assessed using application-specific performance metrics.

This thesis develops and validates two custom conditional generative models: VolGAN [134], for simulating arbitrage-free implied volatility surfaces, and Fin-GAN [136], for directional return forecasting. In both cases, we evaluate model robustness and performance on real-world financial data and demonstrate their use in hedging and trading applications. These models illustrate how tailored generative modelling can address real financial problems in a scalable, data-efficient, and interpretable manner.

Unlike many existing approaches that rely on model-based simulations or assume access to abundant stationary data, our framework adapts to real-world conditions, requires minimal assumptions, and remains robust under changing market regimes.

1.1 Generative models for financial time series

Generative models are a class of machine learning models designed to learn the (unknown) data-generating distribution through sample observations. Their objective is to approximate the (conditional) probability distribution of observed data, and to generate new samples which statistically resemble it. They represent a very powerful tool, since they learn from the data directly and do not presume strict distributional forms.

Although much attention has been devoted to applying a particular class of generative models to financial settings, namely Large Language Models (LLMs) [28, 91, 111], one of the most natural use cases of GenAI in finance is generating forward-looking market scenarios which are consistent with historical observations. Such scenarios can be used directly for tail risk simulation [40], trading strategies [136], portfolio construction [30], and hedging [44].

At first glance, it might appear preferable to learn optimal solutions to financial tasks, such as trading or hedging, directly from data, bypassing the modelling of market dynamics. This leads to a reinforcement learning approach [64], where actions are learned end-to-end to optimise performance. However, reinforcement learning

typically requires a large number of stationary, repeated paths to learn optimal policies, which is not compatible with the nature of financial markets. In addition, each shift in market conditions or task specification would necessitate costly retraining, reducing adaptability and scalability.

To address data limitations, a growing body of work focuses on Market Generators [72], which are machine learning models trained to generate synthetic market paths. These simulated paths are then used to train reinforcement learning algorithms such as Deep Hedging [17]. While this improves data availability, it introduces an additional modelling layer, and does not eliminate the need for repeated retraining of downstream policies when market conditions change.

Instead, this thesis focuses on learning the one-step-ahead joint distribution of market prices and risk factors, conditional on current (and past) market information. Once a conditional generative model is trained and validated on historical market data, it can be used to simulate plausible forward-looking scenarios based on current market conditions. These scenarios are then used to make informed decisions through an optimisation approach. The market dynamics is learned by the generative model once, and the scenarios are simulated given current market conditions. This leads to a more adaptive, faster, and computationally efficient approach compared to reinforcement learning, avoiding repeated trial-and-error learning or heavy and restrictive assumptions on market dynamics tied to analytical modelling.

We now review the most common families of generative models. A more comprehensive comparison is provided in [14].

- **Generative Adversarial Networks (GANs)** [60] are composed of two neural networks: a generator and a discriminator. The generator transforms input noise, typically Gaussian, into synthetic data samples. Rather than directly comparing its outputs to the actual data samples, the generator relies on the discriminator for learning, i.e. it is trained adversarially through the feedback it receives from the discriminator. The discriminator's role is to differentiate between real and generated samples, and it is trained as a classifier. Unlike the generator, the discriminator has access to the real data and compares the real and simulated samples.

- **Variational Autoencoders (VAEs)** [83] also consist of two neural networks: an encoder and a decoder. The encoder learns to map its input data to a distribution over a lower-dimensional latent space (which is typically Gaussian),

from which samples can be drawn. The decoder reconstructs the data sample from its latent representation.

- **Energy-Based Models (EBMs)**[90] define an unnormalised data-generating probability density by introducing an energy function $E_\theta(x)$, parameterised by a neural network, which assigns lower values to regions of higher probability. The associated density is proportional to $\exp(-E_\theta(x))$ and the normalisation constant is typically intractable. EBMs can be trained by likelihood maximisation.

- **Diffusion Models** [126, 127], consist of a forward and a reverse process. The forward process gradually adds noise to the data until it reaches a limiting distribution, which is typically Gaussian. The reverse process, which aims to map the limiting noise back to the data, is learned and is typically modelled as a parameterised stochastic differential equation, or as a sequence of de-noising steps. The reverse process is usually trained by score matching [75] or variational inference [69, 82].

- **Normalising Flows** [113] apply a sequence of invertible transformations, called flows, to a latent (noise) prior distribution in order to match the target data distribution. These transformations are parameterised by neural networks and allow for exact likelihood computation via the change-of-variables formula, unlike EBMs. Contrary to diffusions, the backwards process (mapping noise to data) is not learned separately, but rather given by the inverse of the forward transformation.

- **Neural Stochastic Differential Equations (neural SDEs)** [31, 81] model the temporal evolution of training data with a stochastic differential equation, whose drift and volatility terms are represented by neural networks.

- **(Deep) Autoregressive Models** [132] factor the joint distribution of sequential data into a product of conditional distributions. They generate data sequentially, based on the previous elements, by modelling conditional probabilities with neural networks. Large Language Models belong to this class, and they often use a Transformer architecture [133].

These diverse classes of generative models each offer different levels of explainability/interpretability, target data distribution flexibility, computational efficiency, sampling tractability, and analytical availability of the learned data-generating distribution.

To leverage the strengths of different models while addressing their limitations, hybrid approaches have been proposed. For example, VAE-GANs [89] aim to improve VAEs by treating the decoder as a generator and introducing a discriminator for adversarial training. Flow-GANs [62] impose an invertible generator and train it using a combination of adversarial and maximum likelihood losses. Diffusion-GANs [142] replace the reverse process of diffusion models with a GAN to increase the sampling efficiency. Although these hybrid architectures aim to enhance generative model performance in general settings, our focus is learning market dynamics for financial decision making.

In the setting of this thesis, we are interested in being able to sample from the one-step-ahead distribution of market variables and risk factors based on the current state of the market. One-step-ahead conditional generative modelling provides a flexible and data-efficient way to learn market dynamics. Compared to path-based approaches, this setup benefits from a larger effective sample size, as overlapping windows allow the use of every time step in training. It is also well-suited to settings with limited data or changing market regimes, as it can quickly adapt to new information without requiring the simulation of long trajectories. Despite its apparent simplicity, the one-step simulation proves highly effective for financial decision-making tasks such as hedging and forecasting, as we show throughout this thesis.

We distinguish two sets of variables of interest:

- market prices $S_t^1, ..., S_t^n$ of tradable primitive/underlying assets;
- other (non-price) risk factors denoted $X_t = (X_t^1, ..., X_t^d)$. Examples of such risk factors may be: implied volatilities, interest rates, credit spreads, etc.

A generative model G should be capable of generating one-step ahead (e.g. one-day ahead) scenarios ω for the co-movements of prices S_t and risk factors X_t

$$a_t = (S_t, X_t, S_{t-\Delta t}, X_{t-\Delta t}, ...), \quad \omega \qquad \xrightarrow{\mathbf{G}} \quad (S_{t+\Delta t}(\omega), X_{t+\Delta t}(\omega)). \tag{1.1}$$

The input of the generative model $(S_t, X_t, ...)$ throughout this thesis is denoted by a_t, and it represents a summary of the market at time t. The generative model G therefore provides samples from a distribution similar to that of $(S_{t+\Delta t}, X_t)$ conditional on a_t, denoted by $\mathbb{P}_{a_t}^G$. Although the distribution $\mathbb{P}_{a_t}^G$ may not be available analytically, we aim to be able to efficiently sample from it. In order for the generative model G to be feasible, it is necessary for it to be able to handle conditional generation (having a direct input a_t) and to easily obtain i.i.d. samples from its learned probability distribution $\mathbb{P}_{a_t}^G$.

Although methods such as Path Shadowing Monte Carlo [104] have been proposed to transform unconditional generative models into conditional models by comparing the simulations to the observed data history, directly modelling the conditional one-step-ahead distribution offers a more efficient and interpretable solution. One-step-ahead simulations can support both local decision-making and path generation via recursive application in a Markovian framework. Moreover, when simulated scenarios are not used for data augmentation, but for downstream tasks such as risk management or portfolio construction, it becomes crucial to be able to rapidly generate a large number of i.i.d. samples from the conditional distribution $\mathbb{P}^G_{a_t}$ learned by the generative model. In addition, conditioning directly on current market information a_t allows faster and more adaptive responses to regime changes (such as the Covid-19 pandemic), without requiring global retraining or historical search procedures. As demonstrated throughout this thesis, the models presented here can generalise effectively across changing market conditions even without retraining, highlighting the robustness of the one-step-ahead conditional approach.

Practical requirements such as efficiency, adaptability, and conditional sampling highlight the importance of selecting an appropriate generative architecture. We now compare the suitability of different classes of generative models in addressing these challenges.

Sampling from GANs is highly efficient: once trained, the generator is a deterministic function, and samples can be drawn in a single pass by re-sampling the noise. In contrast, sampling from diffusion models typically involves simulating a discretised reverse stochastic process over many steps, which can be computationally expensive.

Conditional VAEs encode a dependency of the latent space on the conditioning input, which may complicate sampling consistency if latent priors differ in training and generation. This is particularly important if the condition is continuous, rather than belonging to a small discrete set, since it might be impossible to sample from the correct latent space. Conditional GANs [60, 103], in contrast, allow for a clean separation between latent priors and the input condition.

Contrary to GANs, which do not yield explicit likelihoods, EBMs directly model the unnormalised log-likelihood of the data via an energy function. However, sampling from them often requires MCMC techniques, which are computationally intensive and sensitive to initialisation. Neural SDEs provide a continuous-time generative framework well-suited to financial time series, but they often assume Brownian motion as the driving noise, limiting their flexibility for modelling non-Gaussian features like jumps or rough paths.

Flow-based models offer exact likelihood evaluation and efficient sampling, but their architecture is constrained by the requirement of invertibility and tractable Jacobian computation.

Autoregressive models, including large language models (LLMs), directly model conditional likelihoods, and have achieved significant success in natural language processing. However, they typically require millions of parameters and generate samples sequentially, making it less straightforward to obtain independent, identically distributed (i.i.d.) samples. When applying such models to financial problems, additional care must be taken to avoid issues such as lookahead bias [111], which can arise when the temporal structure of market data is not properly respected.

Given these trade-offs, this thesis focuses on conditional GANs, which offer a strong balance between flexibility, sampling efficiency, and conditional modelling. Although GANs are well-known for challenges such as mode collapse and training instability, we demonstrate that these limitations can be mitigated through bespoke training objectives. As shown in later chapters, these conditional GAN models perform robustly on real financial data, including high-dimensional and low-frequency settings such as daily returns.

Beyond purely data-driven approaches, another promising direction is hybrid generative–factor modelling, where data are first projected onto a small number of latent factors, whose dynamics are then learned using generative models. This structure provides better interpretability and can encode known economic relationships, though at the cost of potential restrictions on model adaptability. Recent examples include diffusion-based factor models [30] and GAN-based versions [26]. Although such hybrid approaches are not explored in this thesis, they represent a natural extension of the generative modelling frameworks developed here. Instead, our focus remains on learning directly from the data, without constraining the model through predetermined factor structures.

1.1.1 Generative Adversarial Networks

Generative adversarial networks (GANs) are often studied from a game-theoretic perspective, which is also the origin of their name. Before formalising the framework mathematically, we begin with an intuitive explanation.

Consider two players playing a minimax game: a generator G and a discriminator D. The generator attempts to produce samples that resemble real data, while the discriminator's task is to distinguish between real samples (from the data set) and samples simulated by the generator. The generator is trained to produce samples that

the discriminator classifies as indistinguishable from real data, while the discriminator learns to become better at classifying samples provided to it. Both G and D are implemented as neural networks.

In the case of conditional GANs [60, 103], the generation process is conditioned on some additional information, such as class labels or other conditioning variables. Both the generator and the discriminator receive the same conditioning input. For example, if a model aiming to generate photos of animals is tasked with generating images of cats, both G and D receive the label 'cat' as input. The generator produces an image that it believes to be a cat, and the discriminator checks whether the image looks like a real cat photo given the label. This setup is illustrated in Figure 1.1.

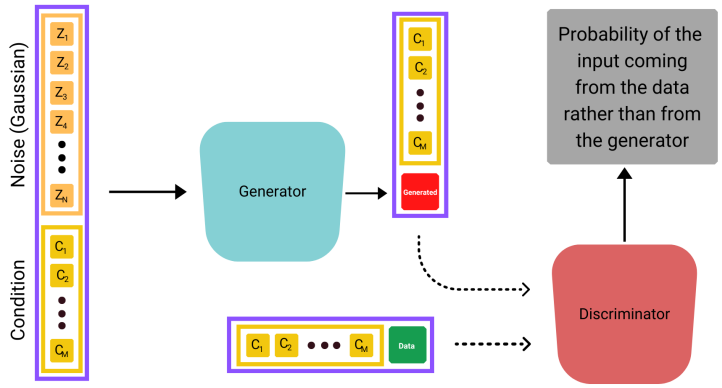


Figure 1.1: An illustration of a conditional GAN pipeline.

An alternative way to motivate the GAN framework is through the lens of generative modelling more broadly. Many generative models, such as diffusion models, VAEs, and flow-based models, are based on learning a pair of mappings: one from the data to a latent representation (often seen as noise) and one from the latent space back to the data domain. Once trained, new samples are generated by sampling from the latent (noise) distribution and applying the learned inverse mapping.

However, in the conditional setting, this bidirectional structure becomes more intricate. The mapping from data to latent variables must now account for the conditioning input, and the generative map must reconstruct data conditional on that same input. This introduces dependencies between the latent representation and the input condition, which can complicate sampling and lead to inconsistencies, particularly if the latent distributions differ between training and generation.

GANs offer an elegant alternative by completely decoupling the bidirectional structure. Instead of mapping data to its latent representation (noise) and back,

GANs begin directly with random noise and a conditioning input, and learn to map these into realistic samples. This is a more challenging task, as the generator does not receive direct access to the real data samples, but only indirect feedback via the discriminator's evaluations.

Training a GAN involves an adversarial game between the generator and the discriminator, introduced in the foundational work of [60]. Although elegant in theory, this setup poses practical challenges. Since the generator and discriminator are trained alternately (rather than jointly), convergence is not guaranteed, and training can suffer from instability or mode collapse.

We now formally define the conditional GAN framework [60, 103] used in this thesis, and discuss its advantages and limitations in the context of financial modelling.

Building on the previous discussion, we consider the task of learning the one-step-ahead evolution of market variables (S_t, X_t) over a horizon Δt, on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t), \mathbb{P})$, where the true data-generating distribution $\mathbb{P}$ is unknown. As before, we define $a_t \in \mathbb{R}^{l_a}$ as an $\mathcal{F}_t$-measurable vector summarising market information available at time t. The minimal form is $a_t = (S_t, X_t)$, but richer summaries such as $a_t = (S_t, X_t, S_{t-\Delta t}, X_{t-\Delta t}, \dots)$ are also possible. In general, a_t may include any $\mathcal{F}_t$-measurable feature of the market.

The objective is to approximate the conditional distribution of $(S_{t+\Delta t}, X_{t+\Delta t})$ given a_t, by learning directly from data.

Definition 1. *Let noise $Z \in \mathbb{R}^{l_z}$ be distributed according to $\mathbb{P}_z$, which is easy to independently sample from. The generator $G : \mathbb{R}^{l_a} \times \mathbb{R}^{l_z} \mapsto \mathbb{R}^{n+d}$ is a differentiable function that maps the market information space and the latent (noise) space to the data space. It is parameterised by a neural network whose parameters are denoted by θ_g. The value $G(a_t, Z; \theta_g) = (\hat{S}_{t+\Delta t}(Z), \hat{X}_{t+\Delta t}(Z))$ represents a simulated scenario for S and X at time $t + \Delta t$, conditional on the market information a_t at time t.*

Remark 1. *Typically, $\mathbb{P}_z$ is a standard multivariate normal distribution, $\mathcal{N}(0, I_{l_z})$. Conditional samples $(\hat{S}_{t+\Delta t}, \hat{X}_{t+\Delta t})$ are generated by drawing noise samples $Z_1, \dots, Z_N \overset{\text{iid}}{\sim} \mathbb{P}_z$, and using them as inputs of the generator to reach $\{G(a_t, Z_i; \theta_g)\}_{i=1,\dots,N}$. Hence, noise Z corresponds to a scenario ω in (1.1).*

Definition 2. *The discriminator $D : \mathbb{R}^{l_a} \times \mathbb{R}^{n+d} \mapsto (0, 1)$ is a differentiable function, represented by a neural network, of market information a_t at time t and a scenario $(S_{t+\Delta t}, X_{t+\Delta t})$ for time $t + \Delta t$. The parameters of the discriminator are denoted by θ_d. Its output $D(a_t, (S_{t+\Delta t}, X_{t+\Delta t}); \theta_d)$ is a real number between 0 and 1, interpreted as the probability that $(S_{t+\Delta t}, X_{t+\Delta t})$ comes from the data rather than the generator.*

These components form the basis for the conditional GAN framework used throughout this thesis.

The discriminator D is trained to maximise the probability of assigning the correct label to both the true data samples and the generated ones. It assesses the compatibility of the simulated scenario $G(a_t, Z; \theta_g)$ with the market state a_t.

The generator is trained to produce outputs to which the discriminator assigns a high probability of being real. While many alternative training objectives have been proposed [73], including zero-sum losses [60], f-divergences [107], and the Wasserstein-1 distance [5], we focus on binary cross-entropy, as both custom GAN-based models developed in this thesis are trained using this loss.

Definition 3. *The binary cross-entropy function for the discriminator is* $J^{(D)}(\theta_d, \theta_g)$ *is*

$$J^{(D)}(\theta_d, \theta_g) = -\frac{1}{2}\mathbb{E}\left[\log[D(a_t, (S_{t+\Delta t}, X_{t+\Delta t}); \theta_d)]\right] - \frac{1}{2}\mathbb{E}\left[\log[1 - D(a_t, G(a_t, Z; \theta_g); \theta_d)]\right]. \tag{1.2}$$

As discussed in [60], under ideal conditions, including infinite model capacity, optimal discriminator, and exact optimisation, the minimax GAN game with binary cross-entropy loss for the discriminator and $J^{(G)} = -J^{(D)}$ for the generator corresponds to minimising the Jensen–Shannon divergence between the true data distribution and the model distribution. The game converges to a Nash equilibrium if both policies can be updated in the function space, which is usually not the case in practice, as argued by [61]. When convergence takes place, the discriminator views all outputs as being equally likely to be real as they are to be simulated, therefore having its outputs converging to $\frac{1}{2}$. However, [60, 61] favour the cross-entropy loss over the zero-sum loss for the generator. Early during training, the zero-sum loss may not provide sufficiently large gradients for the generator to be able to learn well, since the simulated samples may be easily distinguished as such, resulting in saturated discriminator ratings, and low gradients.

The binary-cross entropy loss results in the generator maximising the log-score it receives from the discriminator.

Definition 4. *The binary cross-entropy loss for the generator is* $J^{(G)}(\theta_d, \theta_g)$ *is*

$$J^{(G)}(\theta_d, \theta_g) = -\frac{1}{2}\mathbb{E}\left[\log[D(a_t, G(a_t, Z; \theta_g); \theta_d)]\right]. \tag{1.3}$$

The parameters of the two networks are updated in an alternating manner, although sometimes one of the networks is updated more often. However, [60] argues for one-to-one alternating updates.

As already mentioned, there are several issues that commonly arise when training GANs [4]. The most prominent is *mode collapse*, which can be full or partial. Full mode collapse refers to the generator producing identical or nearly identical outputs, effectively providing point estimates only. Partial mode collapse denotes a situation in which the generator outputs a small number of distinct, but limited scenarios, failing to capture the full diversity of the target distribution.

In our setting, this would manifest as the distribution of simulated values degenerating towards a Dirac delta measure. Such behaviour can arise when the generator converges towards sharp local minima [53]. In this case, the generator may repeatedly produce samples close to real data, receiving high discriminator scores. This leads to vanishing gradients and negligible updates, ultimately resulting in mode collapse. A similar issue arises if the discriminator is overly strong, since the generator is not able to learn much due to weak gradients.

Regularisation of the discriminator is often used in a Wasserstein-GAN setting [5], when the Lipschitz constraint is softly enforced through a gradient penalty term [63]. However, it is also possible to regularise the generator to adapt to the task and data at hand, which is what we opt for when training both VolGAN [134] (Chapter 3) and Fin-GAN [136] (Chapter 5). However, such an approach has also been used for Ex-GAN [12], to simulate extreme weather events.

Despite their growing popularity in diverse fields, [87] shows that standard GANs in general do not reliably reproduce stylised facts of financial time series [38]. Hence, it is necessary to adapt GANs to tasks and to carefully examine whether the outputs of these models mimic these empirically observed properties of financial time series. Although visualisation may help detect major flaws, relying solely on visual inspection is not sufficient. Model validation should be performed on the basis of the task for which the simulated data would be used. For example, in Chapter 3 we carefully study the dynamics of the implied volatility co-movements produced by VolGAN, and in Chapter 5 we estimate Fin-GAN's trading capabilities.

There are very few data assumptions which are necessary in order to train a (conditional) GAN. As with any statistical model, there is an underlying stationarity and ergodicity assumption on the training data, in order to be able to approximate the population average of loss functions with sample means. This may require some data transformation or reparametrisation. For example, call/put prices with

fixed (absolute) strike and maturity do not satisfy this property, due to obvious time-dependence especially near expiry. On the other hand, implied volatility parameterised by 'moneyness' and time-to-maturity is more likely to be stationary. However, there are no other assumptions on the distributional form and the dynamics of the input and output data beyond stationarity and ergodicity.

In this thesis, we develop custom GAN architectures, VolGAN and Fin-GAN, along with the theoretical and practical tools necessary for their implementation in finance. Since implied volatility modelling must adhere to strict no-arbitrage constraints, the development of VolGAN begins with a framework capable of enforcing these constraints even when simulations originate from potentially black-box models. Once VolGAN is established, a general methodology for data-driven hedging is derived, and its effectiveness in dynamically hedging option portfolios is demonstrated. Similarly, after introducing Fin-GAN, the thesis explores its extension to leverage cross-sectional information. The methodology in this thesis highlights the interplay between modern machine learning techniques and traditional mathematical and statistical methods.

1.2 Outline and contributions

This thesis develops generative modelling frameworks tailored to high-dimensional financial applications, addressing simulation, hedging, and forecasting in multi-asset markets. Its contributions include the design of custom conditional GANs, validation on real-world financial data, and integration with downstream financial decision-making tasks.

Across the chapters, we develop novel conditional generative models, including VolGAN and Fin-GAN, and demonstrate their effectiveness in generating arbitrage-free implied volatility surfaces, forecasting return distributions, and data-driven hedging.

A central theme is the fusion of finance, data-driven modelling, and traditional mathematical and statistical approaches such as LASSO regression [129]. We enforce arbitrage constraints via post-training scenario re-weighting, incorporate task-specific objectives into GAN training, and combine generative models with financially motivated optimisation approaches. Our results show how probabilistic generative models can be successfully deployed across a wide range of financial tasks. The methodology is empirically validated on real market data, demonstrating improved hedging per-

formance, Sharpe Ratios, and generalisation across assets, especially in low-sample regimes.

This work contributes to the growing literature on generative modelling in finance, transfer learning for time series, and data-driven risk management, and it proposes scalable alternatives to more rigid parametric or reinforcement-learning-based methods.

Beyond methodological contributions, this thesis addresses both theoretical and practical challenges in quantitative finance. It contributes new modelling tools that overcome the limitations of traditional modelling approaches, offering more flexible and efficient alternatives rooted in modern generative modelling. It shows how domain knowledge can be systematically incorporated into the learning process.

The models are also designed with practical deployment in mind, particularly in settings with limited data, minimal rebalancing, or changing market regimes. Their flexibility and robustness make them suitable for integration into real-world risk management or trading systems.

We now provide an overview of the chapters in this thesis.

1.2.1 Chapter 2: 'Simulating arbitrage-free implied volatility surfaces'

Market prices of options are expressed via their Black-Scholes implied volatilities, derived by inverting the Black-Scholes formula given the option's market price. In numerous options markets, it is widely observed that the implied volatility $\Sigma_t(K,T)$ for a call option with strike price K and maturity T is actually dependent on the values of (K,T) [41, 51, 52, 67, 57]. The function $\Sigma_t : (K,T) \to \Sigma_t(K,T)$, which illustrates this relationship, is known as the *implied volatility surface* at time t and provides a representation of the options market pricing structure [79]. Figure 1.2 shows an example for SPX index options.

Two characteristics of this surface have drawn interest from researchers in financial modelling. Firstly, the non-uniform instantaneous profile, whether it exhibits a 'smile', 'skew', or term structure, highlights the shortcomings of the Black-Scholes model in fitting a set of option prices at any specific moment. This has inspired various extensions of the Black-Scholes model that strive to replicate realistic instantaneous profiles for the surface $\Sigma_t(K,T)$. Secondly, the fact that the surface itself fluctuates unpredictably over time due to supply and demand dynamics in the options market implies that a good risk management model must not only accommodate the shape of the surface at a particular date but also provide realistic dynamics for co-movements of implied volatilities across different strikes and maturities.

Market models of implied volatility [8, 21, 37, 41, 42, 58, 122, 120, 99, 3] attempt to directly capture both the cross-sectional characteristics and the evolution over time of implied volatilities. One of the challenges in modelling implied volatility surfaces is ensuring compliance with static arbitrage constraints. The implied volatility surface cannot be unrestricted: static arbitrage constraints on the values of call and put options [47] impose conditions on its possible shape. Analytical approaches have concentrated on developing parameterisations for implied volatility surfaces that inherently comply with these arbitrage constraints [21, 122, 37]. Nevertheless, deploying these models is computationally intensive, and calibrating them to achieve realistic surface dynamics proves even more challenging.

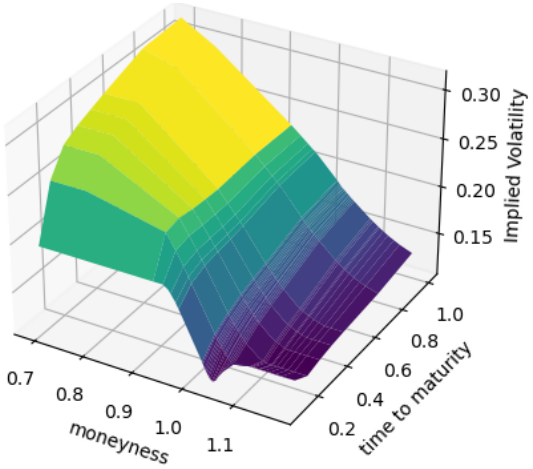


Figure 1.2: SPX implied volatility surface on 01/11/2021.

We present a computationally tractable method for simulating arbitrage-free implied volatility surfaces, which correctly captures the co-movements of implied volatility across a range of strikes and maturities. We first perform data analysis on the SPX implied volatility surface, and we then illustrate how our method may be combined with a factor model for the implied volatility surface to generate dynamic scenarios for arbitrage-free implied volatility surfaces. We give two examples: a stylised model using basis functions representing level, skew and curvature, and a data-driven example based on principal component analysis of daily changes in the logarithm of the SPX implied volatility surfaces. Our approach conciliates static arbitrage constraints with a realistic representation of statistical properties of implied volatility co-movements.

The empirical study in this chapter presents a number of statistical properties that any good model for implied volatility simulation should satisfy. Furthermore, we

introduce an arbitrage penalty that quantifies potential static arbitrage violations. The arbitrage scenario re-weighting, which is a Weighted Monte Carlo Approach [6] based on the arbitrage penalty, introduced in this chapter, enables the use of complex machine learning models without requiring the strict enforcement of no-arbitrage constraints in the model architecture itself.

1.2.2 Chapter 3: 'VolGAN- a generative model for arbitrage-free implied volatility surfaces'

Given the high dimensionality of the volatility surface and the complexity of its dynamics, it is challenging to capture all of the properties discussed in the previous chapter in a parametric model. It is therefore of interest to examine whether a data-driven approach can be used to overcome these modelling challenges.

We introduce VolGAN, a *fully data-driven* generative model for the dynamic simulation of arbitrage-free implied volatility surfaces. The model is trained on a time series of market-quoted implied volatilities and is capable of generating realistic dynamic scenarios for implied volatility surfaces. We illustrate the performance of the model by training it on SPX implied volatility time series and show that it is able to learn the covariance structure of co-movements in implied volatilities and generate realistic dynamics for the (VIX) volatility index [25]. In particular, the generative model is capable of simulating scenarios with non-Gaussian increments, as well as time-varying correlations.

Our model builds on previous work on generative adversarial networks (GANs) for scenario simulation in finance, starting with [128] and [139] for price dynamics. More recently, GANs have been deployed for scenario simulation in options markets. [138] use a classical GAN approach. [46] and [35] use a 'neural SDE' to parameterise volatility surface dynamics. [19] use a supervised learning approach to extract information from historical implied volatility dynamics, while [106] combines SDEs with Variational Autoencoders [84].

In contrast with the aforementioned approaches which deploy the classical GAN methodology of [60] using binary cross-entropy (BCE) as a training objective (1.2)-(1.3), we propose a bespoke training criterion adapted to the financial application at hand, combined with a scenario re-weighting [45] approach introduced in Chapter 2 to take care of arbitrage constraints.

Chapter 2 provided a modelling baseline and a number of statistical properties that a good implied volatility model should satisfy. As one of the main concerns around GANs is stability, we perform out-of-sample tests over a four and a half

year long period spanning both times of market turbulence and calm, without any retraining. In order to show that VolGAN learns the dynamics of implied volatility co-movements, we design a number of model validation tests, based on the empirical study from Chapter 2. To our knowledge, this is one of the first generative models in the literature that is empirically validated for producing arbitrage-free surfaces with realistic dynamics.

1.2.3 Chapter 4: 'Data-driven hedging with generative models'

Having shown that VolGAN can realistically simulate dynamic implied volatility surfaces in the previous chapter, we explore how such generative models can be used in downstream applications such as hedging and risk management.

In this chapter we propose a nonparametric data-driven methodology for hedging using generative models. The key idea is to learn the co-movements of potential hedging instruments and the target portfolio from the market data, and then use this information to compute hedging strategies.

In contrast with model-based hedging approaches relying on sensitivity analysis of model-based pricing functions, our approach uses a conditional generative model trained on market data to simulate realistic market scenarios given current market conditions, and computes hedge ratios that minimise risk across these scenarios. This optimisation procedure enables the automated selection of hedging instruments, and allows for the incorporation of transaction costs and market impact.

The result is a fully data-driven method for implementing hedging strategies: market data is used to train a generative model, which is then employed to compute hedge ratios. Opting for conditional variance minimisation, as opposed to more general risk measures, leads to linear hedge ratios which are easily computable via linear regression. We illustrate the effectiveness of this methodology for hedging option portfolios using VolGAN [134], and compare its performance with delta and delta-vega hedging.

1.2.4 Chapter 5: 'Fin-GAN: forecasting and classifying financial time series via generative adversarial networks'

Although Chapter 4 focused on risk management via generative models, forward-looking scenarios obtained from generative models can also be used for forecasting, a task typically performed through a supervised learning approach. Since trading

strategies are constructed based on informed beliefs in the direction and magnitude of future market prices movements, providing uncertainty estimates and being able to model the full conditional distribution of future returns are highly beneficial.

However, in a trading context, it is necessary to convert the information encapsulated by the learned distribution of the future returns into actual trades. The two main approaches to do so are regression-based and classification-based trading. The former places a trade proportional to the expectation of the forecast. This approach does not take into account any other distributional information, so a low-confidence forecast may still lead to large trades.

Instead, the classification approach focuses on the expected direction of the asset move, i.e. on the expected sign of the forecast. This alternative formulation accounts for the uncertainty of the forecast. If the probability forecast probability distribution is equally supported on positive and negative values, there will be no trades. This is the same approach as the one used in [136], labelled as the *weighted strategy*.

There is evidence suggesting that classifying the direction of returns can result in more robust trading strategies [92]. While we opt for the certainty in the direction, generative models allow for both approaches, by modelling the full conditional distribution of returns, from which directional signals (e.g., expected sign) or level-based forecasts (e.g., expected return) can be derived.

As we demonstrate in Chapter 5, even a GAN that is built for time series forecasting, such as ForGAN [85], results in mode collapse when trained on equity returns via the binary cross-entropy loss (1.2)-(1.3), and leads to poor financial performance. To alleviate these issues, we introduce Fin-GAN, by adapting to the classification setting of forecasting the directionality of the asset movements. To do so, we introduce a custom economics-driven loss function for the generator, ultimately placing Fin-GAN in a supervised learning setting, while still obtaining distributional forecasts through adversarial training.

We compare our approach with a commonly used deep learning method for time-series forecasting, LSTM [70], and with a standard time-series model used in econometrics, ARIMA [131]. We further benchmark our results against long-only strategies and against the same ForGAN architecture trained via the binary cross-entropy loss. Additionally, we investigate the effect of the economics-driven loss function terms on the LSTM, denoting the approach by LSTM-Fin. Although the loss function is highly beneficial in the adversarial training setting of Fin-GAN, it actually worsens the performance of an LSTM. This highlights how GANs can benefit from rich, non-convex loss functions that encode domain-specific structure. Unlike

standard supervised learning methods (e.g., LSTMs), where such losses may hinder convergence, adversarial training enables GANs to handle these complexities more robustly.

An extensive set of numerical results demonstrate that our proposed methodology attains superior performance on daily stock ETF-excess returns and on daily ETF raw returns, when compared to ARIMA, LSTM, LSTM-Fin, and ForGAN trained using the BCE loss.

The aforementioned loss function shifts the generated data distributions, performing classification and providing uncertainty estimates on the direction of the movement. In addition, the Fin-GAN loss helps alleviate mode collapse.

Our approach allows for scalability, and potentially modelling of the joint co-movements of different stocks. We explore the notion of universality, in the spirit of [125]. We consider 22 different stocks, across different sectors, and 9 sector ETFs, with each industry sector having at least two different stocks included in the training data. We train the networks on the data set created by pooling together the data on all 31 tickers. We perform a similar experiment using stocks that belong to a single sector (XLP). We test our models both on the stocks included in the training set, and on stocks not seen by the model during the training stage. We find that even for the latter category of new stocks, it is possible to achieve significant Sharpe Ratios [124].

1.2.5 Chapter 6: 'Graph-based ensemble generative modelling for multi-asset forecasting'

While Chapter 5 showcases how a custom conditional GAN might be designed and trained in a forecasting setting, cross-asset interactions are not addressed. Chapter 6 develops a scalable ensemble forecasting framework for financial time series that balances the benefits of stock-specific models and universal settings[125]. Forecasting in multi-asset settings poses a trade-off: universal models may overlook asset-specific dynamics, while stock-specific models fail to exploit cross-sectional relationships. To address this, we introduce a hybrid strategy that leverages cross-asset relationships through a directed graph-based probabilistic ensemble of generative models.

This framework operates by independently training a generative model on each asset, then constructing a directed graph that encodes generalisation performance between models and assets. Using this graph structure, we form meta-generators by combining probabilistic forecasts from neighbouring nodes, enabling information sharing only where transfer is empirically validated. This structure captures asymmetric and

non-stationary relationships between assets, avoiding the rigidity of hard clustering or pooled training.

We learn the ensemble weights through a sparsity-inducing LASSO-based optimisation procedure grounded in financial performance, ensuring interpretability and scalability in large universes. The resulting meta-generators (generative ensembles) produce full conditional distributions of returns, from which both direction and magnitude of moves can be extracted. This allows for uncertainty-aware trading decisions, outperforming baseline approaches including self-forecasting models and correlation-driven ensembles.

We demonstrate the method using the Fin-GAN framework from Chapter 5, trained on a large panel of U.S. equities. Our results highlight the benefits of cross-asset generalisation, outperforming baselines in terms of Sharpe Ratio, successfully leveraging cross-sectional information.

Chapter 2

Simulating arbitrage-free implied volatility surfaces

2.1 Introduction

The possible shapes of implied volatility surfaces are constrained by static no-arbitrage conditions [47, 57]. While parametric models incorporate these constraints by design, they often lead to unrealistic or intractable dynamics, and can be computationally demanding to simulate. In contrast, data-driven approaches, including analytical factor models that reproduce statistical features of historical data, struggle to enforce arbitrage constraints in a straightforward way. This creates the need for a methodology that allows arbitrage-free simulation of data-driven models.

Any option pricing model inherently defines a dynamic model for the implied volatility surface. However, these dynamics are frequently complex and intractable. So-called "market models" of implied volatility instead aim to model the surface directly, targeting the co-movements of implied volatility across strikes and maturities while preserving no-arbitrage conditions. Achieving this balance has remained a long-standing and challenging problem in the literature for more than two decades.

Statistical models of implied volatility dynamics [41, 7] are designed to capture empirical co-movements and other statistical properties of the market data. These models are computationally efficient and are widely used in risk management applications. However, they can produce implied volatility surfaces that violate arbitrage constraints.

In parallel, several analytical models have been developed to satisfy static [58, 3, 99, 147] and dynamic [122, 140, 21, 37] arbitrage constraints. While these models are theoretically sound, they are often difficult to implement, simulate, or calibrate in practice.

In this chapter, which is based on the article [45], we introduce a computationally tractable method for simulating arbitrage-free implied volatility surfaces that accurately captures the co-movements of implied volatility across strikes and maturities. We begin with an empirical analysis of the SPX implied volatility surface, then demonstrate how our methodology can be combined with a factor model to generate dynamic, arbitrage-free scenarios.

We provide two illustrative examples: a stylised model using basis functions representing level, skew, and curvature; and a data-driven model based on principal component analysis of daily changes in the logarithm of SPX implied volatility surfaces. Our approach reconciles static arbitrage constraints with a realistic representation of the statistical properties of implied volatility co-movements. Notably, it is compatible with deep-learning, data-driven, models that replicate statistical properties of market data, without requiring strict enforcement of arbitrage constraints during training.

Outline Section 2.2 introduces notation for implied volatility surfaces and reviews key properties that market models aim to capture. Section 2.3 presents an empirical analysis of SPX implied volatility data, showing that a small number of principal components explain most of the variation. Section 2.4 reviews static arbitrage constraints and introduces a penalty function to quantify violations. Section 2.5 proposes a Weighted Monte Carlo method [6] that uses this penalty function to filter arbitrage-violating scenarios generated from a base model. Finally, Section 2.6 illustrates how this framework can be applied to a factor model of the implied volatility surface, such as the one introduced in [41].

2.2 Implied volatility surfaces

2.2.1 Properties of implied volatility surfaces

Denote the price of the underlying asset by S, strike price by K, expiry by T, and current time by t. The implied volatility may be parameterised in terms of moneyness $m = K/S$ and time to maturity $\tau = T - t$ of the option. The implied volatility associated with a call option with moneyness m and time-to-maturity τ on a non-dividend paying asset S is the unique value $\sigma(m, \tau)$ such that the Black-Scholes price $C_{BS}(S, K, \tau, \sigma(m, \tau))$ matches the market price $C(m, \tau)$ of the call:

$$\begin{aligned} C_t(m,\tau) = C_{BS}(S,K,\tau,\sigma_t(m,\tau)) &= SN(d_1) - Ke^{-r\tau}N(d_2) \\ d_1 = \frac{-\log m + \tau(r + \frac{\sigma^2}{2})}{\sigma\sqrt{\tau}} \qquad & d_2 = \frac{-\log m + \tau(r - \frac{\sigma^2}{2})}{\sigma\sqrt{\tau}}, \end{aligned}$$

where N is the c.d.f of a standard Gaussian $\mathcal{N}(0,1)$ variable, and r is the risk-free interest rate. The implied volatility surface $\sigma_t(m,\tau)$ at date t provides a snapshot of options prices in the market [57]: specifying the implied volatility surface is equivalent to specifying the prices of all European calls and puts available in the market, given the current term structure of interest rates and dividends.

This representation proves to be practical because, typically, there exists a range of moneyness near $m = 1$, where options exhibit high liquidity, making empirical data more accessible.

The possible shapes of implied volatility surfaces are limited by the arbitrage constraints on option prices [47]. Call prices should be:

- increasing in time to maturity: $\partial_\tau C_{BS}(S, K, \tau, \sigma(m,\tau)) \geq 0$,
- decreasing in moneyness: $\partial_m C_{BS}(S, K, \tau, \sigma(m,\tau)) \leq 0$,
- convex in moneyness: $\partial_m^2 C_{BS}(S, K, \tau, \sigma(m,\tau)) \geq 0$.

These constraints translate to nonlinear inequalities involving $\sigma(m,\tau)$, $\partial_m\sigma(m,\tau)$, $\partial_m^2\sigma(m,\tau)$, $\partial_\tau\sigma(m,\tau)$ [42]. The resulting constraints on the implied volatility surface $\sigma(m,\tau)$ and the appropriate derivatives impose restrictions on the possible shapes.

Empirical studies of the behaviour of implied volatilities of exchange-traded options on various market indices (SP500, FTSE, DAX and others) point to many common statistical properties across markets [7, 41], which we summarise here and demonstrate on SPX data from 2000 to 2021 in the following subsection:

- The implied volatility has a non-flat cross-section, and exibits both strike and term structure.
- The shape of the implied volatility surface undergoes deformation in time.
- Implied volatilities display high positive autocorrelation and mean-reverting behaviour.
- Daily variations in the implied volatilities can be satisfactorily explained with a small number of principal components.
- The first principal component corresponds to an overall shift in the *level* of all implied volatilities.
- The second principal component corresponds to a *skew* factor. It reflects opposite movements in (out-of-the money) call and put implied volatility.

- The third and fourth principal components reflect the term structure and the changes in convexity of the implied volatility surfaces.

- The returns of the underlying are negatively correlated with the projections of log-increments of implied volatility on the *level* and *skew* principal components, which is a more precise formulation of the so-called 'leverage effect'.

These dynamical properties of co-movements of implied volatilities and the underlying have important implications for hedging and should be reflected in any model used for risk management.

In the remainder of the chapter, we describe an approach which aims to conciliate computational tractability, arbitrage constraints and realistic dynamics for the surface, and demonstrate its performance in two examples.

2.3 Case study: dynamics of the SPX implied volatility surface

We consider a grid $(\boldsymbol{m}, \boldsymbol{\tau})$ with 10 equispaced moneyness values between 0.6 and 1.4, and 8 time-to-maturity values of 30, 60, 91, 122, 152, 182, 273, 365 calendar days. We use daily time series of implied volatility for SPX options from the OptionMetrics SPX Implied Volatility Surface File for the period 2000-2021. Surfaces are interpolated linearly first in moneyness, and then in time to maturity to yield values on the grid $(\boldsymbol{m}, \boldsymbol{\tau})$. The average SPX implied volatility profile $\overline{\sigma}$ shown in Figure 2.1.

We note that the Implied Volatility Surface File is based on a previous interpolation of listed option prices so may not necessarily be arbitrage-free, as already noted in [37]. We perform principal component analysis on the daily changes in the logarithm of the implied volatility surface

$$Y_t(m, \tau) = \log \sigma_t(m, \tau) \tag{2.1}$$

using a Karhunen-Loève decomposition [41]. We denote by f_i the eigenvectors of the covariance operator of $\Delta Y_t = Y_{t+\Delta t} - Y_t$ ordered by decreasing eigenvalue. Each eigenvector may be represented as a function $(m, \tau) \mapsto f_i(m, \tau)$ of moneyness and time to maturity. We project $Y_t - \tilde{Y}$, where $\tilde{Y} = \log \overline{\sigma}(m, \tau)$, onto the eigenbasis:

$$Y_t(m, \tau) = \tilde{Y}(m, \tau) + \sum_{i=1}^{k} X_t^i f_i(m, \tau) + \epsilon_t(m, \tau), \tag{2.2}$$

where

$$X_t^i = \langle Y_t - \tilde{Y}, f_i \rangle = \sum_{(m,\tau)\in(\boldsymbol{m},\boldsymbol{\tau})} \Big(Y_t(m,\tau) - \tilde{Y}(m,\tau) \Big) f_i(m,\tau) \tag{2.3}$$

and $\epsilon_t(m,\tau)$ is the projection error. The rank-k approximation of implied volatility dynamics is given by

$$\sigma_t(m,\tau) \approx \overline{\sigma}(m,\tau) \exp\left[\sum_{i=1}^{k} X_t^i f_i(m,\tau) \right]. \tag{2.4}$$

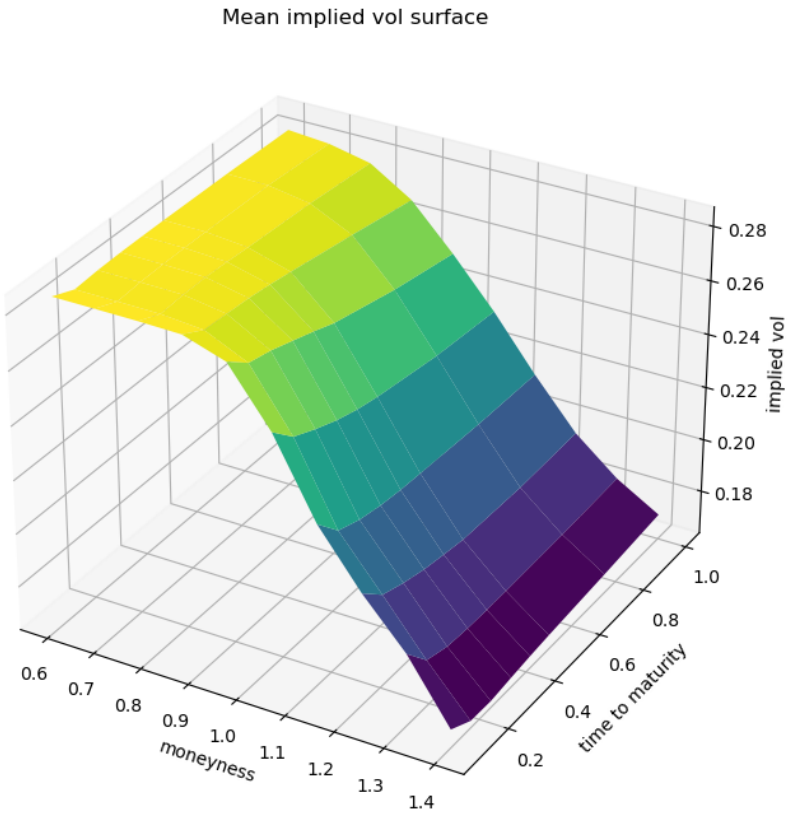


Figure 2.1: Average SPX implied volatility surface (2000-2021).

2.3.1 Principal component analysis

To determine the number k of significant factors we consider the eigenvalues of the correlation matrix of the daily log-variations in the SPX implied volatility surface ΔY_t and compare them with the corresponding Marčenko-Pastur threshold λ_+ [7, 50]:

$$\lambda_+ = \left(1 + \sqrt{\frac{N}{M}} \right)^2$$

where N is the number of points on the grid ($N = N_m N_\tau$), and M is the number of observations. We treat the eigenvalues below λ_+ as statistically insignificant.

As shown in Figure 2.2, there are $k = 4$ eigenvalues clearly above the Marčenko-Pastur threshold. The first four principal components of the daily changes in the log SPX implied volatility surface explain over 90% of the variance in the corresponding data.

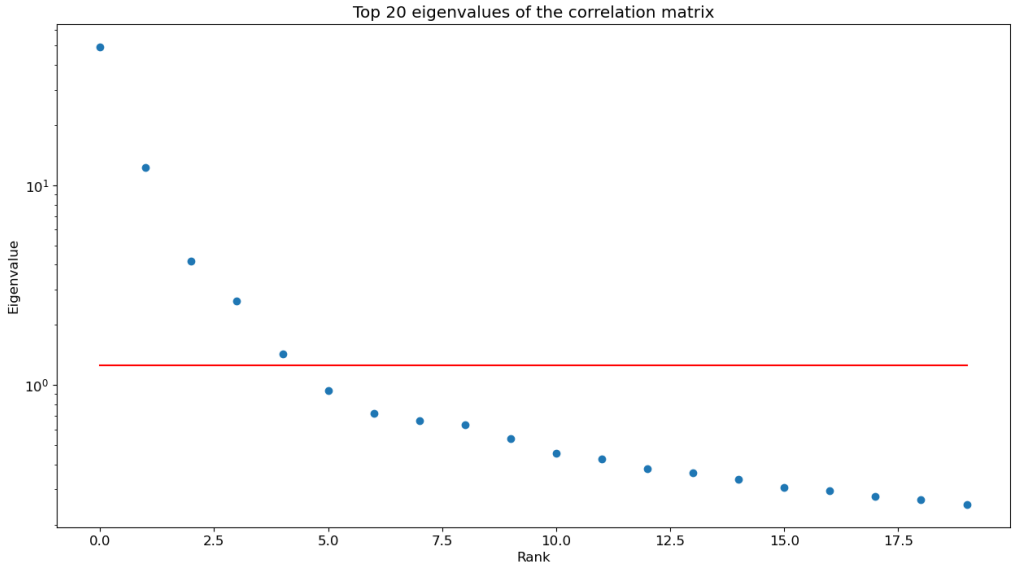


Figure 2.2: Eigenvalues of the correlation matrix of the daily changes in the log SPX implied volatility surface and the Marčenko-Pastur threshold λ_+ (in red).

	PC1	PC2	PC3	PC4	PC5
Variance explained (covariance)	68.88%	12.17%	5.67%	2.86%	1.41%
Cumulative variance explained (covariance)	68.88%	82.05%	87.72%	90.59%	92.00%
Cumulative variance explained (correlation)	61.39%	76.67%	81.88%	85.18%	86.96%

Table 2.1: Variance explained by the first five eigenvectors of the covariance and the correlation operator of the daily log returns of SPX implied volatilities.

The first four eigenfunctions are shown in Figure 2.3. All four significant principal components f_1, f_2, f_3, f_4 (Eqn. (2.4)) have natural interpretations:

- The first principal component can be interpreted as the average *level* of implied volatilities. A positive shock along this mode would result in a global shift in implied volatility.

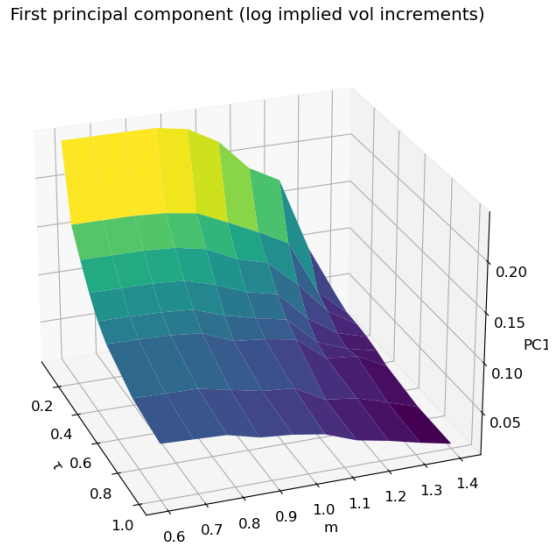
- The second principal component corresponds to the *skew*. Shocks along this direction result in opposite movements in out-of-the-money call and put implied volatilities.

- The third principal component reflects the *term structure* of implied volatilities.

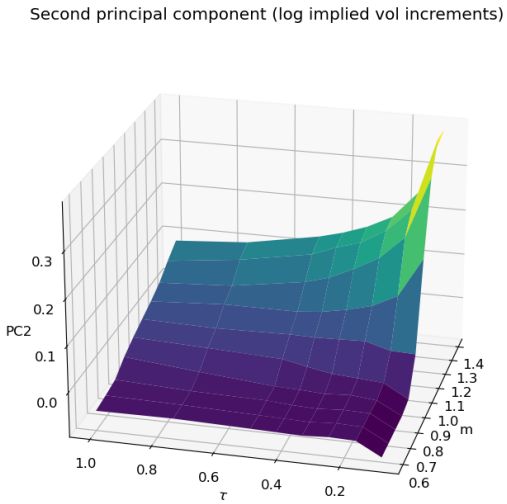
- The fourth principal component can be interpreted as *curvature*. A positive shock along this mode would have an opposite effect on the implied volatilities close to the money and on the far out-of-the-money and in-the-money implied volatilities.

Projections of $Y_t(m,\tau) - \tilde{Y}_t(m,\tau)$ onto the first four principal components X_t^i, $(i = 1, \ldots, 4)$ (defined by Eqn. (2.3)) are shown in Figure 2.4. All processes exhibit high positive autocorrelation and mean-reverting behaviour with a period of mean reversion of several months. The ACF and PACF of X_t^1 are shown in Figure 2.5. As shown in Figure 2.5 the autocorrelation functions decay exponentially, suggesting that they can be modelled as Ornstein-Uhlenbeck/AR(1) processes, as already observed by [41].

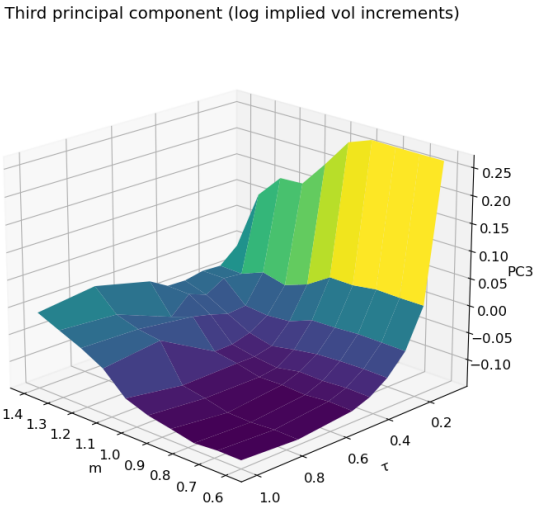
Correlations of daily increments $\Delta X_t^i = X_{t+\Delta t}^i - X_t^i$ $(i = 1, \ldots 4)$ and the log-returns of the underlying $R_t = \log S_{t+\Delta t} - \log S_t$ over a two-year rolling window are shown in Table 2.2. We note that the log-returns R_t are negatively correlated with $\Delta X_t^1, \Delta X_t^2, \Delta X_t^4$ while there is a positive correlation between the log-returns and the increments of the term-structure process ΔX^3.



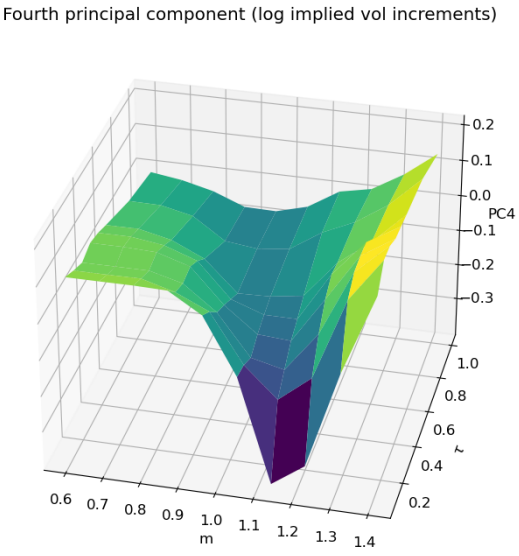
(a) Level



(b) Skew



(c) Term structure



(d) Curvature

Figure 2.3: The first four principal components of the daily changes in log SPX implied volatility surface.

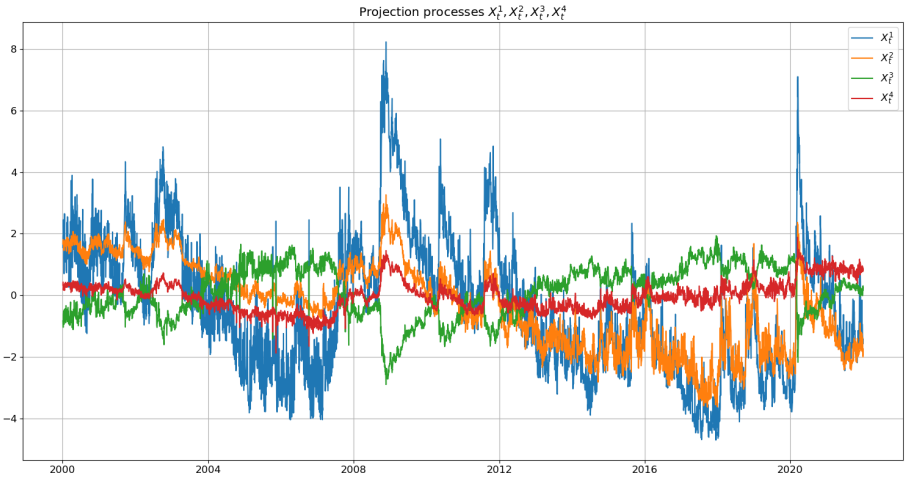


Figure 2.4: Principal component processes $X_t^1, X_t^2, X_t^3, X_t^4$.

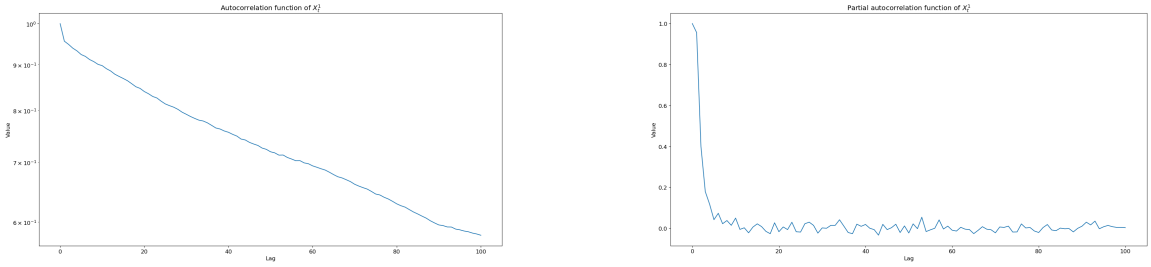


Figure 2.5: Autocorrelation and partial autocorrelation of the log implied volatility projection on the first principal component. The autocorrelation function (above) in logarithmic scale shows an exponential decay characteristic of OU processes.

Correlation between R_t and	ΔX_t^1	ΔX_t^2	ΔX_t^3	ΔX_t^4
Long-term	-38.21%	-25.64%	26.42%	-26.39%
Rolling: mean	-48.83%	-37.75%	31.60%	-32.63%

Table 2.2: Long-term and 2-year rolling daily correlation between the log-increments of SPX and the increments of $X_t^1, X_t^2, X_t^3, X_t^4$ (Eqn. (2.3)).

2.3.2 Relationship with the VIX

The CBOE Volatility Index (VIX) is constructed as a nonlinear combination of out-of-the-money tradable calls and puts with one month to expiry [25, 22]. We investigate the relationship between the VIX and different variables of interest by considering the historical closing VIX prices available on the CBOE website. Figure 2.6 displays the correlations between the log returns of VIX, one-month at-the-money SPX implied

volatility, SPX and the increments of the level process X_t^1 over a rolling 2-year window. We note a high positive correlation between the level movements and the log-returns of VIX and ATM vol. Similarly, the returns of the underlying are negatively correlated with the increments of the level process (Table 2.2), log-returns of VIX and with the log-returns of the ATM vol. Correlations increasing in magnitude from 2006 onwards.

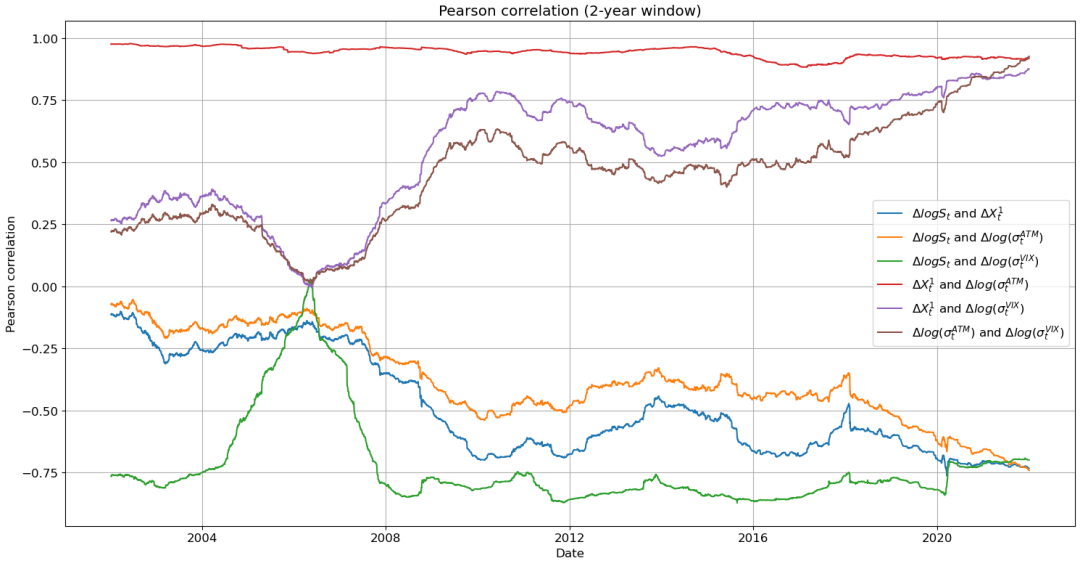


Figure 2.6: Correlation over a 2-year window between daily changes in the level process X_t^1 and daily log-returns of one-month ATM vol, VIX, and SPX.

The one-month realised volatility $\hat{\sigma}_t$ is estimated as

$$\hat{\sigma}_t = \sqrt{\frac{21}{252}\sum_{i=0}^{20} R_{t-i\Delta t}^2}. \tag{2.5}$$

Figure 2.7 shows that the realised volatility is usually below the implied volatility. The average ratio of realised volatility to ATM volatility is 0.59, with a standard deviation of 0.209 (Figure 2.7).

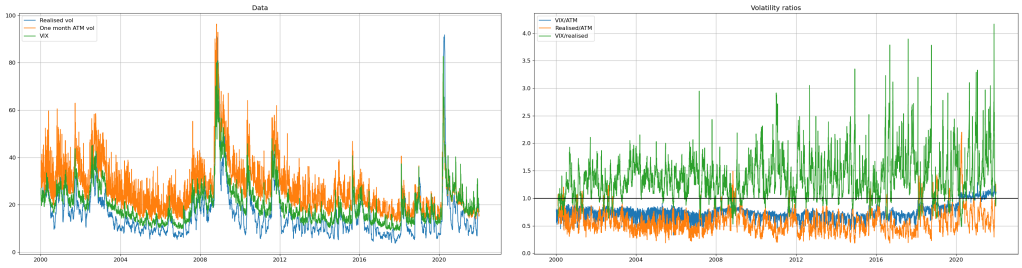


Figure 2.7: Above: SPX realised volatility (blue), one-month ATM volatility (orange) and VIX (green). Below: ratio of VIX to one-month ATM volatility (blue) and ratios of 21-day realised volatility to one-month ATM volatility, VIX to ATM volatility, and VIX to realised volatility.

2.4 Static arbitrage constraints

We now consider shape constraints on the implied volatility surfaces arising from static arbitrage inequalities [47]. We are interested in a realistic setting where only a finite number of options are available. We fix a grid in moneyness and time to maturity $(\boldsymbol{m}, \boldsymbol{\tau}) = (m_i, \tau_j)_{i=1,\dots,N_m; j=1,\dots N_\tau}$, with $m_i < m_{i+1}$ and $\tau_j < \tau_{j+1}$ for all i, j. Using the notation introduced in Section 2.2, denote by

$$c(m, \tau) := \frac{1}{S} C_{BS}(S, K, \tau, \sigma) = N(d_1) - me^{-r\tau} N(d_2)$$

the relative call price, which is a dimensionless quantity with $0 \leq c(m, \tau) \leq 1$.

2.4.1 Arbitrage constraints and arbitrage penalty

As shown by Davis and Hobson (Corollaries 4.2 and 4.3 in [47]), absence of static arbitrage among options with strikes and maturities defined by $(\boldsymbol{m}, \boldsymbol{\tau})$ is equivalent to the following three conditions:

1. Absence of calendar spread arbitrage:

$$\tau_j \frac{c(m_i, \tau_j) - c(m_i, \tau_{j+1})}{\tau_{j+1} - \tau_j} \leq 0, \tag{2.6}$$

   for $j = 1, \dots, N_\tau - 1$ and $i = 1, \dots, N_m$.

2. Absence of call spread arbitrage:

$$\frac{c(m_{i+1}, \tau_j) - c(m_i, \tau_j)}{m_{i+1} - m_i} \leq 0 \tag{2.7}$$

   for $i = 1, \dots, N_m - 1$ and $j = 1, \dots, N_\tau$.

3. Absence of butterfly spread arbitrage:

$$\frac{c(m_i,\tau_j)-c(m_{i-1},\tau_j)}{m_i-m_{i-1}}-\frac{c(m_{i+1},\tau_j)-c(m_i,\tau_j)}{m_{i+1}-m_i}\leq 0 \tag{2.8}$$

for $i=2,\ldots,N_m-1$ and $j=1,\ldots,N_\tau$.

Conversely, a non-zero positive part of the left-hand side of these inequalities indicates the presence of static arbitrage. We investigate whether an implied volatility surface $\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})$ is arbitrage-free by considering the inequalities (2.6), (2.7) and (2.8).

Hence, for an implied volatility surface $\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})$, we define the **arbitrage penalty** $\Phi\left(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})\right)$ as

$$\Phi\left(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})\right)=p_1(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau}))+p_2(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau}))+p_3(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})), \tag{2.9}$$

where

$$p_1(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau}))=\sum_{i=1}^{N_m}\sum_{j=1}^{N_\tau}\left(\tau_j\frac{c(m_i,\tau_j)-c(m_i,\tau_{j+1})}{\tau_{j+1}-\tau_j}\right)^+, \tag{2.10}$$

$$p_2(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})=\sum_{i=1}^{N_m}\sum_{j=1}^{N_\tau}\left(\frac{c(m_{i+1},\tau_j)-c(m_i,\tau_j)}{m_{i+1}-m_i}\right)^+, \tag{2.11}$$

$$p_3(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau}))=\sum_{i=1}^{N_m}\sum_{j=1}^{N_\tau}\left(\frac{c(m_i,\tau_j)-c(m_{i-1},\tau_j)}{m_i-m_{i-1}}-\frac{c(m_{i+1},\tau_j)-c(m_i,\tau_j)}{m_{i+1}-m_i}\right)^+. \tag{2.12}$$

The quantities p_1,p_2,p_3 correspond to deviations from the calendar, call, and butterfly spread arbitrage constraints, respectively. They are the positive parts of the left-hand sides of the inequalities (2.6), (2.7), and (2.8). If p_1,p_2,p_3 are all equal to zero, there is no arbitrage. Conversely, if any of p_1,p_2,p_3 are non-zero, then there is arbitrage present in $\boldsymbol{\sigma(m,\tau)}$. Therefore,

$$\Phi\left(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})\right)=0 \iff \boldsymbol{\sigma(m,\tau)} \text{ is arbitrage-free.}$$

We introduce the $N_m\cdot N_\tau$ penalty matrices P_1,P_2,P_3 defined as

$$(P_1)_{i,j}=\left(\tau_j\frac{c(m_i,\tau_j)-c(m_i,\tau_{j+1})}{\tau_{j+1}-\tau_j}\right)^+,\quad (P_2)_{i,j}=\left(\frac{c(m_{i+1},\tau_j)-c(m_i,\tau_j)}{m_{i+1}-m_i}\right)^+$$
$$(P_3)_{i,j}=\left(\frac{c(m_i,\tau_j)-c(m_{i-1},\tau_j)}{m_i-m_{i-1}}-\frac{c(m_{i+1},\tau_j)-c(m_i,\tau_j)}{m_{i+1}-m_i}\right)^+,$$

with the appropriate endpoints being set to zero: $(P_1)_{i,N_\tau}=0$ for $i=1,\ldots N_m$, $(P_2)_{N_m,j}=0$, $(P_3)_{N_m,j}=(P_3)_{0,j}=0$ for $j=1,\ldots,N_\tau$. The arbitrage penalty may then be expressed as the 1-norm of the matrix $P_1+P_2+P_3$:

$$\Phi\left(\boldsymbol{\sigma}(\boldsymbol{m},\boldsymbol{\tau})\right)=\sum_{i=1}^{N_m}\sum_{j=1}^{N_\tau}(P_1+P_2+P_3)_{i,j}=\|P_1+P_2+P_3\|_1.$$

Remark 2 (Extension to swaption implied volatility cube)**.** *When working with swaptions, for every tenor T^a there is an implied volatility surface $\boldsymbol{\sigma}^a(m,\tau)$. Hence, one could calculate the arbitrage penalty for each possible surface (across all of the available tenors) in order to reach an aggregated penalty for the swaption implied volatility cube. That is, suppose that we have available tenors $T^1,...,T^A$. Then we may define the arbitrage penalty for the swaption implied volatility cube $\{\boldsymbol{\sigma}_t^a(\boldsymbol{m},\boldsymbol{\tau})\}_{a=1...A}$ by*

$$\Phi^1(\{\boldsymbol{\sigma}_t^a(\boldsymbol{m},\boldsymbol{\tau})\}_{a=1...A}) = \sum_{a=1}^{A} \Phi(\boldsymbol{\sigma}_t^a(\boldsymbol{m},\boldsymbol{\tau})), \tag{2.13}$$

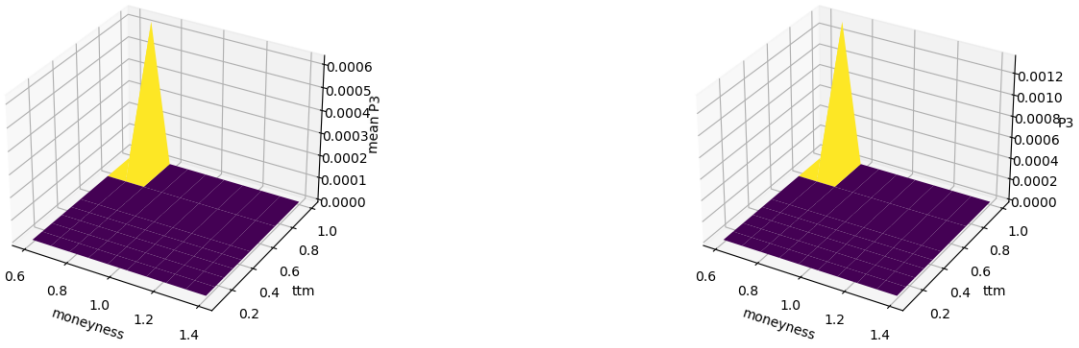
or by

$$\Phi^\infty(\{\boldsymbol{\sigma}_t^a(\boldsymbol{m},\boldsymbol{\tau})\}_{a=1...A}) = \max_{a=1,...,A} \Phi(\boldsymbol{\sigma}_t^a(\boldsymbol{m},\boldsymbol{\tau})). \tag{2.14}$$

2.4.2 Behaviour of arbitrage penalty under perturbations

To gain intuition about the properties of arbitrage penalty (2.9) we investigate its behaviour under perturbations of an arbitrage-free implied volatility surface by i.i.d. noise and parallel shifts. In the numerical results below, the initial implied volatility surface is taken to be the SPX implied volatility surface on 31/12/2021.

Addition of i.i.d. noise We sample $10,000$ implied volatility surfaces by adding i.i.d. noise (i.e. independent across strike and maturity) with a standard deviation of $\epsilon = 0.001$ to the initial arbitrage-free SPX implied volatility surface. We observe that 23% of generated surfaces exhibit butterfly spread arbitrage. The mean butterfly arbitrage penalty matrix P_3 is displayed in Figure 2.8a. We note that violations occur only for far from the money, long-dated options.



(a) Mean effect of adding noise.

(b) Sample parallel shift effect.

Figure 2.8: Butterfly penalty matrices (P3) arising from noise and parallel shifts.

Parallel shifts Rogers and Tehranchi [116] showed that moving implied volatility surfaces by parallel shifts will eventually result in configurations with static arbitrage. We explore this phenomenon quantitatively by adding a parallel shift to an initial arbitrage-free implied volatility surface and testing for static arbitrage. The absolute value of the largest negative shift is taken to be smaller than the smallest implied volatility value, guaranteeing non-negativity. The effect of parallel shifts on the arbitrage penalty are displayed in Figure 2.9: arbitrage constraints are violated for large enough positive shifts, and the arbitrage penalty grows linearly thereafter. For such large shifts, the constraint which is violated is convexity. A sample butterfly arbitrage penalty matrix P_3 is displayed in Figure 2.8b. These results give a quantitative perspective on the results of Rogers and Tehranchi [116].

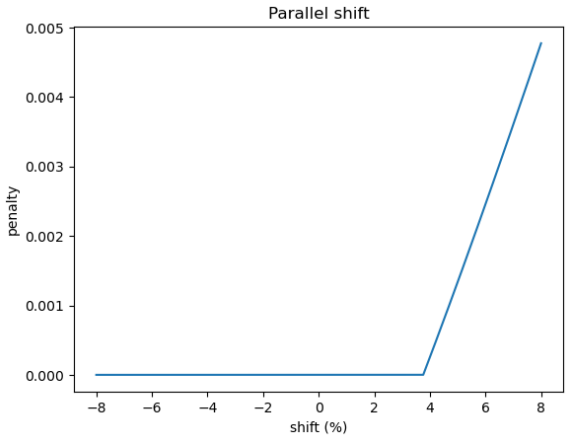


Figure 2.9: Arbitrage violations induced by parallel shifts on SPX implied volatility surface (31/12/2021).

2.4.3 Arbitrage penalty in SPX implied volatility data

Data sources on implied volatility, such as OptionMetrics, are often interpolated from actual market quotes, a procedure which may itself introduce static arbitrage. This has been previously noted by several studies, see e.g. [37]. We study this phenomenon using daily SPX implied volatility surfaces from 2000 to end 2021. We observe non-zero arbitrage penalties, with a decomposition displayed in Figure 2.10 and Table 2.3. 90.5% of dates display no calendar arbitrage, 97.3% display no call spread arbitrage and 84.9% no butterfly arbitrage. Overall 80.2% of the observations correspond to abitrage-free surfaces.

We observe a number of spikes in arbitrage penalties. The two largest spikes happen on 29/09/2008 (during the 2008 financial crisis) and on 13/03/2020 (the start

of the Covid-19 pandemic). Figure 2.10 shows that the majority of arbitrage violations happen during the 2008 financial crisis and during the start of the Covid-19 pandemic. For comparisons, the calendar, call, and butterfly arbitrage penalty matrices P_1, P_2, P_3 on dates 29/09/2008 and 13/03/2020 are displayed in Figures 2.11, 2.12, and 2.13, respectively. This also shows that before using such data as input for model calibration, it needs to be 'cleaned'. Our observations concur with those of Cohen, Reisinger and Wang [34].

Penalty	**Median**	**90th quantile**	**95th quantile**	**99th quantile**
Total Φ	0	0.075	0.13	0.5
Calendar spread p_1	0	0	0.002	0.038
Call spread p_2	0	0	0	0.009
Butterfly spread p_3	0	0.01	0.06	0.458

Table 2.3: Quantiles of arbitrage penalties for SPX options.

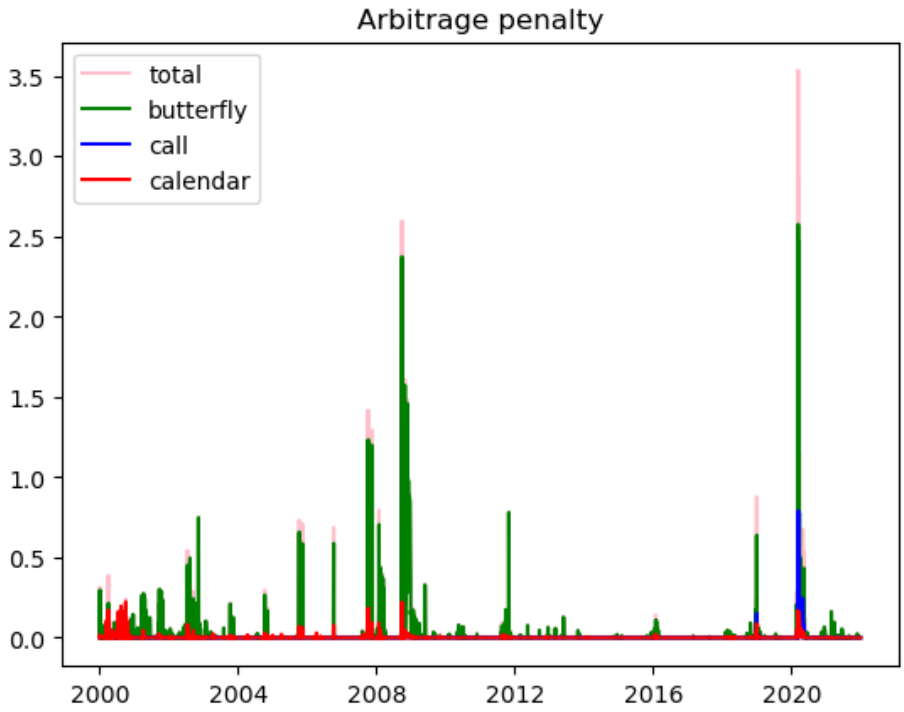
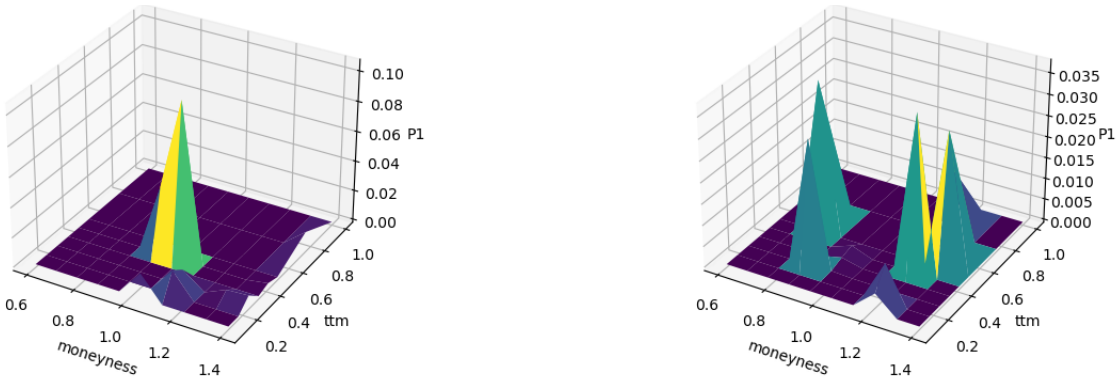
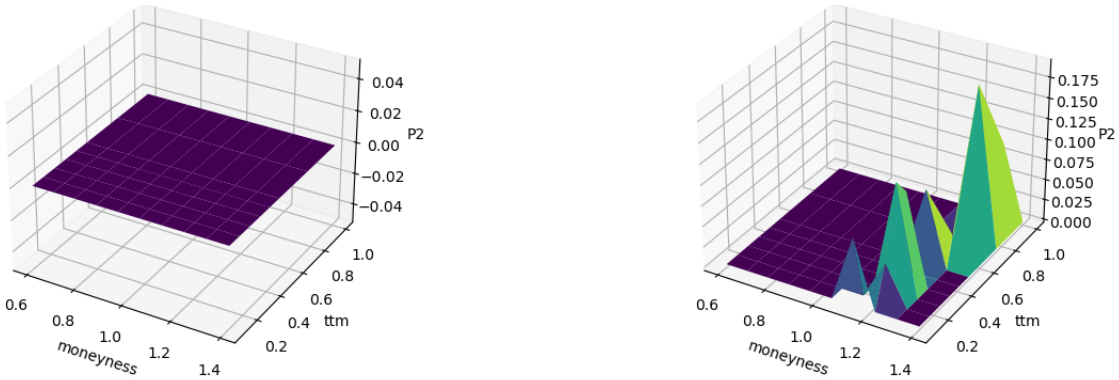


Figure 2.10: Arbitrage penalty decomposition for SPX options.



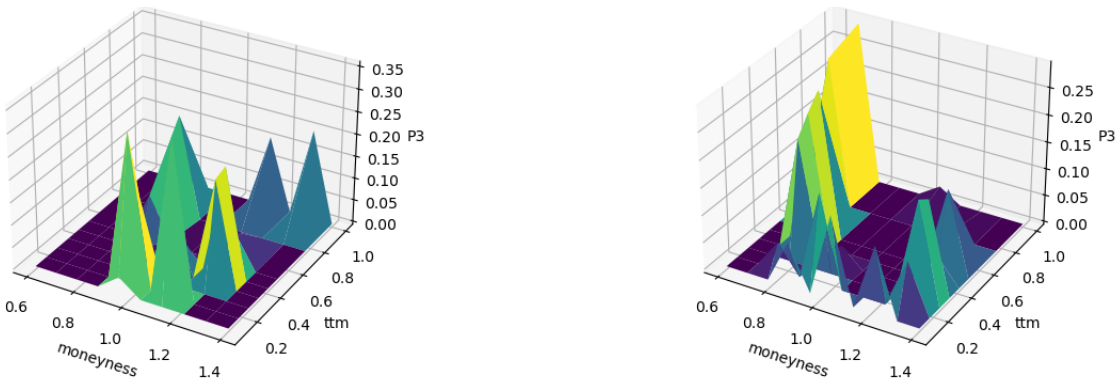
(a) 29.09.2008. (b) 13.03.2020.

Figure 2.11: Calendar spread arbitrage (P1) for SPX options.



(a) 29.09.2008. (b) 13.03.2020.

Figure 2.12: Call spread arbitrage (P2) for SPX options.



(a) 29.09.2008. (b) 13.03.2020.

Figure 2.13: Butterfly spread arbitrage (P3) for SPX options.

2.5 Penalising static arbitrage

2.5.1 Penalisation via scenario reweighting

Our starting point is a baseline model $\mathbb{P}_0$ for implied volatility surface dynamics, which correctly captures the co-movements and statistical properties of implied volatilities and the underlying asset, but may not necessarily be arbitrage-free. For example, this may be a factor model based on PCA, such as [7, 41].

We are interested in generating market scenarios over a time grid $\mathbb{T} = \{0, \ldots, t_{max}\}$. $\mathbb{P}_0$ may be a discrete-time or continuous-time model. For ease of notation, we will continue to denote by $\mathbb{P}_0$ the joint law of the variables $(S_t, \sigma_t(\boldsymbol{m}, \tau), t \in \mathbb{T})$ under $\mathbb{P}_0$.

Our idea is to penalise arbitrage along the paths generated by $\mathbb{P}_0$ by 'tilting' the probabilities associated with such paths. We choose $\beta > 0$ and define a new probability measure $\mathbb{P}_\beta$ on the space of market scenarios by

$$\frac{d\mathbb{P}_\beta}{d\mathbb{P}_0}(\omega) = \frac{\exp\left(-\beta \sum_{t\in\mathbb{T}} \Phi(\sigma_t(\boldsymbol{m}, \boldsymbol{\tau}; \omega))\right)}{Z(\beta)} \tag{2.15}$$

where $Z(\beta)$ is a normalisation factor:

$$Z(\beta) = \mathbb{E}^{\mathbb{P}_0}\left[\exp\left(-\beta \sum_{t\in\mathbb{T}} \Phi\left(\sigma_t(\boldsymbol{m}, \boldsymbol{\tau})\right)\right)\right]. \tag{2.16}$$

If the baseline model $\mathbb{P}_0$ is arbitrage-free then $\Phi(\sigma_t(\boldsymbol{m}, \boldsymbol{\tau})) = 0$ $\mathbb{P}_0$-almost surely so $Z(\beta) = 1$ and $\mathbb{P}_\beta = \mathbb{P}_0$. If, however, $\mathbb{P}_0$ generates surfaces which violate static arbitrage constraints, then the change of measure (2.15) penalises such scenarios, and may be thought of as an importance sampling method which penalises static arbitrage violations. This penalisation increases if we take large β and as $\beta \to \infty$ we reject all scenarios violating static arbitrage constraints. Thus one may think of $1/\beta$ as a 'tolerance' for static arbitrage.

Note that $\mathbb{P}_\beta$ is absolutely continuous with respect to $\mathbb{P}_0$ so we are keeping the same paths but re-weighting them. In the case where the dynamics of variables are given by stochastic differential equations driven by Brownian motion, Girsanov's theorem implies that the re-weighting will impact the drift, but not the quadratic covariation of the variables. However, our approach does not assume that variables are driven by Brownian motion factors, and may be applied in a more general setting. Indeed, the whole procedure also makes sense for a discrete-time time series model.

2.5.2 A 'Weighted Monte Carlo' approach

We now propose a method for sampling from $\mathbb{P}_\beta$, using a Weighted Monte Carlo approach [6]. We proceed as follows:

- Simulate N independent paths $(\omega_i, i = 1, \dots, N)$ from $\mathbb{P}_0$. Each path corresponds to the joint evolution of the underlying asset and the implied volatility surface:

$$\omega_i = (S_t(\omega_i), \boldsymbol{\sigma}_t(\boldsymbol{m}, \boldsymbol{\tau}; \omega_i); t \in \{0, \dots, t_{max}\})$$

- Compute the arbitrage penalty $\varphi(\omega_i)$ along each path:

$$\varphi(\omega_i) = \sum_{t \in \{0,\dots,t_{max}\}} \Phi(\boldsymbol{\sigma}_t(\boldsymbol{m}, \boldsymbol{\tau}; \omega_i)). \tag{2.17}$$

- Associate a weight $w_i(\beta)$ with each path:

$$w_i(\beta) = \frac{\exp\left(-\beta\varphi(\omega_i)\right)}{\sum_{j=1}^{N} \exp\left(-\beta\varphi(\omega_j)\right)}. \tag{2.18}$$

- Sample from the **weighted** model $\mathbb{P}_\beta^N$ defined as

$$\mathbb{P}_\beta^N(\omega_i) = w_i(\beta) \qquad i = 1, \dots, N. \tag{2.19}$$

  That is, instead of sampling each simulated path ω_i with probability $\frac{1}{N}$, we sample it with probability $w_i(\beta)$.

This step-by-step procedure is summarised in Table 2.4. Note that we keep the same paths but modify their weight. Thus, all quantities computed along the path, such as realised volatility and realised covariances will remain the same.

As $\beta \to \infty$, $w_i(\beta) \to 0$ as soon as $\varphi(\omega_i) > 0$ so only paths with arbitrage-free implied volatility surfaces survive for large β. Hence, $\dfrac{1}{\beta}$ can be viewed as an *arbitrage tolerance* parameter.

If the model $\mathbb{P}_0$ is arbitrage-free, then the re-weighting will have no influence as for every $\beta > 0$ we will have $w_i(\beta) = \dfrac{1}{N}$, implying that $\mathbb{P}_\beta^N$ is simply the empirical distribution associated with the N simulated paths. In the general case, the relative entropy of $\mathbb{P}_\beta^N$ with respect to this empirical distribution (i.e. the uniform distribution on $\{\omega_i, i = 1, \dots, N\}$ is an indicator of the 'distance to no-arbitrage':

$$\mathcal{E}_N(\beta) = H(\mathbb{P}_\beta^N | \mathbb{P}_0^N) = -N \log N - N \sum_{i=1}^{N} w_i(\beta) \log w_i(\beta). \tag{2.20}$$

When there is no static arbitrage in the scenarios ω_i generated by $\mathbb{P}_0$, then the relative entropy is zero: $\mathcal{E}_N(\beta) = 0$. On the other hand, the model $\mathbb{P}_0$ is far from being arbitrage-free, the arbitrage penalties $\varphi(\omega_i)$ are large, and the relative entropy $\mathcal{E}_N(\beta)$ will be large.

Our approach is more efficient than rejection sampling, as we sample a fixed number of paths, regardless of the initial model $\mathbb{P}_0$, so the complexity is of order $O(N)$. If the scenarios generated by $\mathbb{P}_0$ are likely to admit static arbitrage, even if the penalty is small and arises from interpolation, rejection sampling may result in an infinite loop.

The following result, which we state for completeness, clarifies the relation between the various probability measures involved. We use the notation of Section 2.5.1.

Proposition 1. *(i) $\mathbb{P}_\beta^N$ weakly converges to $\mathbb{P}_\beta$ as the number of scenarios $N \to \infty$.*

(ii) Let $U = \{\omega \in \Omega,\ \varphi(\omega) = 0\}$ be the set of scenarios free of static arbitrage. If $\mathbb{P}_0(U) > 0$ then, as $\beta \to \infty$, the support of $\mathbb{P}_\beta$ concentrates on U:

$$\forall \varepsilon > 0, \qquad \mathbb{P}_\beta\left(\{\varphi > \varepsilon\}\right) \overset{\beta\to\infty}{\to} 0. \tag{2.21}$$

Proof. (i) is a consequence of the weak law of large numbers. To show (ii), first note that if $\mathbb{P}_0$ is supported on arbitrage-free scenarios, then so is $\mathbb{P}_\beta$. Hence, suppose that $\mathbb{P}_0$ is not supported on the (closed) set $U = \{\omega \in \Omega,\ \varphi(\omega) = 0\}$. The arbitrage penalty

$$\varphi : \omega \in \Omega \mapsto \sum_{t\in\mathbb{T}} \Phi(\sigma_t; \omega)$$

defines a random variable on scenario space and $U = \{\varphi = 0\} \in \mathcal{F}_{t_{\max}}$. Note that since Φ defined by (2.9) is bounded, so is φ. Define

$$A_n = \{\omega \in \Omega,\ \varphi(\omega) > 1/n\} \in \mathcal{F}_{t_{\max}}.$$

Then $A = U^c = \{\varphi > 0\} = \cap_{n\geq 1} A_n$. If $\mathbb{P}_0$ is not supported on U, there exists $n \geq 1$ such that $\mathbb{P}_0(A_n) > 0$.

$$\mathbb{P}_\beta(A_n) = \frac{\int_{A_n} \exp\left(-\beta\varphi(\omega)\right) d\mathbb{P}_0(\omega)}{Z(\beta)}.$$

Since $\varphi = 0$ on $U \subset A_n^c$, we have

$$Z(\beta) = \int_{U^c} \exp\left(-\beta\varphi(\omega)\right) d\mathbb{P}_0(\omega) + \int_U d\mathbb{P}_0(\omega) \geq \mathbb{P}_0(U)$$

Also, since $\varphi > 1/n$ on A_n, we have

$$\int_{A_n} \exp\left(-\beta\varphi(\omega)\right) d\mathbb{P}_0(\omega) \leq \mathbb{P}_0(A_n) \exp(-\beta/n) \to 0 \text{ as } \beta \to \infty.$$

Hence,

$$\mathbb{P}_\beta(A_n) \leq \frac{\mathbb{P}_0(A_n)\exp(-\beta/n)}{\mathbb{P}_0(U)} \to 0 \text{ as } \beta \to \infty.$$

Taking $n > 1/\varepsilon$ yields the result. □

2.6 Factor models for implied volatility dynamics

In order to illustrate our approach, we consider factor models for implied volatility dynamics as the baseline model $\mathbb{P}_0$. We first simulate scenarios from the three-factor model introduced in [41], and then from a four-factor model for the SPX implied volatility surface based on our previous analysis in Section 2.2.

2.6.1 Example: a stylised factor model for implied volatility

We first consider a three-factor model for implied volatility dynamics introduced in [41], based on a Karhunen-Loeve decomposition of co-movements in implied volatilities. The evolution of implied volatility surface paths in this model is given by

$$\sigma_t(m,\tau) = \sigma_0(m,\tau)\exp\left(x_t^1 f_1(m,\tau) + x_t^2 f_2(m,\tau) + x_t^3 f_3(m,\tau)\right) \tag{2.22}$$

where the factors x_t^1, x_t^2, x_t^3 correspond to *level, skew* and *curvature*, as projections on principal components with the analogous representations [41]. Their dynamics are modelled as independent Ornstein-Uhlenbeck processes:

$$dx_t^i = \lambda_i\left(\alpha_i - x_t^i\right)dt + \gamma_i dW_t^i, \tag{2.23}$$

where W_t^i are independent Brownian motions. The basis functions f_1, f_2, f_3 are the first three principal components of the log-implied volatility surface. The price of the underlying asset S is modelled as a diffusion with stochastic volatility $\sigma_t(1,0)$, which corresponds to the short-term at-the-money implied volatility:

$$dS_t = \sigma_t(1,0)S_t dW_t^0, \qquad W_t^0 = \rho W_t^1 + \sqrt{1-\rho^2}\, B_t, \tag{2.24}$$

where $\rho < 0$ and B is a Brownian motion independent from $W^i, i = 1,2,3$. The increments of the first factor x_t^1 are negatively correlated with the returns of the underlying asset: $\text{cov}(W_t^0, W_t^1) = \rho t < 0$. We use $\rho = -0.5$ and $r = 0$ as an example.

Given that this model is based on a principal component analysis of market data, the simulated paths correctly capture the covariance structure of implied volatility

INGREDIENTS

- 'Baseline model' $\mathbb{P}_0$ for implied volatility surface dynamics.
- Time grid $\mathbb{T} = \{0 = t_0 < t_1 < \cdots < t_{max}\}$.
- Moneyness and time to maturity grid $(\boldsymbol{m}, \boldsymbol{\tau}) = (m_i, \tau_j)_{i=1,\ldots,N_m; j=1,\ldots N_\tau}$.
- Number of paths N.
- Arbitrage penalty parameter $\beta > 0$.

Step 1: Simulate N independent scenarios $\omega_i, i = 1, \ldots, N$ from the baseline model $\mathbb{P}_0$. Each scenario ω_i represents a joint evolution of the underlying asset S_t and the implied volatility surface $\boldsymbol{\sigma}_t(\boldsymbol{m}, \boldsymbol{\tau})$ for $t \in \{0, \ldots, t_{max}\}$:

$$\omega_i = (S_t(\omega_i), \boldsymbol{\sigma}_t(\boldsymbol{m}, \boldsymbol{\tau}; \omega_i); t \in \{t_0, \ldots, t_{max}\}) .$$

Step 2: For each simulated path $\boldsymbol{\sigma}(\boldsymbol{m}, \boldsymbol{\tau})^i$, compute the arbitrage penalty

$$\varphi(\omega_i) = \sum_{t \in \mathbb{T}} \Phi(\sigma_t(\boldsymbol{m}, \boldsymbol{\tau}; \omega_i) \ ).$$

Step 3: If $\varphi(\omega_i) = 0$ for all $i = 1 \ldots N \rightarrow$ **STOP**, else.
Step 4: Compute the weights

$$w_i(\beta) = \frac{\exp(-\beta\varphi(\omega_i))}{\sum_{j=1}^{N} \exp(-\beta\varphi(\omega_j))}.$$

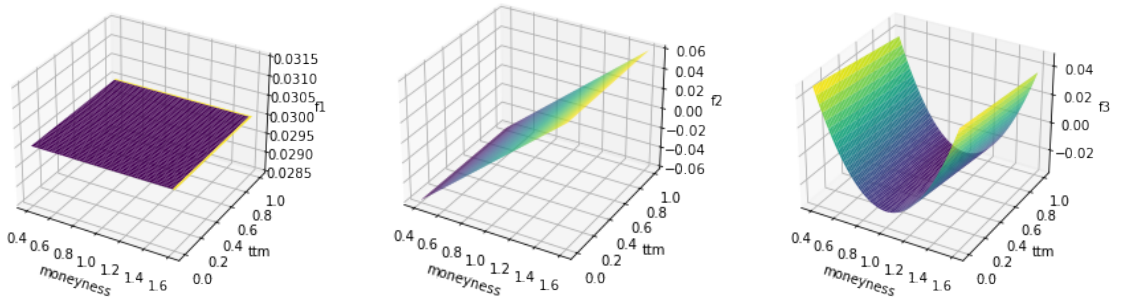
Step 5: Compute the relative entropy $\mathcal{E}_N(\beta) = H(\mathbb{P}_\beta^N | \mathbb{P}_0^N)$.
Step 6: Sample the scenarios $\omega_i, i = 1, \ldots, N$ with probability $w_i(\beta)$:

$$\mathbb{P}_\beta^N(\omega_i) = w_i(\beta).$$

Table 2.4: Weighted Monte Carlo for implied volatility scenarios.

co-movements. Furthermore, the functional form (2.22) of the implied volatility surface guarantees smoothness of the surface and continuity of simulated paths.

In the first example, we suppose $\alpha_i = 0, \gamma_i = 1$ and use the coefficients λ_i given in [41] for the SPX implied volatilities to drive the level, skew and curvature processes x_t^1, x_t^2, x_t^3. The basis functions are based on the first three main components of the market data and are shown in Figure 2.14. The initial surface σ_0 was taken to be the arbitrage-free SPX implied volatility surface on 31/12/2021.



(a) $f_1(m, \tau)$ (b) $f_2(m, \tau)$ (c) $f_3(m, \tau)$

Figure 2.14: Basis functions f_1, f_2, f_3 corresponding to level, skew and curvature used to simulate scenarios from the factor model (2.22)-(2.23)..

As shown by Rogers and Tehranchi [116], an affine factor model such as (2.22)-(2.23) may violate static arbitrage constraints, so we apply the weighting procedure described in Table 2.4. We simulate $N = 100,000$ 3-month scenarios from the factor model (2.22)-(2.23).

Among these scenarios, 64.8% were arbitrage-free. However, even when the arbitrage penalty of a simulated path was non-zero, it was much lower than that of SPX implied vol data. Quantiles of arbitrage penalties across different $\mathbb{P}_\beta$ are displayed in Table 2.5. When comparing the arbitrage penalty quantiles to those of SPX implied volatility displayed in Table 2.3, it is important to note that the factor model arbitrage penalties are calculated for paths, whereas the SPX market data arbitrage penalties are for individual surfaces only. In the scenarios generated by the factor model (2.22)-(2.23), only butterfly arbitrage was observed. All simulated implied volatility surfaces satisfied the absence of calendar and call spreads. The pattern of violations in the butterfly constraint resembles the violations induced by the addition of i.i.d. noise perturbations in Section 2.4.

Table 2.5 displays the quantiles of the arbitrage penalty φ defined by (2.17) under $\mathbb{P}_\beta^N$. To compute the q-th quantile of the arbitrage penalty under $\mathbb{P}_\beta^N$, we sort the

scenarios in increasing order of arbitrage penalties $\varphi(\omega_{(1)}) \leq \varphi(\omega_{(2)}) \leq \cdots \leq \varphi(\omega_{(N)})$. The q-th quantile is then estimated as:

$$F_{\varphi}^{-1}(q) = \varphi(\omega_{(k)}), \quad k = \min\{j \in \{1, \ldots, N\} : \sum_{i=1}^{j} w(\omega_{(i)}) \geq q\}. \tag{2.25}$$

$\boldsymbol{\beta}$	**0**	$\mathbf{10^2}$	$\mathbf{10^3}$	$\mathbf{10^4}$	$\mathbf{10^5}$
90th quantile	0.044	0.001	0	0	0
95th quantile	0.09	0.004	0.0001	0	0
99th quantile	0.206	0.014	0.001	0.0004	0

Table 2.5: Quantiles of arbitrage penalty under $\mathbb{P}_{\beta}^{N}$ in scenarios simulated from the factor model (2.22)-(2.23).

As we increase β, we are less and less likely to sample a path with non-zero arbitrage penalty. Table 2.5 shows that with $\beta = 10^5$, 99% of scenarios are arbitrage-free. In this example, for $\beta = 10^{10}$, we are left with arbitrage-free paths only.

Figure 2.15 shows the relative entropy $\mathcal{E}(\beta) = H(\mathbb{P}_{\beta}^{N} | \mathbb{P}_{0}^{N})$ as a function of β. We observe a sharp transition around $\beta = 100$, suggesting that for $\beta \gg 10^2$ the penalisation eliminates scenarios with arbitrage.

The histogram of weights $w_i(10^2)$ (Figure 2.16) illustrates the clustering of weights into two groups: those corresponding to arbitrage-free scenarios, which are equally weighted, and those corresponding to scenarios with arbitrage penalty $\varphi(\omega_i) > 0$ whose weights are driven very close to zero. We note that even with $\beta = 10^2$, some of the weights are already very close to zero.

We conclude that for the factor model (2.22)-(2.23) the impact of the penalty step is small in terms of entropy distance.

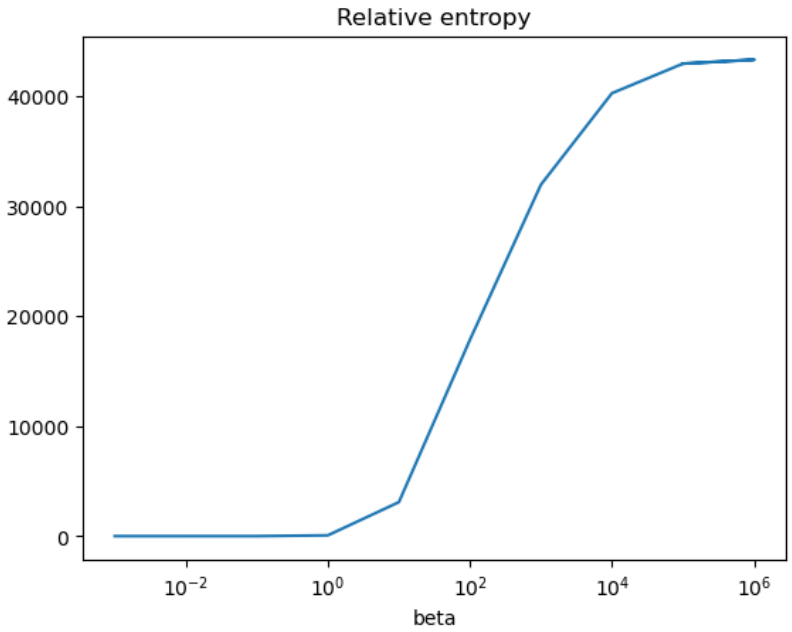


Figure 2.15: Relative entropy $H(\mathbb{P}_\beta^N|\mathbb{P}_0^N)$ in (2.22)-(2.23) as a function of β.

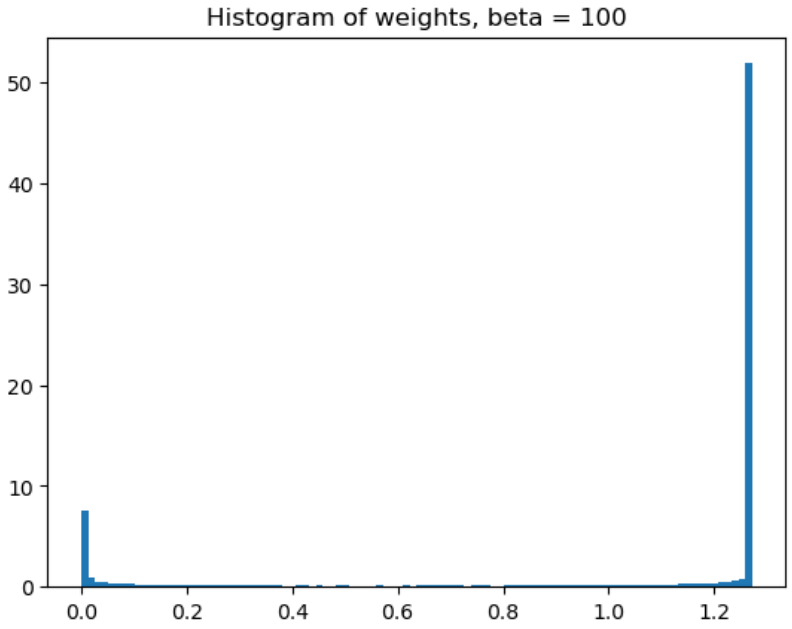


Figure 2.16: Histogram of $Nw(\beta)$ with $\beta = 10^2$ in (2.22)-(2.23).

2.6.2 Example: factor model for the SPX implied volatility surface

We now give an example of a factor model for the SPX implied volatility surface based on the findings from Section 2.2. As in the factor model (2.22)-(2.23), we consider the following dynamics:

$$\sigma_t(m,\tau) = \overline{\sigma}(m,\tau) \exp\left(\sum_{i=1}^{4} x_t^i f_i(m,\tau)\right), \tag{2.26}$$

where the factors $x_t^1, x_t^2, x_t^3, x_t^4$ in this case correspond to *level, skew, term structure* and *curvature*, as projections on principal components with the analogous representations. Their dynamics are once again modelled as independent Ornstein-Uhlenbeck processes:

$$dx_t^i = \lambda_i \left(\alpha_i - x_t^i\right) dt + \gamma_i dW_t^i, \quad i = 1, \ldots, 4. \tag{2.27}$$

The underlying asset S is modelled as a process with stochastic volatility proportional to the one-month ATM volatility $\sigma_t(1, \frac{1}{12})$, as discussed in Section 2.2:

$$dS_t = \nu \ \sigma_t\left(1, \frac{1}{12}\right) \ S_t dW_t^0, \tag{2.28}$$

$$W_t^0 = \rho_1 W_t^1 + \rho_2 W_t^2 + \rho_3 W_t^3 + \rho_4 W_t^4 + \sqrt{1 - (\rho_1^2 + \rho_2^2 + \rho_3^2 + \rho_4^2)} \ B_t, \tag{2.29}$$

where $\nu > 0$, $\rho_1, \rho_2, \rho_4 < 0$, $\rho_3 > 0$ and $B, W^i, i = 1, \ldots, 4$ are independent Brownian motions.

The factor model (2.26)-(2.27) may be adapted to any underlying asset. We demonstrate the approach for SPX options, by using as factors the first four principal components displayed in Figure 2.3 for f_1, f_2, f_3, f_4. We estimate $\alpha_i, \lambda_i, \gamma_i, i = 1, \ldots, 4$ via a Generalised Method of Moments (GMM), using the first two moments and the autocorrelation function at various lags as moment conditions. Estimates are shown in Table 2.6. We set $\nu = \frac{1}{2}$ and the correlations ρ_i, $i = 1, \ldots, 4$ to be the historical correlations from Table 2.2.

	λ	α	γ
X_t^1	2.018	−0.422	4.414
X_t^2	0.986	−0.312	1.993
X_t^3	1.258	0.097	1.295
X_t^4	1.497	−0.021	0.824

Table 2.6: Estimated OU parameters for SPX implied volatility factors.

As in the previous example, we simulate 100,000 3-month paths from the model (2.26)-(2.27), using the above hyperparameters. The initial surface is the average SPX implied volatility $\overline{\sigma}$ (Figure 2.1), and the starting values for the level, skew, term structure, and curvature processes are those observed on 31/12/2021. The percentage of paths and surfaces admitting a non-zero arbitrage penalty is shown in Table 2.7.

	Total	Calendar	Call	Butterfuly
Paths	62.761%	21.245%	36.548%	36.548%
Surfaces	16.055%	4.004%	9.117%	7.073%

Table 2.7: Arbitrage presence in scenarios from the SPX factor model (2.26)-(2.27).

The relative entropy, shown in Figure 2.17, exhibits a sharp transition around $\beta = 10$ indicating that the penalisation efficiently eliminates scenarios with arbitrage. The effect of β on arbitrage penalty is shown in Table 2.8. The arbitrage penalty in simulated scenarios is much lower than the historically observed penalties in the SPX implied volatilities (Table 2.3), considering that the penalties in Table 2.8 correspond to 3-month paths. It becomes negligible with $\beta = 10^2$, and arbitrage is effectively removed for $\beta > 10^4$ in this example. Furthermore, we note that the quantiles of the arbitrage penalty in the SPX model (2.26)-(2.27) (Table 2.8) decay faster with β compared to the corresponding quantiles in the factor model (2.22)-(2.23) (Table 2.5).

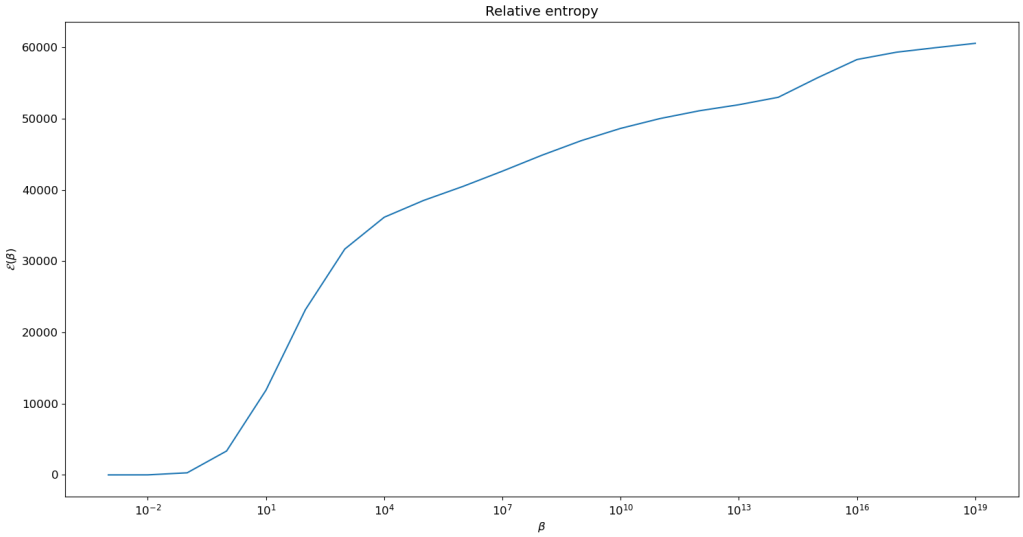


Figure 2.17: Relative entropy $H(\mathbb{P}^N_\beta|\mathbb{P}^N_0)$ as a function of β: SPX factor model (2.26)-(2.27).

β	**0**	$\mathbf{10^2}$	$\mathbf{10^4}$	$\mathbf{10^6}$	$\mathbf{10^{10}}$	$\mathbf{10^{15}}$
90th quantile	0.20	$5 \cdot 10^{-5}$	$5 \cdot 10^{-10}$	$2 \cdot 10^{-13}$	0	0
95th quantile	0.77	0.002	$2 \cdot 10^{-7}$	$3 \cdot 10^{-9}$	$4 \cdot 10^{-15}$	0
99th quantile	4.49	0.01	$3 \cdot 10^{-7}$	$2 \cdot 10^{-9}$	$1 \cdot 10^{-11}$	$4 \cdot 10^{-16}$

Table 2.8: Quantiles of arbitrage penalty under $\mathbb{P}^N_\beta$ for SPX factor model (2.26)-(2.27).

Simulating the volatility index We simulate the VIX dynamics associated with the SPX four-factor model (2.26)-(2.28) using the CBOE methodology [25, 22]. We fix the moneyness grid by taking 100 equispaced values between 0.5 and 1.5. We simulate 10-year VIX and SPX paths using a frequency of one day $\Delta t = \frac{1}{252}$ with the average SPX implied volatility surface, and the SPX price on the 31st Dec 2021 as the starting point. The remaining hyperparameters are as in the previous simulations. Figure 2.18 displays simulated sample paths for the underlying and VIX. We note that the model (2.26)-(2.28) is able to produce high VIX values as historically observed during the 2008 financial crisis and during the Covid-19 pandemic. Figure 2.19 displays simulated paths for one-month realised vol, one-month ATM volatility and VIX. We note that the VIX and the ATM volatility are higher than the one-month forward realised volatility. The ATM volatility is usually below the VIX in the simulations, which is consistent with the post-pandemic dynamics (Figure 2.7).

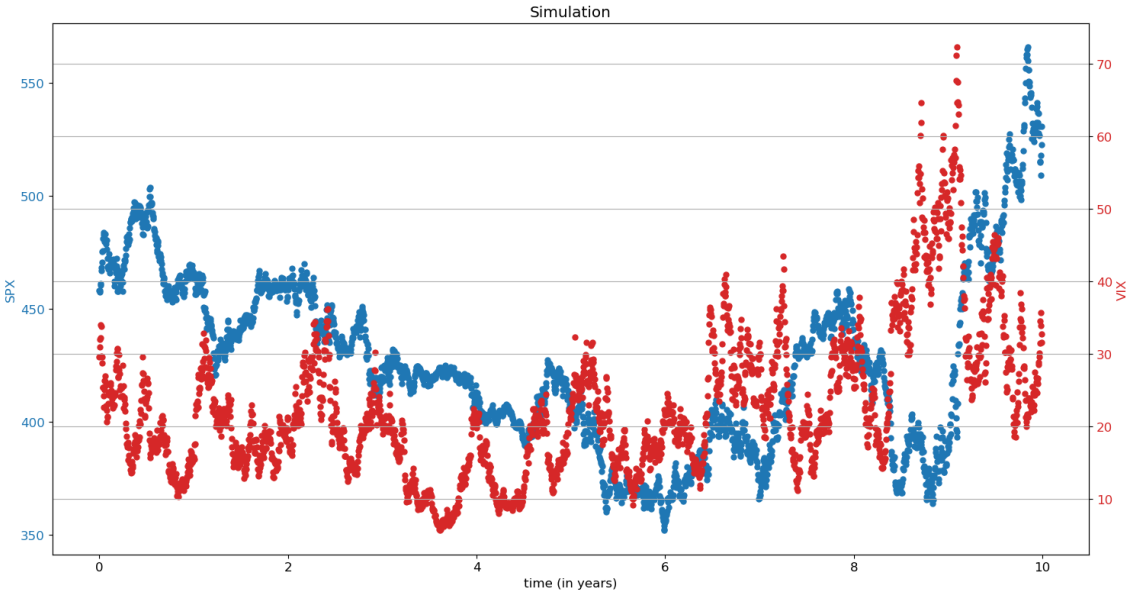


Figure 2.18: Simulation of a 10-year scenario for VIX (red) and SPX (blue) using the SPX factor model (2.27)-(2.28).

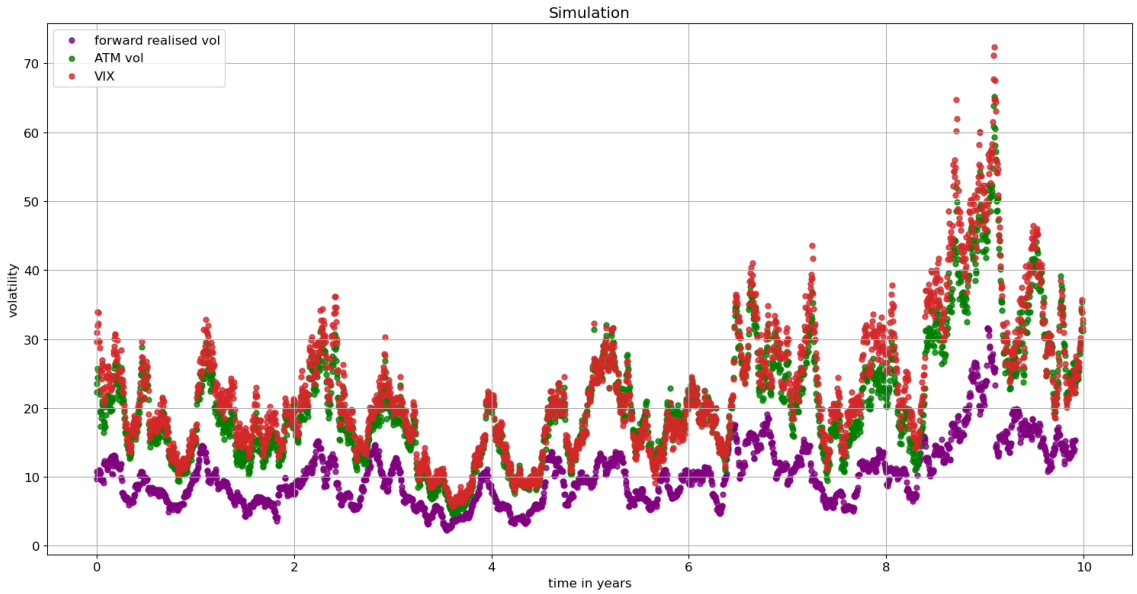


Figure 2.19: Simulation of VIX (red), ATM volatility (green), and the 30-day realised volatility (purple) using the SPX factor model (2.27)-(2.28).

We further investigate the relationship between the simulated values of VIX, ATM vol, SPX, and the level process. Pearson correlation between the simulated log-increments of SPX, ATM vol, VIX, and the increments of the level process is shown in Table 2.9. We note that the log-returns of the underlying are negatively correlated with the increments of the level process, the log-returns of ATM vol, and the log-returns of VIX. There is a high positive correlation between the log increments of the ATM vol, VIX, and the increments of the level process. This is consistent with the historical correlations shown in Figure 2.6.

	$\Delta \log S_t$	ΔX_t^1	$\Delta \log \sigma_t^{ATM}$	$\Delta \log \sigma_t^{VIX}$
$\Delta \log S_t$	1.00	−0.45	−0.31	−0.30
ΔX_t^1	−0.45	1.00	0.96	0.94
$\Delta \log \sigma_t^{ATM}$	−0.31	0.96	1.00	0.99
$\Delta \log \sigma_t^{VIX}$	−0.30	0.94	0.99	1.00

Table 2.9: Pearson correlation between simulated values of log-returns of SPX, returns of the level process, log-returns of the ATM vol, log-returns of VIX using the SPX factor model (2.27)-(2.28).

In Figure 2.20 we compare the historical (2012-2021) and the simulated joint distribution of the log-returns of SPX and of VIX. The means of the two distributions align, and the corresponding marginal distributions of the simulated and historical values are close to each other. However, we note that as the historical correlation between the log-returns of SPX and of VIX is non-constant (Figure 2.6), the joint distribution changes through time as well. The correlation between the log-returns SPX and VIX being lower in the simulations than in the 2012-2021 historical data (Figure 2.20) can be contributed to the correlation ρ_1 used in simulations being lower than the average historical daily correlation of R_t and ΔX_t^1 for the time period 2012-2021 (Figure 2.6). Overall, we conclude that the four-factor model (2.27)-(2.28) is able to generate realistic scenarios for VIX, consistent with the historical observations.

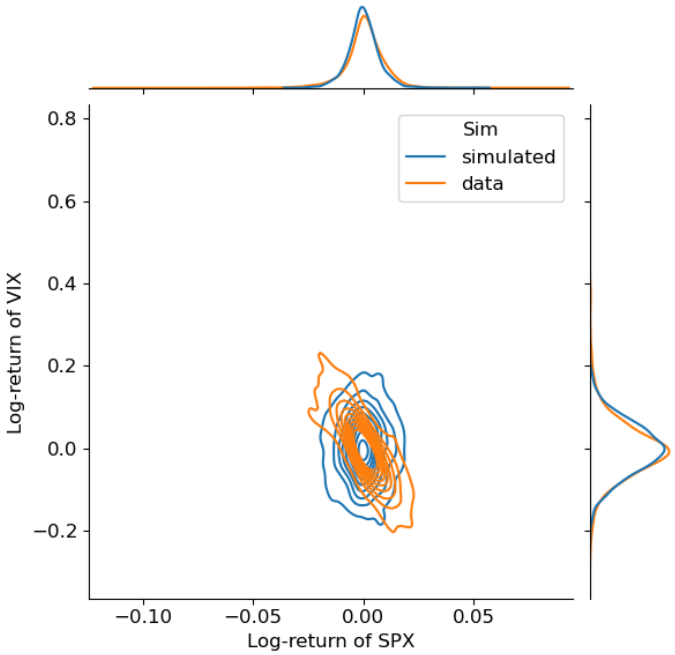


Figure 2.20: Joint distribution of log-returns of VIX and the log-returns of SPX in simulations via the SPX factor model (2.27)-(2.28) and in the historically observed data (2012-2021).

We therefore have a scalable, explainable model for implied volatility surfaces, which correctly captures the co-movements of implied volatilities, is tractable, and when combined with our Weighted Monte Carlo approach, is arbitrage-free.

Chapter 3

VolGAN: a generative model for arbitrage-free implied volatility surfaces

3.1 Introduction

Building on the statistical analysis and modelling framework of Chapter 2, this chapter introduces VolGAN, a fully data-driven generative model designed to simulate arbitrage-free implied volatility surfaces. Chapter 2 identified key stylised facts and structural constraints, such as static arbitrage conditions, that any realistic model must satisfy. Instead of explicitly constructing a parametric model to enforce these properties, as previously done, we now pursue a learning-based approach. We investigate whether a conditional generative model can learn the complex dynamics of implied volatilities directly from market data, while remaining consistent with financial constraints and adaptive to changing market conditions.

One of the biggest hurdles in using generative models for the task of implied volatility modelling has been satisfying the no-arbitrage constraints. While studies such as [37] opt for neural-SDEs and impose strict conditions on the drift coefficients, the arbitrage-scenario re-weighting [45], introduced in Chapter 2, allows for a much richer class of generative models to be used in this setting.

As outputs from a data-driven model may not be all arbitrage-free, it is necessary to quantify potential violations and correct for them. The arbitrage penalty (2.9) allows for precisely this. Furthermore, if the generative model is trained on market mid-prices, there might be periods of time (such as during the Covid-19 pandemic) during which the arbitrage penalty of the actual implied volatility data is non-zero, as shown in Chapter 2. During such periods, it might be very unrealistic, or unfavourable,

to produce arbitrage-free mid-prices, when the market input mid-prices are not. Of course, if the generative model is not appropriate, the arbitrage penalty of its outputs will be very far away from the arbitrage penalty of its inputs.

The arbitrage scenario re-weighting described in Table 2.4 can be viewed as a *safety switch*, which is applied after training. When paired with a generative model capable of reproducing the market dynamics, it represents a powerful tool that allows decoupling the training of the generative model and handling the arbitrage constraints. Furthermore, in some instances, it might be favourable to opt for *raw* outputs of the generative model, rather than opting to *reduce* the levels of static arbitrage. Hence, the methodology from Chapter 2 allows a generative model to both replicate the market as it is, and to separately ensure the no-arbitrage constraints. If a model were to be trained to strictly satisfy the no-arbitrage constraints, the simulated values would be very far away from the real data during periods of market turbulence. Our approach allows a seamless switch between different regimes and goals, whether it is more accurate forecasting, or ensuring that the no-arbitrage constraints are satisfied.

This chapter, based on the article [134], introduces VolGAN, which is a custom generative model based on conditional GANs [60, 103], capable of simulating next-day scenarios for the implied volatility surface and the underlying, given today's implied volatility surface, the last two returns of the underlying, and realised volatility. We train it on the historical time series of options on S&P500 index, and carefully study the statistical properties of the learned dynamics. We show that VolGAN learns the co-movements of implied volatility and the underlying, adapts to the changing market regimes, and produces correct relationships with the CBOE volatility index (VIX) [25].

Outline. Section 3.2 introduces VolGAN. Section 3.3 demonstrates its ability to learn implied volatility dynamics.

Code availability. The code used to implement VolGAN is publicly available at `https://github.com/milenavuletic/VolGAN`.

3.2 A generative model for implied volatility surfaces

VolGAN is a customised conditional generative adversarial network with a smoothness penalty incorporated into the generator's loss function, combined with scenario re-

weighting applied to the output scenarios [45].

VolGAN receives as input

- the implied volatility surface at the previous date,
- the two previous underlying returns,
- the realised volatility from the previous period,

and outputs (joint) scenarios for

- the return of the underlying asset and
- the implied volatility surface

for the next period, along with a set of weights (probabilities) associated with these scenarios. We now discuss the methodology in more detail.

3.2.1 Architecture

Suppose that we have observations at times $t \in \mathbb{T}$, in increments of $\Delta t = 1/252$ (1 day), with S_t the price of the underlying, and $\sigma_t(\boldsymbol{m}, \boldsymbol{\tau})$ the implied volatility surface on the grid $(\boldsymbol{m}, \boldsymbol{\tau})$ at time t. Denote by $g_t(\boldsymbol{m}, \boldsymbol{\tau})$ the log-implied volatility surface at time t:

$$g_t(\boldsymbol{m}, \boldsymbol{\tau}) = \log \sigma_t(\boldsymbol{m}, \boldsymbol{\tau}), \quad \Delta g_t(\boldsymbol{m}, \boldsymbol{\tau}) = g_{t+\Delta t}(\boldsymbol{m}, \boldsymbol{\tau}) - g_t(\boldsymbol{m}, \boldsymbol{\tau}). \tag{3.1}$$

Let R_t be the log-return of the underlying:

$$R_t = \log\left(\frac{S_{t+\Delta t}}{S_t}\right), \tag{3.2}$$

and denote by γ_t the one-month realised volatility:

$$\gamma_t = \sqrt{\frac{252}{21}\sum_{i=0}^{20} R^2_{t-i\Delta t}}. \tag{3.3}$$

We aggregate $R_{t-\Delta t}, R_{t-2\Delta t}, \gamma_{t-\Delta t}, g_t(\boldsymbol{m}, \boldsymbol{\tau})$ into a *condition/input* vector a_t:

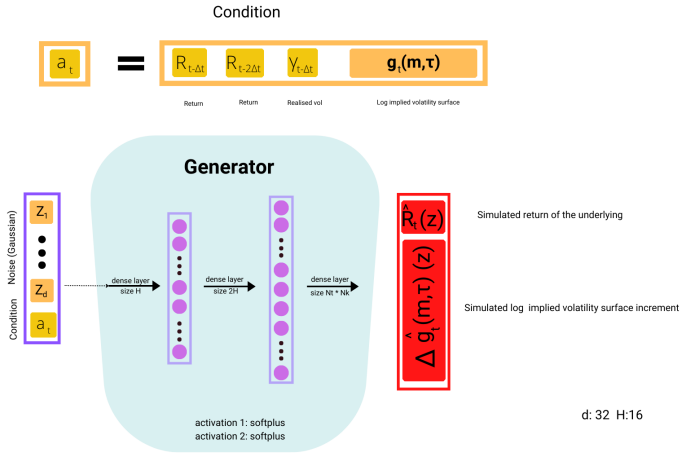
$$a_t = (R_{t-\Delta t}, R_{t-2\Delta t}, \gamma_{t-\Delta t}, g_t(\boldsymbol{m}, \boldsymbol{\tau})). \tag{3.4}$$

Following the notation introduced in Chapter 1, the generator G takes as input this condition a_t and i.i.d. noise $z_t \sim \mathcal{N}(0, I_d)$ and outputs simulated values $\hat{R}_t(z), \Delta\hat{g}_t(\boldsymbol{m}, \boldsymbol{\tau})$ for the return and implied volatility (log-)increments:

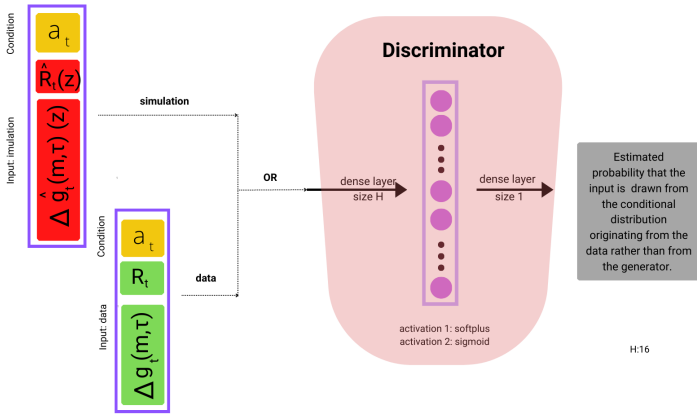
$$G(a_t, z_t) = (\hat{R}_t(z_t), \Delta\hat{g}_t(\boldsymbol{m}, \boldsymbol{\tau})(z_t)), \tag{3.5}$$

We denote by $G(a_t, z)|_{2:} = \Delta\hat{g}_t(\boldsymbol{m}, \boldsymbol{\tau})(z)$ the second component of the generator's output, which corresponds to the simulated log implied volatility increment.

The discriminator operates in a classical conditional GAN setting and is trained via the binary cross-entropy loss (1.2). Both the generator and the discriminator are fully-connected feed-forward neural networks. Their architectures are shown in Figures 3.1a and Figure 3.1b, respectively.



(a) Generator.



(b) Discriminator.

Figure 3.1: VolGAN network architectures.

3.2.2 Training objective

The core component of VolGAN is a customised loss function for the generator, catering to the desired properties of the output volatility surface. A classical GAN trained using binary cross-entropy (BCE) loss (1.2)-(1.3) would result in irregular

surfaces. In order to generate smooth surfaces, we use a smoothness penalty [10, 76] defined as a discrete Sobolev semi-norm in m and τ on the grid $(\boldsymbol{m}, \boldsymbol{\tau})$:

$$L_m(\boldsymbol{g}) = \sum_{i,j} \frac{\left(\boldsymbol{g}(m_{i+1}, \tau_j) - \boldsymbol{g}(m_i, \tau_j)\right)^2}{\left|m_{i+1} - m_i\right|^2} \simeq \|\partial_m g\|^2_{L^2}, \tag{3.6}$$

$$L_\tau(\boldsymbol{g}) = \sum_{i,j} \frac{\left(\boldsymbol{g}(m_i, \tau_{j+1}) - \boldsymbol{g}(m_i, \tau_j)\right)^2}{\left|\tau_{j+1} - \tau_j\right|^2} \simeq \|\partial_\tau g\|^2_{L^2}. \tag{3.7}$$

These terms are included in the training objective $J^{(G)}(\theta_d, \theta_g)$ for the generator:

$$\begin{aligned} J^{(G)}(\theta_d, \theta_g) = -\frac{1}{2}\mathbb{E}\big[\log\left(D\left(a_t, G(a_t, z_t; \theta_g); \theta_d\right)\right)\big] \\ + \alpha_m \mathbb{E}\big[L_m\left(g_t(\boldsymbol{m}, \boldsymbol{\tau}) + G(a_t, z_t; \theta_g)|_{2:}\right)\big] \\ + \alpha_\tau \mathbb{E}\big[L_\tau\left(g_t(\boldsymbol{m}, \boldsymbol{\tau}) + G(a_t, z_t; \theta_g)|_{2:}\right)\big], \end{aligned} \tag{3.8}$$

where $a_t = (R_{t-\Delta t}, R_{t-2\Delta t}, \gamma_{t-\Delta t}, g_t(\boldsymbol{m}, \boldsymbol{\tau}))$, as defined in (3.4). The first term is a binary cross-entropy for the output of the discriminator, and $\alpha_m > 0$ and $\alpha_\tau > 0$ are regularisation parameters. The expectation is computed over the law of the i.i.d. (Gaussian) input $\boldsymbol{z}_t \sim N(0, I_d)$. The smoothness penalties L_m and L_τ are applied to the simulated log-implied volatility surfaces:

$$g_t(\boldsymbol{m}, \boldsymbol{\tau}) + G(a_t, z_t; \theta_g)|_{2:} = g_t(\boldsymbol{m}, \boldsymbol{\tau}) + \Delta\hat{g}_t(\boldsymbol{m}, \boldsymbol{\tau})(z_t) = \hat{g}_t(\boldsymbol{m}, \boldsymbol{\tau})(z_t).$$

It is possible to incorporate the arbitrage penalty (2.9) into the loss function of the generator (3.8). However, we have not done so, and our numerical experiments indicate no notable difference when including it, suggesting that the smoothness penalty is enforcing shape constraints indirectly.

3.2.3 Scenario re-weighting

The outputs of the generator described above are not guaranteed to satisfy the static arbitrage constraints described in Chapter 2. To correct for this, we apply the arbitrage scenario re-weighting methodology [45] to the one-day-ahead scenarios generated by the GAN.

As outlined in Chapter 2, VolGAN samples from the target distribution (2.15) using a Weighted Monte Carlo approach. Given N samples from the generator $(\hat{R}^i, \hat{\boldsymbol{\sigma}}^i)$, $i = 1, \ldots, N$, we compute the arbitrage penalty $\Phi(\hat{\boldsymbol{\sigma}}^i)$ corresponding to each output scenario $(\hat{R}^i, \hat{\boldsymbol{\sigma}}^i)$ using (2.9) and sample the scenario $(\hat{R}^i, \hat{\boldsymbol{\sigma}}^i)$ with probability

$$w^i = \frac{\exp(-\beta\Phi(\hat{\boldsymbol{\sigma}}^i))}{\sum_{j=1}^N \exp(-\beta\Phi(\hat{\boldsymbol{\sigma}}^j))}. \tag{3.9}$$

These weighted scenarios may then be used to compute expectations and quantiles of various quantities of interest under $\mathbb{P}_\beta$. Let X be a function of the state variables, and let x_i be its value in scenario i. Denote by $F_{X,\beta}$ the law of X under $\mathbb{P}_\beta$ and by $\mathbb{E}_\beta[X]$ its expectation. We can estimate $\mathbb{E}_\beta[X]$ by

$$\widehat{\mathbb{E}_\beta[X]} = \sum_{i=1}^{N} w_i x_i, \tag{3.10}$$

while the quantiles of X are estimated as

$$\widehat{F^{-1}_{X,\beta}(q)} = x_{(k)}, \qquad \text{where} \quad k = \min\{j \in \{1,\dots,N\} : \sum_{i=1}^{j} w_{(i)} \geq q\}, \tag{3.11}$$

where $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$ are the order statistics of $x_1, \dots, x_N$.

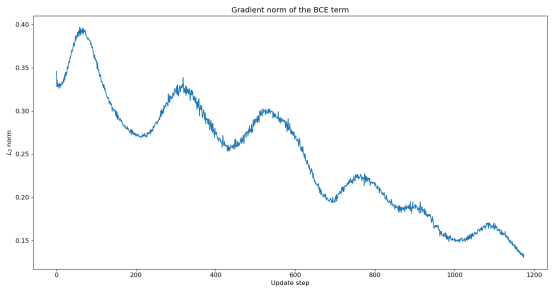
3.2.4 Numerical implementation

The generator G is a three-layer feedforward dense neural network, with the first two activations softplus, and the final layer an affine layer. The random input is (standard) i.i.d. Gaussian noise with dimension $d = 32$. The first layer consists of $H = 16$ neurones, whereas the second layer contains $2H = 32$ neurones. Similarly, the discriminator D is a two-layer feedforward neural network, with softplus and sigmoidal activation functions and layer sizes of $H = 16$ and 1, respectively. The discriminator has a simpler architecture than the generator, as it is of the utmost importance to keep the two neural networks in balance. The architecture of the discriminator is shown in Figure 3.1b, and the architecture of the generator is displayed in Figure 3.1a.

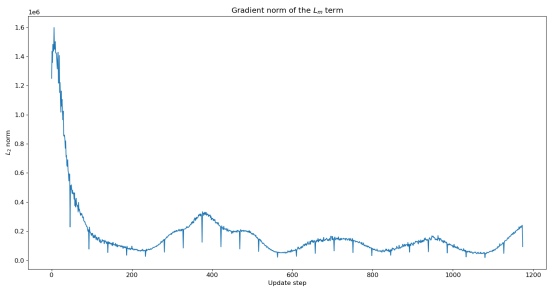
The hyperparameters $\alpha_m, \alpha_\tau > 0$ are chosen by *gradient norm matching*. We first train VolGAN for $n_{grad} = 25$ epochs by performing optimisation via the binary cross-entropy loss only (classical GAN setting). At each update, we calculate the gradient norms of each of the three loss function terms in (3.8): BCE, L_m, $L_\tau$with respect to θ_g. We then set α_m and α_τ, to be the means of observed ratios of the gradient norms of the BCE term to the gradient norms of the L_m and L_τ, respectively. The gradient norms of the BCE, L_m, L_τ terms with respect to θ_g during this stage are shown in Figure 3.2. We note that all three gradients behave similarly, that they stabilise over time, and that there is no gradient explosion or vanishing gradient phenomena.

We then restart training VolGAN (from the same initialisation used for the start of the gradient norm matching procedure) with the loss function defined by Equation (3.8) for $n_{epochs} = 10000$ epochs, using an alternating direction method, that is, one

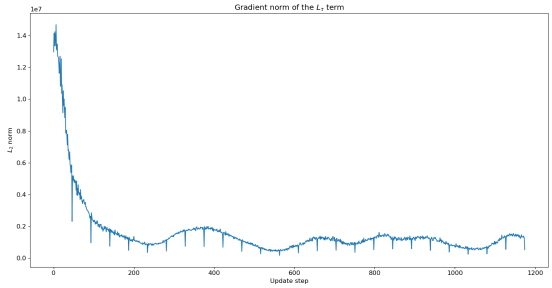
discriminator update for each generator update. The optimiser used is RMSProp [68], and the learning rates of both networks are set to 0.0001. We take $N = 10000$ raw samples from the generator. The mini-batch size is $n_{batch} = 100$.



(a) BCE term.



(b) L_m term.



(c) L_τ term.

Figure 3.2: Norm of gradient of the BCE term, L_m term, and L_τ term with respect to θ_g during the first stage of VolGAN training. n

Calibration of β The hyperparameter β might be chosen by considering the Kullblack-Leibler divergence between the distribution of the weights and the uniform distribution on the scenarios [45]. Based on the results in [45], we set

$$\beta(t) = \frac{500}{\max\{\Phi_i(t)\}}, \tag{3.12}$$

where $\Phi_i(t)$ are the arbitrage penalties associated with the generator outputs on day t.

3.3 Learning to simulate SPX implied volatility surfaces

To demonstrate VolGAN's ability to generate realistic scenarios for SPX implied volatility dynamics, we train VolGAN on the daily time series of market data and examine the properties of the generator thus trained. The same approach might be applied to other equity options.

3.3.1 Data

Unlike the study in Chapter 2, in this chapter, we are also interested in shorter times to maturity. Hence, the data is obtained from the Option Prices file from OptionMetrics. The time period in question is from the 3rd January 2000 to the 28th February 2023, with 3rd Jan 2000-16th Jun 2018 corresponding to the training, and 17th Jun 2019-28th Feb 2023 to the test set. Historical VIX closing prices are available on the CBOE website. The implied risk-free interest rate for each day is calculated as the median rate implied by the put-call parity from the option mid-prices. We construct smooth implied volatility surfaces using the kernel smoothing methodology of [41, 108]. Our grid $(\boldsymbol{m}, \boldsymbol{\tau})$ consists of $m \in \{0.6, 0.7, 0.8, 0.9, 0.95, 1, 1.05, 1.1, 1.2, 1.3, 1.4\}$ and of times to maturity $\tau \in \{\frac{1}{252}, \frac{1}{52}, \frac{2}{52}, \frac{1}{12}, \frac{1}{6}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$, one day to one year. Suppose that on a fixed day we have available implied volatility data $\sigma(m, \tau)$ for $m \in \mathcal{M}$ and $\tau \in \mathcal{T}$, with corresponding values of vega $\kappa(m, \tau)$. We consider a vega-weighted Nadaraya-Watson kernel smoothing estimator with a 2D Gaussian kernel:

$$\hat{\sigma}(m', \tau') = \frac{\sum_{m \in \mathcal{M}, \tau \in \mathcal{T}} \kappa(m, \tau) k(m - m', \tau - \tau') \sigma(m, \tau)}{\sum_{m \in \mathcal{M}, \tau \in \mathcal{T}} \kappa(m, \tau) k(m - m', \tau - \tau'),} \tag{3.13}$$

where:

$$k(x, y) = \frac{1}{2\pi} \exp\left[-\frac{x^2}{2h_1} - \frac{y^2}{2h_2}\right].$$

In order to determine the values of the bandwidth hyperparameters h_1 and h_2, we sample a day uniformly at random from the first 100 days available (which was 31st Jan 2000) and find the pair of hyperparameters (h_1, h_2) minimising the arbitrage penalty. We conduct the search over values between 0.002 and 0.1 (inclusive) in 0.002 increments, for both h_1 and h_2. The minimiser of the arbitrage penalty was the pair $(h_1, h_2) = (0.002, 0.046)$. The resulting arbitrage penalty over the entire data set after smoothing is shown in Figure 3.3. Note that compared to [45] we include shorter times to maturity and use a different data set.

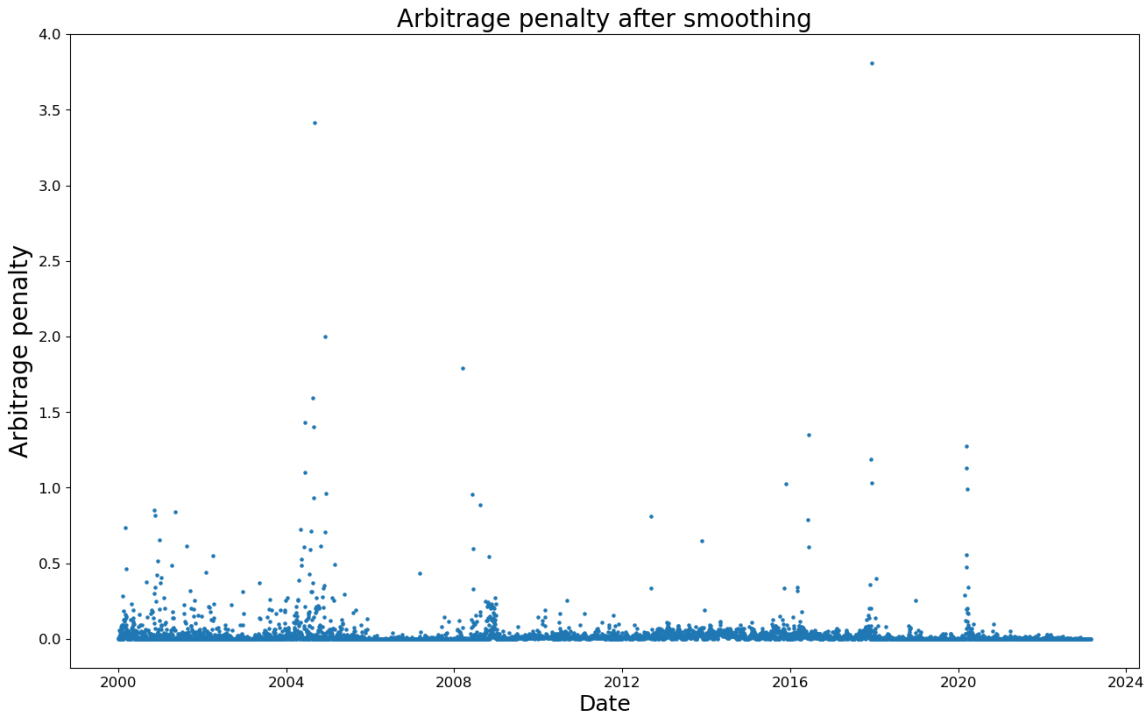


Figure 3.3: Arbitrage penalty for SPX implied volatility surface after smoothing.

To simplify the notation, we will use $\sigma_t(\boldsymbol{m}, \boldsymbol{\tau})$ for the implied volatility surface obtained after smoothing, on the $(\boldsymbol{m}, \boldsymbol{\tau})$ grid. As in Chapter 2, for a general $\sigma_t(m, \tau)$, we first interpolate $\sigma_t(\boldsymbol{m}, \boldsymbol{\tau})$ linearly in moneyness, and then in time to maturity. When extrapolation is necessary, it is linear.

3.3.2 Out-of-sample performance

As discussed in Chapter 2, the main goal of an implied volatility model is to correctly capture the co-movements of implied volatilities, while satisfying static arbitrage constraints. We can measure the latter by considering the 'distance to arbitrage' using the arbitrage penalty (2.9). In order to measure how well VolGAN learns the dynamics and captures the co-movements of implied volatilities, we perform PCA on the generated increments, and compare them with the principal components of the data increments. Furthermore, we simulate the CBOE volatility index VIX [25, 22], which is a non-linear combination of tradable calls and puts. We compare the dynamics of the simulated and SPX implied volatility market data.

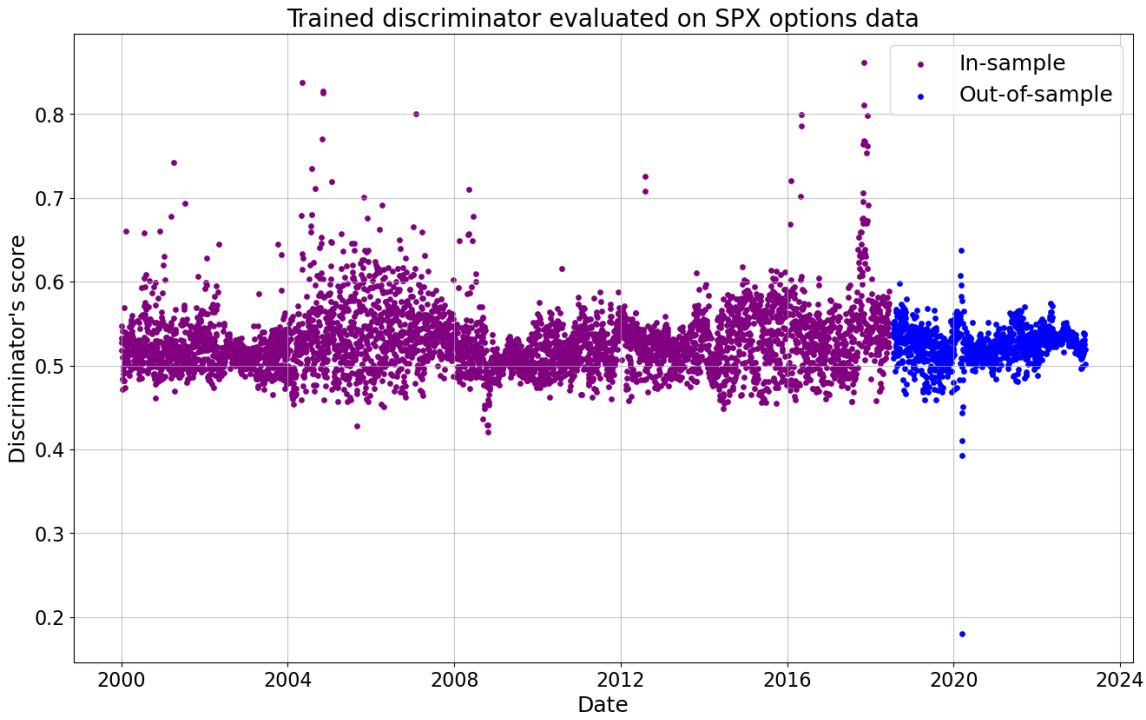


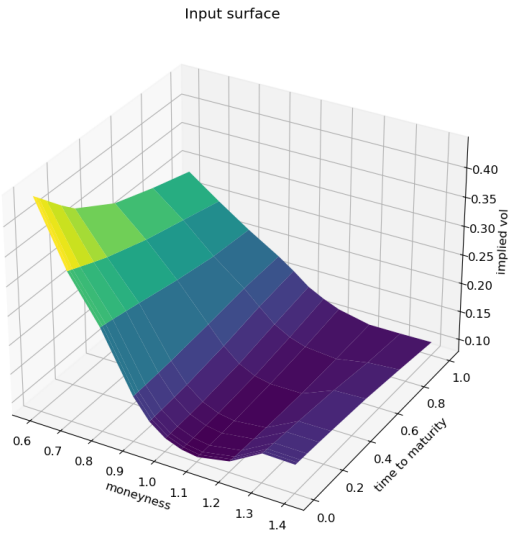
Figure 3.4: Discriminator output score on in-sample and out-of-sample SPX options data.

3.3.2.1 Detecting extreme market events

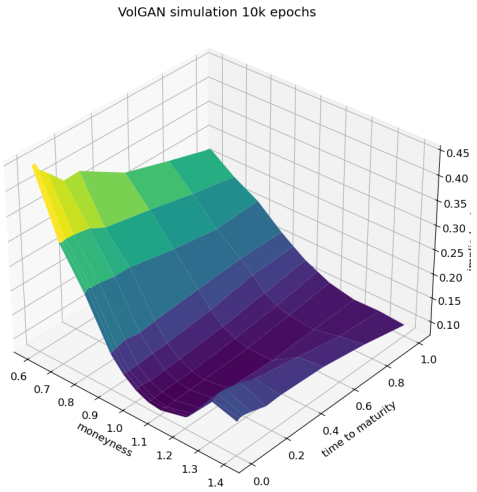
Firstly, we note that the trained discriminator might be used for detecting extreme market events. Figure 3.4 contains discriminator scores on the training and testing data. Since the discriminator has already been trained, it is of no surprise that the outputs cluster around 0.5. There are two clusters of points with scores lower than others: those corresponding to the 2008 financial crisis (in-sample) and to the start of the Covid-19 pandemic (out-of-sample). In particular, the discriminator assigns a score below 0.2 to the data from the start of the Covid-19 pandemic, highlighting the difference in this data compared to the rest of the training and test set.

3.3.2.2 Smoothness and arbitrage constraints

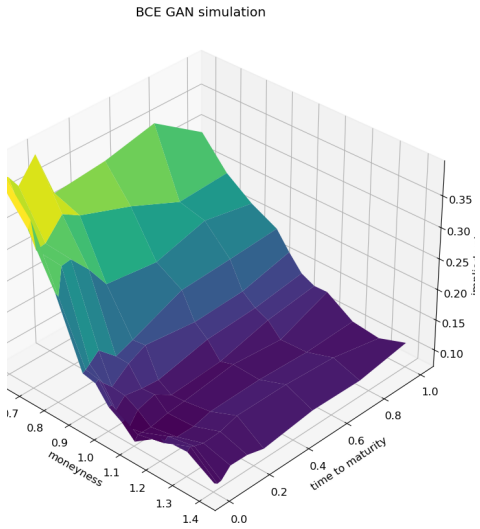
Incorporating the smoothness penalty (3.6)-(3.7) into the loss function (3.8) is crucial for generating smooth surfaces. As shown in Figure 3.5, training via the binary cross-entropy loss (1.2)-(1.3), using the same architecture, hyperparameters, and the same number of training epochs, results in irregular surfaces.



(a) SPX implied volatility surface.



(b) Sample VolGAN output.



(c) Sample BCE GAN output.

Figure 3.5: Implied volatility surfaces generated using (b) VolGAN (c) classical GAN, compared with (a) SPX implied volatility surface.

As the input surfaces might admit static arbitrage, it is not realistic to expect outputs to be completely arbitrage-free. What is plausible, however, is for the outputs to have arbitrage penalties of the same order (or lower) than the inputs. Table 3.1 compares out-of-sample arbitrage penalties for SPX implied volatilities and the outputs of the BCE GAN and VolGAN with/without scenario re-weighting. Arbitrage penalties in the BCE GAN samples are observed to be high: this is linked to the previous observation that BCE GAN fails to generate smooth surfaces, resulting in failure of static arbitrage conditions, which are linked to derivatives of the surfaces.

In contrast, VolGAN outputs have arbitrage penalty levels similar to the input data. Scenario re-weighting leads to a low probability of selecting scenarios with static arbitrage, as shown in Figure 3.6, where the reduction in arbitrage is visualised. The mean, standard deviation, and median values from Table 3.1 correspond to the statistics of the time series displayed in Figure 3.6. We note that during 2022 there is more volatility in arbitrage penalty in VolGAN compared to the remainder of the test period.

	Mean	**Std**	**Median**
Market data	0.0096	0.0628	0.0005
BCE GAN	2.4635	0.9086	2.3164
Raw VolGAN (before weighting)	0.0199	0.088	0.003
VolGAN (after re-weighting)	0.0127	0.0620	0.0014

Table 3.1: Arbitrage penalties in SPX implied volatility market data (test set) vs generated data via GANs trained using (i) BCE loss only (ii) VolGAN loss (iii) VolGAN re-weighted scenarios (adaptive β). Standard deviation and median for GAN outputs correspond to the standard deviation and the median of (re-weighted) average outputs given 10000 samples.

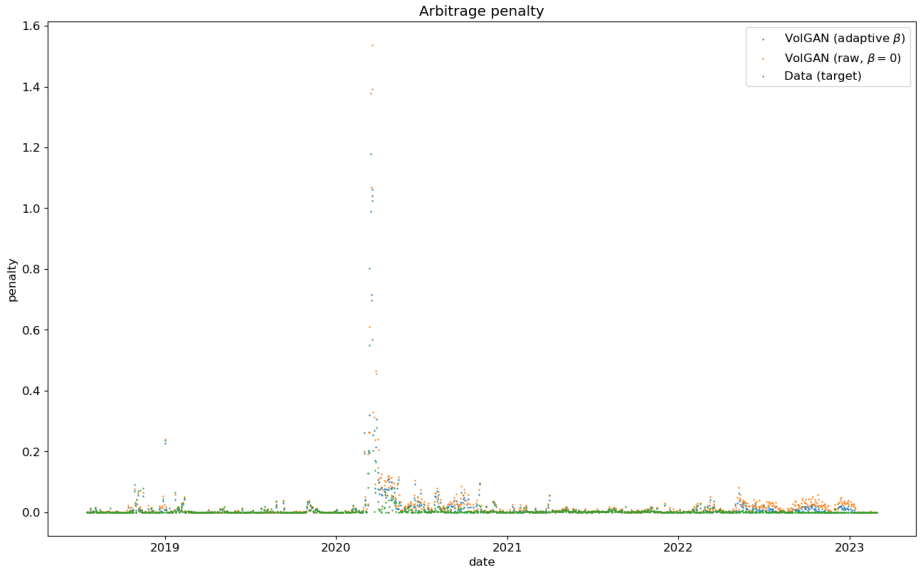


Figure 3.6: Distance to arbitrage as measured by the arbitrage penalty (2.9) in SPX implied volatility data (red) vs. mean arbitrage penalty of surfaces generated via VolGAN, before (blue) and after (green) scenario re-weighting.

3.3.2.3 Next-day forecasting

We use VolGAN to generate next-day forecasts using the conditional expectation of the variable given the history, together with a 95% confidence interval obtained by considering the 2.5% and 97.5% quantiles for the following quantities of interest:

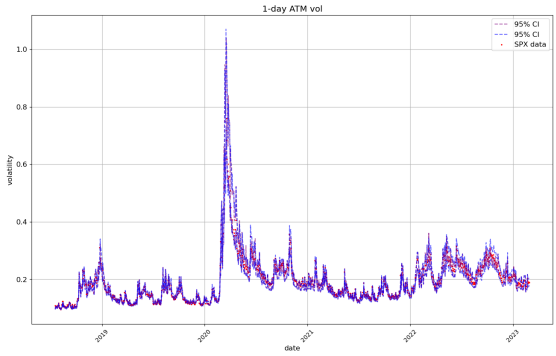
- index level S_t;
- VIX level σ_t^{VIX};
- a range of implied volatilities $\sigma_t(m, \tau)$ with

$$\tau \in \{\frac{1}{252}, \frac{1}{52}, 0.25, 0.125\}, \quad m \in \{0.75, 1, 1.25\}$$

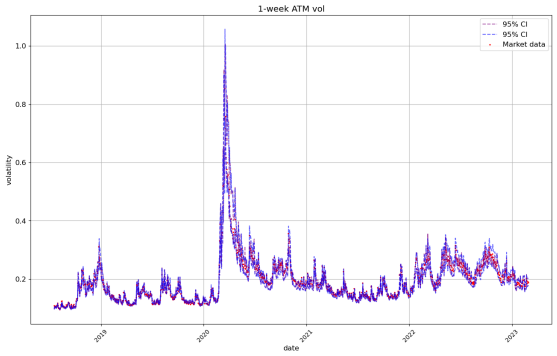
Figures 3.7d, 3.7c, 3.7b, 3.7a compare respectively the 3-month, 1-month, 1-week, and 1-day ATM implied volatility with the VolGAN one-day ahead 95% confidence interval forecast, displaying good agreement with observations. VolGAN appears to slightly overestimate implied volatility levels for $m > 1$ but not for $m < 1$, as shown in Figures 3.7f and 3.7e.

Figure 3.8 displays the simulated and real SPX returns, showing that VolGAN confidence intervals appropriately capture the underlying. We visualise the impact of scenario re-weighting on the confidence intervals in Figure 3.9. During periods of high arbitrage penalty, a small number of simulations hold most of the weight, therefore inducing very narrow confidence intervals. This behaviour is visible not just in the simulations for the underlying, but for the ATM ($m = 1$), OTM ($m = 0.75$), and ITM ($m = 1.25$) implied volatilities (Figures 3.7d, 3.7f, 3.7e respectively). From Figure 3.9, we note that if arbitrage is not penalised ($\beta = 0$), the forecasts are more accurate, including for March and April 2020. However, choosing to use the raw generator might result in static arbitrage of the mid-prices. As before, we note that the width of the confidence intervals varies with time, with the confidence intervals appearing more consistent in 2022. The raw generator ($\beta = 0$) produces stable confidence intervals for all state variables, highlighting VolGAN's stability and not requiring frequent re-calibration.

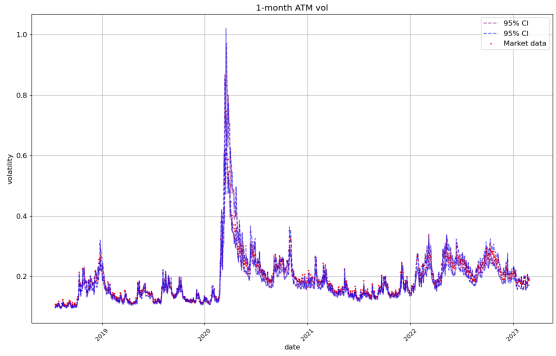
Figure 3.10 compares one-day ahead simulated values of VIX, computed from its definition in terms of simulated call/put prices [22], with the VIX closing prices on target days in the test set. VolGAN simulations are on the same scale as VIX. Some of the differences might be coming from the discrete approximation of the log-contract used for computation of simulated VIX values [25].



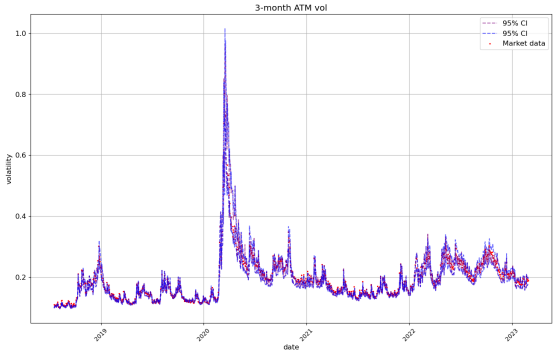
(a) 1-day ATM ($m = 1, \tau = 1/252$)



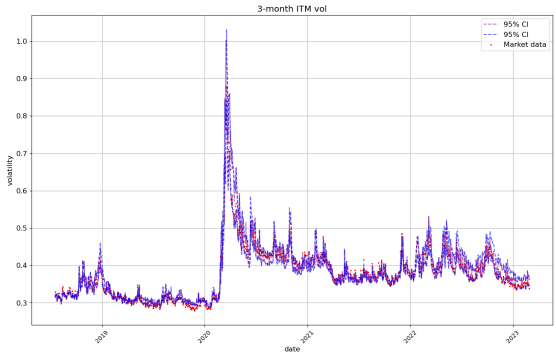
(b) 1-week ATM ($m = 1, \tau = 1/52$)



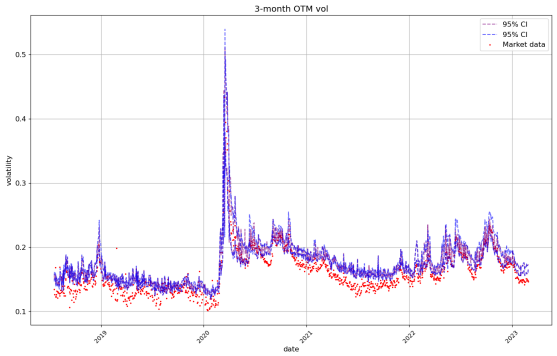
(c) 1-month ATM ($m = 1, \tau = 1/12$)



(d) 3-month ATM ($m = 1, \tau = 0.25$)



(e) 3-month ITM call ($m = 0.75, \tau = 0.25$)



(f) 3-month OTM call ($m = 1.25, \tau = 0.25$)

Figure 3.7: Implied volatility forecasts with 95% confidence intervals from VolGAN based on the 2.5% and 97.5% quantiles. SPX implied volatility market data (red), next-day forecast ($\mathbb{E}_\beta[\sigma_t(m, \tau)|a_{t-\Delta t}]$), confidence intervals shown in blue (without re-weighting) and purple (with re-weighting) where applicable.

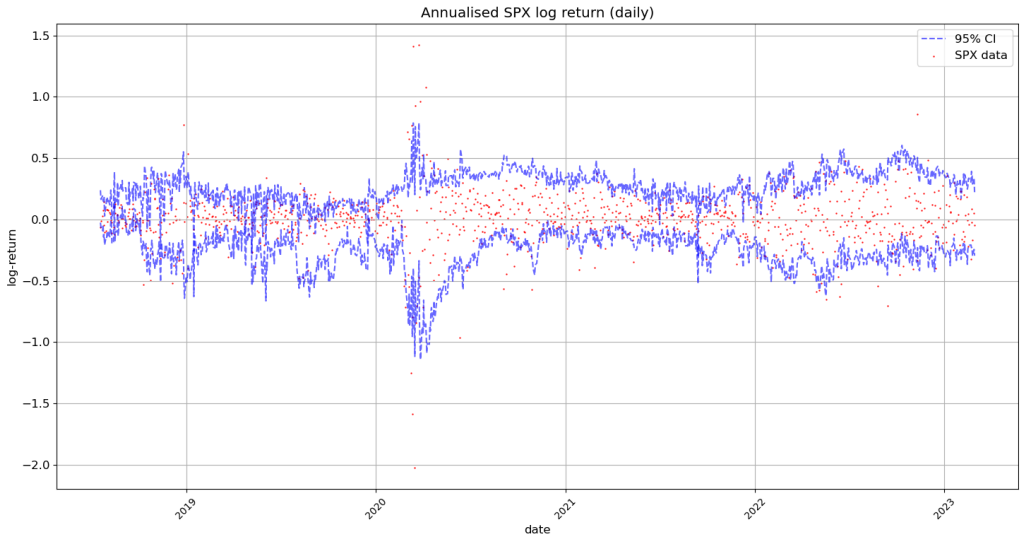


Figure 3.8: Realised and simulated SPX log-return on the test set. SPX implied volatility market data (red), next-day forecast ($\mathbb{E}_\beta[S_t|a_{t-\Delta t}]$) and the 95% confidence interval (blue: without re-weighting).

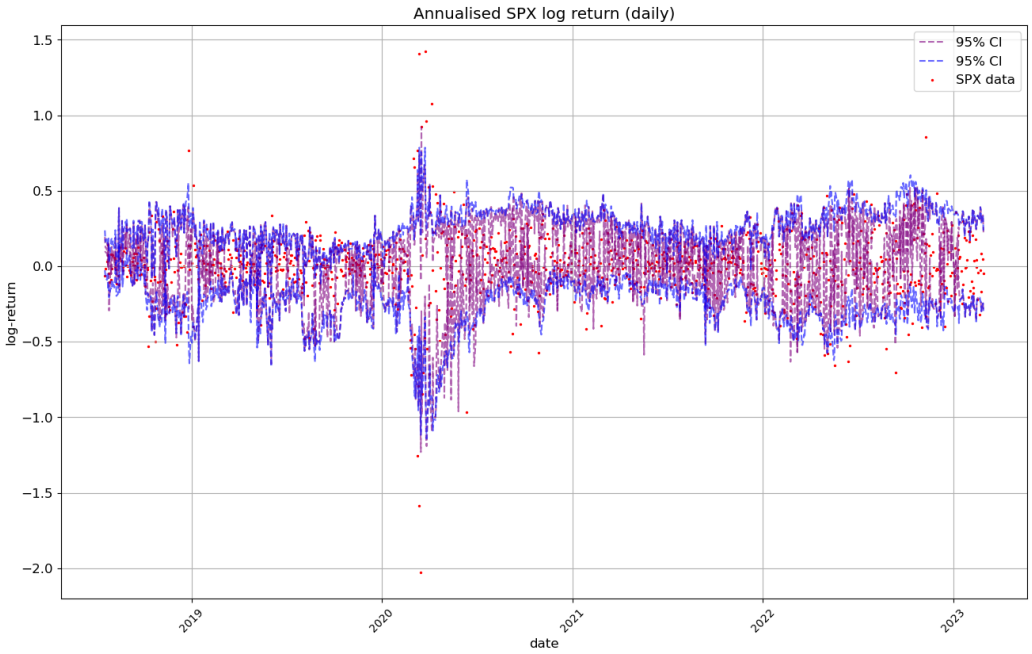


Figure 3.9: Realised and simulated SPX log-return on the test set. SPX implied volatility market data (red), next-day forecast ($\mathbb{E}_\beta[S_t|a_{t-\Delta t}]$) and the 95% confidence interval (blue: without re-weighting, purple: with re-weighting).

Figure 3.10: Historical vs one-day ahead simulation of VIX, on test data set.

We further investigate the prediction score in Table 3.2 by considering the percentage of data realisations falling below the simulated 1%, 2.5%, 97.5%, and 99% quantiles. We note that the best overall forecasts are for the underlying. VolGAN underestimates extremely high values of the implied volatility returns and VIX. Given that the volatility index is a non-linear transformation of the state variables, it is not surprising that VolGAN does not produce as stable confidence intervals as it does for the state variables. The findings from Table 3.2 are in line with the previous observations: VolGAN captures the state variables for which more data is available better. It is important to note that the observed behaviour is out-of-sample, four and a half years after training, including the 2020 data.

As already observed in Figure 3.9, there are instances (of market turbulence) where *not* correcting for the presence of static arbitrage (i.e. setting $\beta = 0$) actually *improves* forecasting performance. We note that when the arbitrage penalty is very low or zero, the penalisation has negligible impact on the simulated confidence intervals.

Table 3.2 shows that choosing $\beta = 0$ can, in fact, improve forecasts, especially for SPX returns, 1-week ATM volatility, and VIX.

Variable/Quantile	0.01	0.025	0.975	0.99
SPX return	25.32%	29.19%	82.00%	83.55%
3-month ATM vol	13.95%	15.16%	49.61%	54.61%
3-month OTM vol	76.978%	78.81%	92.85%	93.80%
3-month ITM vol	29.46%	30.32%	65.46%	69.34%
1-month ATM vol	9.82%	11.28%	42.89%	48.41%
1-week ATM vol	20.41%	22.05%	59.17%	63.22%
1-day ATM vol	19.90%	21.79%	60.12%	64.34%
VIX	34.37%	35.23%	52.67%	55.04%

Table 3.2: Exceedance ratio for VolGAN quantiles on the test set.

Variable/Quantile	0.01	0.025	0.975	0.99
SPX return	4.48%	9.39%	92.33%	93.37%
3-month ATM vol	8.52%	9.56%	64.51%	71.67%
3-month OTM vol	72.18%	73.64%	97.59%	98.02%
3-month ITM vol	20.33%	22.14%	75.62%	81.83%
1-month ATM vol	5.25%	6.55%	57.88%	66.58%
1-week ATM vol	11.80%	13.78%	72.95%	80.10%
1-day ATM vol	11.71%	13.52%	74.68%	81.65%
VIX	25.24%	25.84%	71.23%	71.18%

Table 3.3: Exceedance ratio for VolGAN quantiles on test set with $\beta = 0$.

3.3.2.4 Distributions and correlations learned by the generator

Denote by ρ_t the instantaneous correlation between the 1-month ATM volatility returns and the returns of the underlying at time t. We would like to explore whether VolGAN learns constant correlations. Therefore, we perform the following hypothesis test:

$$H_0\text{:}\quad \rho_t = \rho \text{ is constant}, \qquad H_1\text{:}\quad \rho_t \neq \rho \text{ is time-varying.}$$

By [15], under H_0, the 95% confidence interval for ρ_t is given by $[\rho^L, \rho^U]$, where

$$\rho^U = \frac{\exp(2z_U) - 1}{\exp(2z_U) + 1}, \quad \rho^L = \frac{\exp(2z_L) - 1}{\exp(2z_L) + 1};$$

$$z_U = \frac{1}{2}\log\left[\frac{1+\rho}{1-\rho}\right] + \sqrt{\frac{1}{n-3}}z_{0.975}, \qquad z_L = \frac{1}{2}\log\left[\frac{1+\rho}{1-\rho}\right] - \sqrt{\frac{1}{n-3}}z_{0.975},$$

where n is the sample size. Estimating ρ by the sample mean of ρ_t on the test set, in Figure 3.11 we plot ρ_t and the 95% confidence interval $[\rho^L, \rho^U]$. We note that ρ_t is away from the confidence interval of H_0, indicating strong evidence against H_0. VolGAN learns time-varying instantaneous correlations that would be difficult to capture with a parametric model.

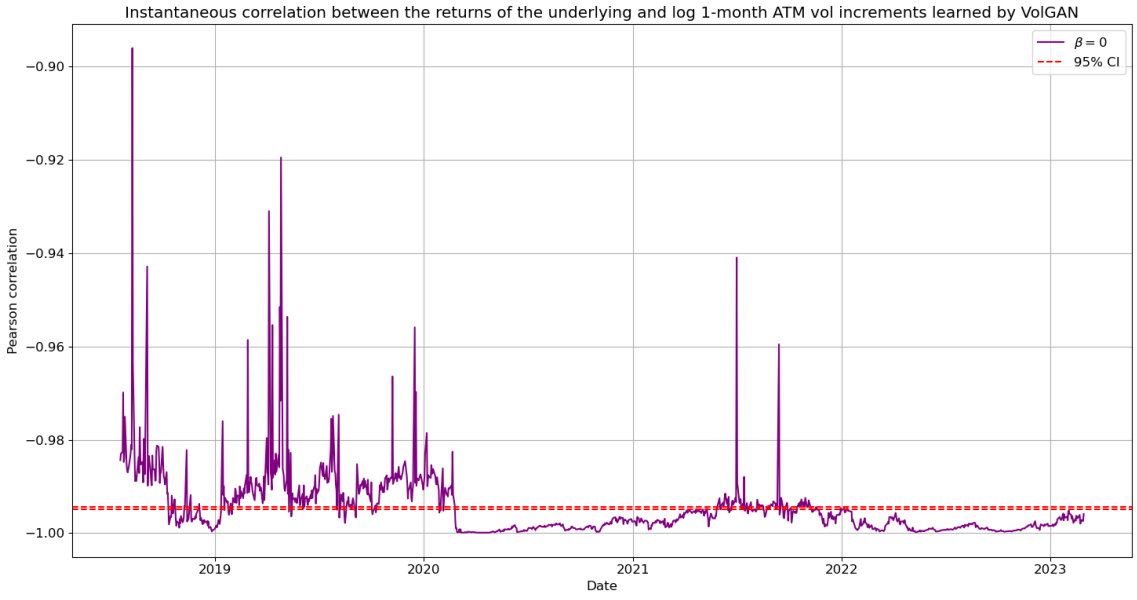
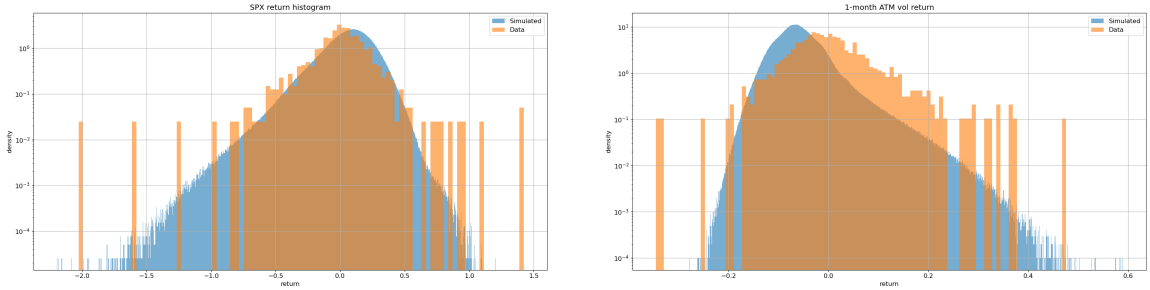


Figure 3.11: Pearson correlation between simulated index returns and 1-month ATM volatility increments (blue), with symmetric 95% confidence interval of constant correlation (red). VolGAN with $\beta = 0$.

We compare the (simulated) distributions of the daily returns for the underlying and 1-month ATM volatility with the corresponding empirical distributions and with

Gaussian distributions with the same mean and variance. Figures 3.12a and 3.12b show that simulated index returns and ATM volatility increments have asymmetric, non-Gaussian, and exponentially decaying tails. Such non-Gaussian, asymmetric distributions are difficult to capture in a model with Brownian increments.



(a) Index returns (b) 1-month ATM volatility increments

Figure 3.12: Histogram of simulated index returns (a) and 1-month ATM implied volatility increments (b) under VolGAN with $\beta = 0$ (blue). Both distributions exhibit asymmetric, exponentially decaying tails.

3.3.2.5 Principal component analysis

In order to investigate VolGAN's ability to appropriately capture the implied volatility co-movements, we perform out-of-sample principal component analysis on the simulated log increments of implied volatility. We compare the first three simulated principal components with the corresponding PCs of the data realisations. When performing PCA on four and a half years of SPX implied volatility data, the eigenvectors change depending on the period of observation, but nonetheless correspond to *level, skew* and *curvature*. In Table 3.4 we show variance explained by the first three eigenvectors in the testing data and in the VolGAN simulations. The significance of the first two principal components is very similar in the test data and in VolGAN. The third principal component is more significant in the simulated data compared to the SPX implied volatility market data.

Rank	Data	VolGAN
First	51.25%	$45.31 \pm 1.84\%$
Second	34.00%	$25.69 \pm 0.88\%$
Third	5.01%	$12.76 \pm 0.55\%$

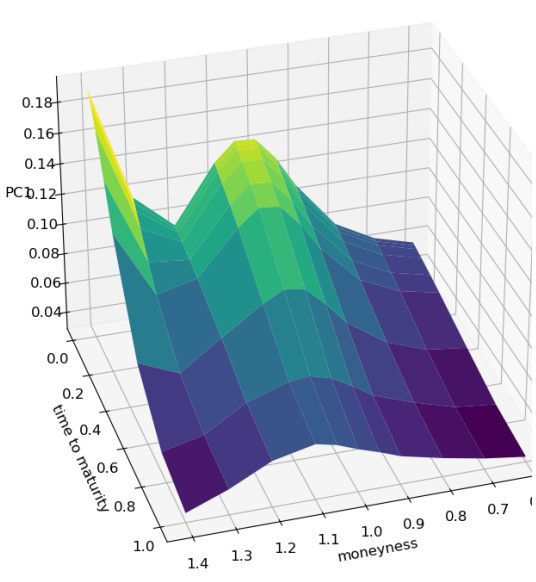
Table 3.4: Out-of-sample percentage of variance explained by the top three principal components of the simulated and the data log implied volatility increments. The VolGAN column contains the average $\pm 1.96\times$ standard deviation of the observed values, across 1000 VolGAN samples.

The first principal components of the sample VolGAN implied volatility log-returns and of the corresponding SPX implied volatility market data are displayed in Figure 3.13. Both surfaces are consistently positive, indicating that they might have a *level* interpretation. The second eigenvectors of both SPX implied volatility market data and of the simulated scenarios (Figure 3.14) can be interpreted as *skew*, while the third eigenvectors (Figure 3.15) can be interpreted as *curvature*. Figures 3.13, 3.14, 3.15 reflect on the clear resemblance between the principal components of the SPX implied volatility market data and of the VolGAN simulations, showing that VolGAN is able to dynamically learn the covariance structure of implied volatility co-movements.

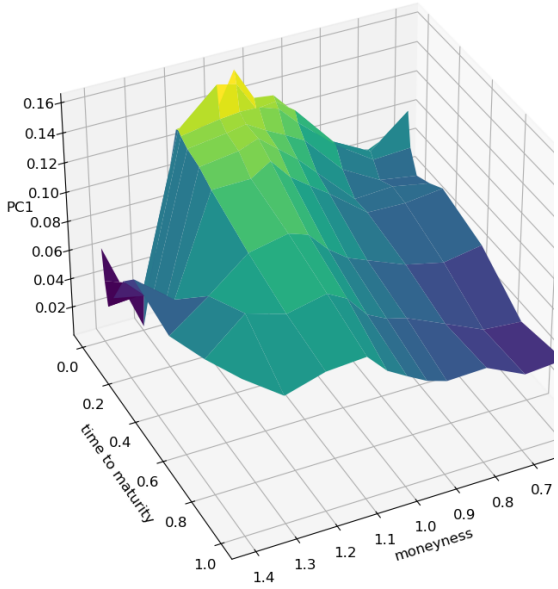
In order to quantify the similarity between the PCs of the simulated and the SPX implied volatility market data, we calculate the inner product between them (as vectors) over 1000 i.i.d. VolGAN samples. A value of one would indicate perfect alignment of the eigenvectors. From Table 3.5 we note that the first two inner products (PC1 with PC1, and PC2 with PC2) are very close to one, especially considering that the quantities are for the out-of-sample data. The inner product between the third eigenvectors of simulations and data realisations is lower than for the first two PCs, but it is nevertheless high. Furthermore, there is a close resemblance in the physical interpretations of the third eigenvectors. Therefore, VolGAN is able to learn the most important eigenvectors both qualitatively and quantitatively, showing the ability to learn the covariance structure of the SPX implied volatility co-movements.

Rank	Mean	Median	Standard deviation
First	0.921	0.922	0.009
Second	0.921	0.922	0.011
Third	0.798	0.798	0.011

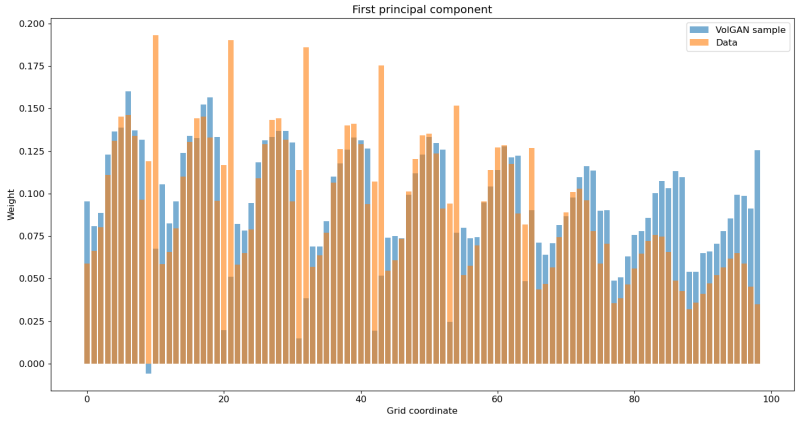
Table 3.5: Out-of-sample inner products of eigenvectors of the covariance matrices of daily log-returns of SPX implied volatility and the corresponding eigenvectors of the covariance matrix of VolGAN implied volatility increments.



(a) Computed using SPX implied volatility data.

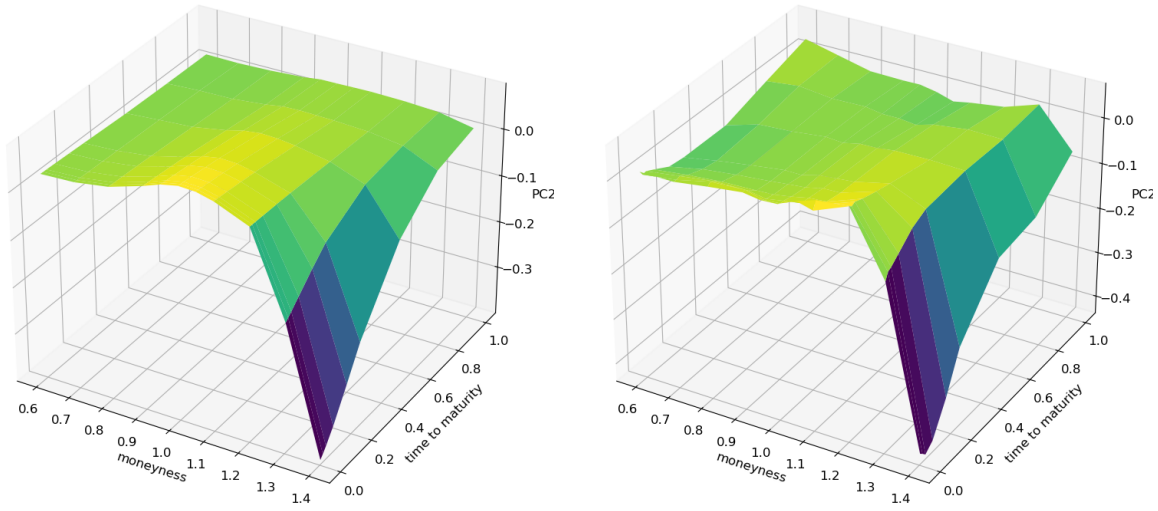


(b) Computed using a sample VolGAN output.



(c) Comparison of the first principal component in the data and in a sample simulation as vectors.

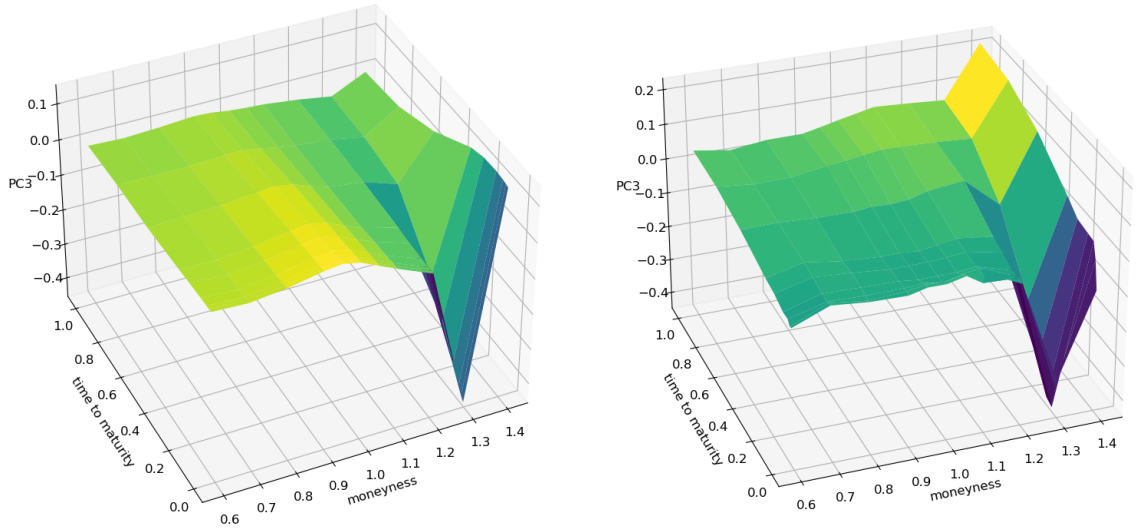
Figure 3.13: Out-of-sample first principal component of the daily log implied volatility increments.



(a) Computed using SPX implied volatility data.

(b) Computed using a sample VolGAN output.

Figure 3.14: Out-of-sample second principal component of the daily log implied volatility increments.



(a) Computed using SPX implied volatility data.

(b) Computed using a sample VolGAN output.

Figure 3.15: Out-of-sample third principal component of the daily log implied volatility increments.

3.3.2.6 Correlation structure of variables

We further investigate VolGAN's ability to simulate realistic scenarios by examining how well it reproduces correlations between variables of interest. First, we consider

the relationship between the projections of the log-implied volatility increments onto the first three principal components and the log-returns of the underlying.

Table 3.6 considers the correlations between index returns and the projections of the log-implied volatility increments onto the first three principal components, comparing their values in SPX options data with those in VolGAN scenarios. The correlation between the first projection process and the simulated log-returns of the underlying is close to that of market data, whereas the projections on the second and the third principal component have slightly stronger correlations with the returns of the underlying in VolGAN than they do in the SPX implied volatility market data. Nevertheless, both quantities are on the same scale. The correlation between the projection on the third principal component and the underlying is low both in VolGAN and in the options data.VolGAN is able to reproduce the correct relationships between the projection processes and the returns of the underlying: the correlations between the returns of the underlying and the projections of the log implied volatility increments onto the level and skew principal component are negative, while the correlation with the projection onto the curvature principal component is low (and positive).

PC rank	Data (test)	VolGAN (test)	Data (train)
First	-0.76	-0.84 ± 0.024	-0.34
Second	-0.29	-0.38 ± 0.055	-0.32
Third	0.06	0.16 ± 0.020	0.28

Table 3.6: Pearson correlation between (simulated) SPX log-returns and the projections of the (simulated) log-implied volatility increments on the principal components. The VolGAN column contains the mean $\pm 1.96\times$ standard deviation of the observed Pearson correlations across 1000 samples. Implied volatility increments in the *Data (train)* column are projected onto the principal components of the test data for consistency.

In order to correctly capture joint dynamics of implied volatilities and the underlying index, we are interested in the relationship between the log increments of the index ($\Delta \log S_t$), the projection of the log-implied volatility increments onto the first principal component (ΔX_t^1), the log increments of the 1-month at-the-money implied volatility ($\Delta \log \sigma_t^{ATM}$), and the log increments of VIX ($\Delta \log v_t$). Table 3.7 contains average Pearson correlations for VolGAN simulations (blue) vs the SPX implied volatility market data (red) on the test set. VolGAN simulations exhibit similar correlations between all variables of interest. The correlations between the

VIX increments and the increments of the other state variables are slightly lower in VolGAN scenarios compared to the data observation on the test set. However, they are of the correct sign and magnitude. The correlation between $\Delta \log S_t$ and $\Delta \log \sigma_t^{ATM}$ became significantly higher in magnitude in the period used for testing compared to the period used for training, as noted in Chapter 2, which could explain why VolGAN results in slightly stronger correlations between the the index returns and ΔX_t^1, that is $\Delta \log \sigma_t^{ATM}$.

	$\Delta \log S_t$	ΔX_t^1	$\Delta \log \sigma_t^{ATM}$	$\Delta \log v_t$
$\Delta \log S_t$	1.00	-0.84 -0.76	-0.86 -0.77	-0.55 -0.71
ΔX_t^1	-0.84 -0.76	1.00	0.95 0.89	0.66 0.84
$\Delta \log \sigma_t^{ATM}$	-0.86 -0.77	0.95 0.89	1.00	0.72 0.96
$\Delta \log v_t$	-0.55 -0.71	0.66 0.84	0.72 0.96	1.00

Table 3.7: Out-of-sample average Pearson correlation for simulated vs real values of log-returns of SPX ($\Delta \log S_t$), implied volatility level factor (ΔX_t^1), 1-month ATM volatility ($\Delta \log \sigma_t^{ATM}$) and VIX ($\Delta \log v_t$). Average VolGAN outcome (blue) and data (red).

We repeat the analysis for the first year in the test set in Table 3.8. We observe that the magnitude of the correlation between the log-increments of VIX and the log SPX returns is a bit lower in simulations compared to the data. In the last year of the test set (Feb 2022-Feb 2023), the correlations between the simulated values of log SPX returns, increments of the level factor, and the at-the-money vol returns increase in magnitude, as noted in Table 3.9. We observe that the same is true for the actual values stemming from the data. The correlation structure of the simulated variables is consistent with the market, regardless of the testing period.

	$\Delta \log S_t$	ΔX_t^1	$\Delta \log \sigma_t^{ATM}$	$\Delta \log v_t$
$\Delta \log S_t$	1.00	−0.67 −0.66	−0.73 −0.82	−0.34 −0.80
ΔX_t^1	−0.67 −0.66	1.00	0.89 0.75	0.64 0.74
$\Delta \log \sigma_t^{ATM}$	−0.73 −0.82	0.89 0.75	1.00	0.77 0.96
$\Delta \log v_t$	−0.34 −0.80	0.64 0.74	0.77 0.96	1.00

Table 3.8: First year out-of-sample average Pearson correlation for simulated vs real values of log-returns of SPX ($\Delta \log S_t$), implied volatility level factor (ΔX_t^1), 1-month ATM volatility ($\Delta \log \sigma_t^{ATM}$) and VIX ($\Delta \log v_t$). Average VolGAN outcome (blue) and data (red).

	$\Delta \log S_t$	ΔX_t^1	$\Delta \log \sigma_t^{ATM}$	$\Delta \log v_t$
$\Delta \log S_t$	1.00	−0.94 −0.80	−0.92 −0.72	−0.63 −0.76
ΔX_t^1	−0.94 −0.80	1.00	0.97 0.96	0.71 0.95
$\Delta \log \sigma_t^{ATM}$	−0.92 −0.72	0.97 0.96	1.00	0.76 0.95
$\Delta \log v_t$	−0.63 −0.76	0.71 0.95	0.76 0.95	1.00

Table 3.9: Last year out-of-sample average Pearson correlation for simulated vs real values of log-returns of SPX ($\Delta \log S_t$), implied volatility level factor (ΔX_t^1), 1-month ATM volatility ($\Delta \log \sigma_t^{ATM}$) and VIX ($\Delta \log v_t$). Average VolGAN outcome (blue) and data (red).

Our results demonstrate that VolGAN is able to simulate realistic co-movements for implied volatilities across a range of moneyness and maturities, as well as the underlying index and VIX. In particular, we are able to reproduce time-varying correlations between increments of these variables.

Chapter 4

Hedging with generative models

A natural application of generative models in finance is scenario generation for risk assessment of portfolios. In this chapter, we describe an approach for using generative models not only for risk measurement but also for the *hedging and risk management* of multi-asset portfolios. Using this data-driven approach, we can deploy generative models to learn hedging strategies from market data. We first describe a general methodology for data-driven hedging with generative models, then illustrate it with a worked-out example based on VolGAN.

The dominant paradigm in the design of hedging strategies is *model-based hedging*, which requires full specification of the dynamics of all risk factors affecting a portfolio over the risk horizon. Hedging is then formulated as an intertemporal optimisation or stochastic control problem, which is typically solved using dynamic programming or backward induction. Hedging strategies obtained in this manner are exposed to model uncertainty and may exhibit a significant level of 'model risk' [39].

Regression-based hedging is arguably the simplest form of 'data-driven' hedging: the idea is to perform a least squares regression of the daily changes of the target portfolio with respect to the daily changes of the hedging instruments and use the coefficients as hedge ratios [123, 43]. As shown by [43], regression-based hedging can exhibit superior performance when compared to model-based sensitivity hedging, especially in situations where models are arguably misspecified. However, historical regression-based hedge ratios are backward-looking, assume stationarity of covariance structure in co-movements, and may not properly account for changes in market conditions.

Starting with the pioneering work of Hutchinson, Lo and Poggio [74], who demonstrated that a simple one-layer neural network with four neurones could learn to hedge S&P500 options, neural networks have been used in various ways to calculate hedging strategies (see survey by [117] and references therein). As noted in [117], in most of

these studies neural networks are used to learn a pricing function and hedging is either not discussed or it is addressed using a sensitivity-based approach [36, 54]. In some cases, a neural network is trained to learn the hedge ratios directly from market data [29, 23, 118].

Deep Hedging [17] uses deep neural networks to learn optimal hedge ratios within a parametric model via reinforcement learning. This approach is able to incorporate transaction costs, alternative risk measures, and trading constraints but is not data-driven: training is done via deep reinforcement learning using (millions of) simulated scenarios based on a parametric model. Similar (model-based) approaches were explored by [18], [97], [94]. These approaches are based on *neuro-dynamic programming* [11], i.e., on the parameterisation of hedging strategies via a neural network, which is then trained using reinforcement learning. In contrast, our approach uses neural networks for the generation of market scenarios, while the computation of hedge ratios is done through direct optimisation, a much simpler approach which results in greater computational efficiency. Also, importantly, these approaches are model-based, not data-driven: training is done using scenarios simulated from a reference model, not on market data.

Our methodology is essentially different from the previously mentioned approaches: neural networks are used to generate forward-looking market scenarios, rather than to parameterise the hedging strategy. The latter is computed through an explicit optimisation step based on the generated scenarios.

As explained in Section 4.2, given the one-step ahead scenarios generated by the conditional generative model, we compute hedge ratios by solving a *local risk minimisation approach* which is a convex optimisation problem. Our optimisation criterion combines a conditional variance term with a ℓ_1 penalty for transaction costs. A similar approach was studied by [88], extending work of [56, 121] on quadratic hedging to include transaction costs. However, differently from these references which focus on (not necessarily self-financing) replication strategies computed by backward induction, our hedging strategy is self-financing and computed sequentially using one-step ahead scenarios, provided by the generative model. In particular, our approach does *not* require knowledge of the full price dynamics up to expiry and is thus applicable in sequential setting to conditional generative models based taking current market conditions as input. Our approach is similar in spirit to the model-predictive control approach [9] and the progressive hedging approach [114, 115], both of which employs scenario-based optimisation and proceed forward, not backward, in time.

Outline. Section 4.1 reviews commonly used approaches for computing hedging strategies. Section 4.2 describes our proposed methodology for data-driven hedging using generative models. Section 4.3 illustrates how our approach can be used to hedge volatility risk with VolGAN, a conditional scenario generator for implied volatility dynamics.

4.1 Sensitivity-based hedging vs optimisation-based hedging

We consider the problem of hedging a portfolio exposed to a set of risk factors using a set of *hedging instruments*. As in Chapter 1, we distinguish two sets of risk factors:

- market prices $S_t^1, ..., S_t^n$ of tradable primitive/underlying assets;
- other (non-price) risk factors denoted $X_t = (X_t^1, ..., X_t^d)$. Examples of such risk factors may be: implied volatilities, interest rates, credit spreads, etc.

Primitive assets are tradable, while risk factors $X_t^1, ..., X_t^d$ are not directly traded, i.e., do not represent prices of tradable instruments. Examples of such non-price risk factors are yields (for bonds) or implied volatilities for options.

We wish to hedge a portfolio whose value at time t is V_t, using a set $(H^i, i \in \mathcal{H})$ of *hedging instruments* whose values H_t^i are sensitive to these risk factors. We assume a rebalancing frequency Δt (for example, one day).

Our risk management framework requires the following ingredients:

1. a *pricing function* f which computes the value of the portfolio as a function of the risk factors

$$V_t = f(t, S_t, X_t) = f(t, S_t^1, ..., S_t^n, X_t^1, ..., X_t^d); \tag{4.1}$$

2. *pricing functions* for the hedging instruments:

$$H_t^j = h_j(t, S_t, X_t) = h_j(t, S_t^1, ..., S_t^n, X_t^1, ..., X_t^d); \tag{4.2}$$

3. a generative model capable of generating one-step ahead (e.g. one-day ahead) scenarios for co-movements of prices S_t and risk factors X_t

$$(S_t, X_t, S_{t-\Delta t}, X_{t-\Delta t}, ...), \text{noise term } \omega \quad \xrightarrow{\mathbf{G}} \quad (S_{t+\Delta t}(\omega), X_{t+\Delta t}(\omega)). \tag{4.3}$$

We consider a self-financing tracking portfolio with positions ϕ_t^i in the hedging instruments (H_t^i, $i \in \mathcal{H}$), and ψ_t in a money market account earning interest at rate r_t. The value Π_t of this tracking portfolio satisfies:

$$\Pi_{t+\Delta t} - \Pi_t = \Delta\Pi_t = \psi_t \, r_t \Delta t + \sum_{i \in \mathcal{H}} \phi_t^i (H_{t+\Delta t}^i - H_t^i), \quad \Pi_0 = V_0. \tag{4.4}$$

The self-financing condition implies

$$\psi_t = \Pi_t - \sum_{i \in \mathcal{H}} \phi_t^i H_t^i - \sum_{i \in \mathcal{H}} c_t^i |\phi_t^i - \phi_{t-\Delta t}^i|, \tag{4.5}$$

where c_t^i is the cost of trading one unit of H^i at time t. The **tracking error** Z_t is the value of the hedged position:

$$Z_t = V_t - \Pi_t, \quad Z_0 = 0. \tag{4.6}$$

The two main approaches to determine the hedging strategies ϕ_t are *sensitivity-based hedging* and *conditional risk minimisation*, with (local) quadratic hedging being a special case of the latter. We now describe these two approaches in some detail.

4.1.1 Sensitivity-based hedging

The most common approach for constructing hedging strategies is based on *sensitivities* to risk factors. Given a 'standard' shift ϵ_i for risk factor i, we define the sensitivity to a risk factor i as the change in portfolio value under a 'standardised shift' $X^i \to X^i + \epsilon_i$ to the risk factor $i \in \{1, \ldots, d\}$. Denote the sensitivity to risk factor X^i at time t by

$$\zeta_t^i(f) = \frac{f(t, S_t, X_t^0, \ldots, X_t^{i-1}, X_t^i + \epsilon_i, X_t^{i+1}, \ldots, X_t^d) - f(t, S_t, X_t)}{\epsilon_i}. \tag{4.7}$$

Similarly, denote sensitivities to market prices S^i, for $i \in \{1, \ldots, n\}$ by

$$\Delta_t^i(f) = \frac{f(t, S_t^0, \ldots, , S_t^{i-1}, S_t^i + \epsilon_j, S_t^{i+1}, \ldots, S_t^n, X_t) - f(t, S_t, X_t)}{\epsilon_j}. \tag{4.8}$$

If the pricing function f is differentiable in S_t, X_t, then

$$\lim_{\epsilon_i \to 0} \zeta_t^i(f) = \frac{\partial f}{\partial X^i}(t, S_t, X_t), \qquad \lim_{\epsilon_i \to 0} \Delta_t^i(f) = \frac{\partial f}{\partial S^i}(t, S_t, X_t),$$

Analogously, we define sensitivities of the hedging instruments H^j to risk factors X^i and market prices S^i, which we denote by $\zeta_t^i(h^j)$ and $\Delta_t^i(h^j)$, respectively. In cases where neural networks represent the pricing functions, sensitivities are readily computable using Automatic Differentiation (AD) techniques [55, 20].

The overall sensitivity to a risk factor X^i of the tracking portfolio Π_t is obtained by summing over positions:

$$\zeta_t^i(h) = \sum_{j \in \mathcal{H}} \phi_t^i \zeta_t^i(h^j),$$

and its sensitivity to S^i is

$$\Delta_t^i(h) = \sum_{j \in \mathcal{H}} \phi_t^i \Delta_t^i(h^j).$$

The idea of sensitivity-based hedging is to choose the positions in the hedging instruments to achieve zero/low overall sensitivity to risk factors (S_t, X_t). The hedged portfolio is immunised to small movements in the risk factor X^i for $i = 1, \ldots, d$ if the sensitivity of the hedged position $Z_t = V_t - \Pi_t$ is zero:

$$\zeta_t^i(f) = \zeta_t^i(h). \tag{4.9}$$

Similarly, the hedged portfolio is immunised to small changes in market prices S_t^i for $i = 1, \ldots, n$ if

$$\Delta_t^i(f) = \Delta_t^i(h). \tag{4.10}$$

Examples of commonly used sensitivity-based hedging strategies include delta hedging and delta-vega hedging of option portfolios, and immunisation strategies of bond portfolios.

Hedging via risk immunisation requires the user to pre-specify both the risk factors and the hedging instruments, a choice that is not necessarily unique. For instance, in delta-vega hedging, volatility risk can be mitigated by holding a position in *any* option on the same underlying asset; sensitivity considerations alone do not guide us in the choice of the hedging instrument.

Additionally, risk immunisation typically accounts for only small perturbations in risk factors. Extensions of this approach consider inclusion of higher order sensitivities, for example gamma hedging for options or duration-convexity immunisation for bond portfolios [48]. However, even these extensions cannot account for tail events.

Finally, these shifts are applied to individual risk factors in isolation, disregarding their *co-movements*. If risk factors are correlated, this may mean that the scenarios underlying these sensitivity computations are unrealistic.

4.1.2 Hedging by local risk minimisation

Another approach for computing hedging strategies is based on the idea of *local risk minimisation* [56, 88, 100, 121]. In this approach, one sequentially computes hedge

ratios to minimise a one step-ahead risk criterion, based on current information in market variables.

As in Chapter 1, denote by $\{\mathcal{F}_t\}_{t\geq 0}$ the filtration representing the information contained in the history of market prices and risk factors up to time t. For a hedging strategy to be implementable, the hedge ratios ϕ_t have to be $\mathcal{F}_t$-measurable. The idea of local risk minimisation is to compute ϕ_t by minimising a risk measure ρ of the portfolio conditional on current market information

$$\inf_{\phi_t} \ \rho(Z_{t+\Delta t}|\mathcal{F}_t).$$

A tractable class is given by conditional risk measures expressible in the form

$$\rho(Z_{t+\Delta t}|\mathcal{F}_t) = \mathbb{E}\left[L(Z_{t+\Delta t}, Y_t)|\mathcal{F}_t\right]. \tag{4.11}$$

where L is a loss function and Y_t is observable at t, that is, $\{\mathcal{F}_t\}$ -adapted. In particular, (4.11) is amenable to computation by simulation.

An important example of a conditional risk measure expressible in the form (4.11) is the conditional variance:

$$\rho(Z_{t+\Delta t}|\mathcal{F}_t) = \mathbb{E}\left[\left(Z_{t+\Delta t} - \mathbb{E}[Z_{t+\Delta t}|\mathcal{F}_t]\right)^2\middle|\mathcal{F}_t\right], \tag{4.12}$$

where

$$L(Z,Y) = (Z-Y)^2, \quad Y_t = \mathbb{E}[Z_{t+\Delta t}|\mathcal{F}_t].$$

Remark 3. *One may of course consider other risk measures, such as Value-at-Risk (VaR), Expected Shortfall, etc. However, among these examples the only risk measure expressible in the form* (4.11) *is conditional variance.*

Minimising the conditional variance of the hedging error (conditional on market variables $\mathcal{F}_t$) leads to a (local) **regression problem**. The objective is to solve

$$\min_{\phi_t} \operatorname{var}\left(Z_{t+\Delta t}|\mathcal{F}_t\right). \tag{4.13}$$

Proposition 2. *The hedge ratios which minimise the variance of the tracking error $Z_{t+\Delta t}$ conditional on $\mathcal{F}_t$ are given as regression coefficients of ΔV_t on $\{\Delta H_t^i\}_{i\in\mathcal{H}}$:*

$$\min_{\phi_t} \operatorname{var}\left(Z_{t+\Delta t}|\mathcal{F}_t\right) = \min_{A_t,\phi_t} \mathbb{E}\left[\left(V_{t+\Delta t} - V_t - A_t - \sum_{i\in\mathcal{H}} \phi_t^i (H_{t+\Delta t}^i - H_t^i)\right)^2 \middle| \mathcal{F}_t\right]. \tag{4.14}$$

Proof. Since ψ_t, V_t, and ϕ_t^i, H_i^t are $\mathcal{F}_t$-measurable, we have

$$\begin{aligned}\operatorname{var}\left(Z_{t+\Delta t}|\mathcal{F}_t\right) &= \operatorname{var}\left(Z_{t+\Delta t} - Z_t|\mathcal{F}_t\right) = \operatorname{var}\left(\Delta V_t - \Delta\Pi_t|\mathcal{F}_t\right) \\ &= \operatorname{var}\left(\Delta V_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i - \psi_t r_t \Delta t \middle| \mathcal{F}_t\right) \\ &= \operatorname{var}\left(\Delta V_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right).\end{aligned}$$

In fact, for any $\mathcal{F}_t$-measurable A_t, the following holds:

$$\operatorname{var}\left(\Delta V_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right) = \operatorname{var}\left(\Delta V_t - A_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right).$$

Hence, we can choose A_t such that

$$\mathbb{E}\left[\Delta V_t - A_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right] = 0, \qquad i.e. \quad A_t = \mathbb{E}\left[\Delta V_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right]. \tag{4.15}$$

Furthermore, for any $\mathcal{F}_t$-measurable α_t:

$$\begin{aligned}\mathbb{E}\left[\left(\Delta V_t - \alpha_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i\right)^2 \middle| \mathcal{F}_t\right] &= \operatorname{var}\left(\Delta V_t - \alpha_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right) \\ &\quad + \mathbb{E}\left[\Delta V_t - \alpha_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right]^2 \\ &= \operatorname{var}\left(\Delta V_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right) \\ &\quad + \mathbb{E}\left[\Delta V_t - \alpha_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right]^2.\end{aligned} \tag{4.16}$$

Hence, the value of α_t which minimises the conditional second moment (4.16) of $\Delta V_t - \alpha_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i$ is precisely the conditional expectation A_t given by (4.15):

$$\alpha_t^* = \operatorname{argmin}_\alpha \mathbb{E}\left[\left(\Delta V_t - \alpha - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i\right)^2 \middle| \mathcal{F}_t\right],$$

$$\text{i.e.} \qquad \alpha_t^* = \mathbb{E}\left[\Delta V_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i \middle| \mathcal{F}_t\right] = A_t.$$

This implies that

$$\operatorname{var}\left(Z_{t+\Delta t}|\mathcal{F}_t\right) = \min_{\alpha_t} \mathbb{E}\left[\left(\Delta V_t - \alpha_t - \sum_{i\in\mathcal{H}} \phi_t^i \Delta H_t^i\right)^2 \middle| \mathcal{F}_t\right].$$

This is an Ordinary Least Squares (OLS) regression, conditional on the information available at time t:

$$\Delta V_t = A_t + \sum_{i \in \mathcal{H}} \phi_t \Delta H_t^i + \epsilon_t,$$

where ϵ_t is uncorrelated with the P&Ls of the hedging instruments and $\mathbb{E}\left[\epsilon_t | \mathcal{F}_t\right] = 0$.

So, finally, we have shown that minimising the conditional variance of the hedging error (conditional on market variables $\mathcal{F}_t$) leads to a least squares **regression problem**:

$$\min_{\phi_t} \operatorname{var}\left(Z_{t+\Delta t} | \mathcal{F}_t\right) = \min_{A_t, \phi_t} \mathbb{E}\left[\left(V_{t+\Delta t} - V_t - A_t - \sum_{i \in \mathcal{H}} \phi_t^i (H_{t+\Delta t}^i - H_t^i)\right)^2 \middle| \mathcal{F}_t\right]. \quad (4.17)$$

□

Remark 4 (Self-financing condition)**.** Equation (4.14), determines the hedge ratios which minimise the conditional variance of the tracking error. The hedging strategy requires to specify as well the cash/money market component ψ_t. This is determined by the self-financing condition (4.5). Any other choice for ψ_t would lead to a *non self-financing* hedging strategy. For example [121, 56, 88] choose $\psi_t = A_t$ given by (4.15), which yields the martingale property for the tracking error but results in the loss of self-financing property.

Remark 5. Note that we are not operating under a 'risk-neutral' measure and there is no martingale assumption on price dynamics.

Remark 6. The conditional variance minimisation problem (4.13) is a regression of the *change* in portfolio value ΔV_t on the changes ΔH_t^i in values of the hedging instruments, conditional on the information available at time t. Using the $\mathcal{F}_t$-measurability of V_t, Π_t, and H_t, we can also express (4.13) as a regression of values, rather than changes, of these same instruments:

$$\min_{\phi_t} \operatorname{var}\left(Z_{t+\Delta t} | \mathcal{F}_t\right) = \min_{A_t, \phi_t} \mathbb{E}\left[\left(V_{t+\Delta t} - A_t - \sum_{i \in \mathcal{H}} \phi_t^i H_{t+\Delta t}^i\right)^2 \middle| \mathcal{F}_t\right]. \quad (4.18)$$

4.1.3 Accounting for transaction costs

Since the transaction cost term $\sum_{i \in \mathcal{H}} c_t^i |\phi_t^i - \phi_{t-\Delta t}^i|$ is $\mathcal{F}_t-$measurable, it contributes to the conditional mean but not to the conditional variance. The optimisation problem (4.14) therefore does not penalise transaction costs. In order to account for transaction costs, one may proceed in different ways. A common approach is to minimise the

conditional second moment (instead of variance) [121, 100, 101]. This leads to an objective function with quadratic penalties for transaction costs, but also cross-terms of the type

$$c_t^i|\phi_t^i - \phi_{t-\Delta t}^i|\ \phi_t^j \Delta H_t^j, \qquad i \neq j,$$

which lacks a financial interpretation and leads to numerical difficulties. Another approach is to use conditional variance, but incorporate transaction costs for the *next* period as in [88]. This leads to a variance contribution, but breaks down the analytical tractability of the quadratic problem (4.14). Furthermore, in this case, we lose the interpretation of the objective as variance of the tracking error.

We propose instead a different formulation, which is to include transaction costs as a penalty in the objective function, leading to:

$$\min_{A_t,\phi_t} \mathbb{E}\left[\left(V_{t+\Delta t} - V_t - A_t - \sum_{i\in\mathcal{H}} \phi_t^i (H_{t+\Delta t}^i - H_t^i)\right)^2 \middle| \mathcal{F}_t\right] + \lambda \sum_{i\in\mathcal{H}} c_t^i |\phi_t^i - \phi_{t-\Delta t}^i|, \quad (4.19)$$

where $\lambda > 0$ is a regularisation parameter. A similar approach has been considered by [93] but in a global risk minimisation setting. In (4.19), each term has a clear financial interpretation. Since A_t does not appear in the second term, by Proposition (2), the first term is still the conditional variance of the tracking error, while the second term is proportional to the transaction cost. As the transaction cost is naturally expressed as a (weighted) ℓ_1 norm, (4.19) is in fact a LASSO regression [129]. This leads to the desirable property of *sparsity* which, in financial terms, corresponds to selecting a sparse set of hedging instruments.

4.2 Dynamic data-driven hedging with generative models

An important feature of the local risk minimisation problem (4.19) is that it only requires knowledge of the one step-ahead conditional distribution. However, this conditional distribution remains inaccessible, rendering the local risk minimisation problem analytically intractable even in simple models. On the other hand, we will demonstrate that this problem may be efficiently solved using conditional generative models. We will now show how to sequentially compute locally risk-minimising hedging strategies using a conditional generative model.

A conditional generative model G for (S_t, X_t) allows generating samples $(S_{t+\Delta t}(\omega), X_{t+\Delta t}(\omega))$ whose distribution approximates the conditional distribution of $(S_{t+\Delta t}, X_{t+\Delta t})$ given

$\mathcal{F}_t$. These generated samples are forward-looking one step-ahead scenarios, which may be used to estimate the conditional mean and variance of the tracking error via

$$\widehat{A}_t = \frac{1}{N}\sum_{k=1}^{N}\left(\Delta V_t(\omega_k) - \sum_{i\in\mathcal{H}}\phi_t^i \Delta H_t^i(\omega_k)\right),$$
$$\widehat{\mathrm{var}(\Delta Z_t|\mathcal{F}_t)} = \frac{1}{N}\sum_{k=1}^{N}\left(\Delta V_t(\omega_k) - \widehat{A}_t - \sum_{i\in\mathcal{H}}\phi_t^i \Delta H_t^i(\omega_k)\right)^2,$$

where $\{\omega_k\}_{k=1,..,N}$ are N i.i.d. scenarios from the generator G. We can then compute an estimator of (4.19), and compute hedge ratios by optimising this estimator.

This leads to the following sequential optimisation problem. At each step t, given previous hedge ratios $\phi_{t-\Delta t}$, we solve the following LASSO regression problem:

$$\inf_{A_t\in\mathbb{R},\phi_t\in\mathbb{R}^{d+1}} \underbrace{\frac{1}{N}\sum_{k=1}^{N}\left(\Delta V_t(\omega_k) - A_t - \sum_{i\in\mathcal{H}}\phi_t^i \Delta H_t^i(\omega_k)\right)^2}_{\text{Hedging error variance}} + \alpha g_0 \underbrace{\sum_{i\in\mathcal{H}} c_t^i \left|\phi_t^i - \phi_{t-\Delta t}^i\right|}_{\text{Rebalancing cost}}, \tag{4.20}$$

where

- c_t^i is the cost of trading one unit of H^i at time t (we use the half the bid-ask spread for instrument H^i as a proxy);
- g_0 is the initial gross position, and $\alpha > 0$ is a dimensionless regularisation parameter.

The solution to (4.20) can be computed via coordinate descent with soft thresholding [65].

The regularisation term in (4.20) favours lower turnover and hedging instruments H^i with lower transaction costs. Transaction costs lead to an ℓ_1-penalty, which is known to induce sparsity in regression: it leads to the "minimal" (sub)set of rebalancing transactions. Sparsity means that not all potential hedging instruments are selected, leading to an **automatic selection of hedging instruments**.

Given that hedging instruments in $\mathcal{H}$ might have highly correlated returns (especially if $|\mathcal{H}| = J$ is large), regularisation is useful in (4.20). Furthermore, other (position, sensitivity) constraints can be easily incorporated in the optimisation problem.

The resulting procedure for data-driven hedging is described in Algorithm 1.

Algorithm 1 Data-driven hedging with no market impact

Input:

- Market prices S_t and risk factors X_t at time t;
- Pricing functions $f(t, S_t, X_t)$ and $h_i(t, S_t, X_t)$ for $i \in \mathcal{H}$;
- A generative model G for simulating $(S_{t+\Delta t}, X_{t+\Delta t})$;
- Previous hedge ratios $\phi_{t-\Delta t}$;
- Regularisation parameter α;
- Initial gross position g_0.

Output: Hedge ratios $(\phi_t^i)_{i\in\mathcal{H}}$, cash position ψ_t, estimate A_t.

Step 1: *Simulate forward-looking scenarios.*

for $k = 1$ to N **do**
  $(S_{t+\Delta t}(\omega_k), X_{t+\Delta t}(\omega_k)) \leftarrow G(S_t, X_t, \omega_k)$
  $V_{t+\Delta t}(\omega_k) \leftarrow f(t+\Delta t, S_{t+\Delta t}(\omega_k), X_{t+\Delta t}(\omega_k))$
  for each $i \in \mathcal{H}$ **do**
    $H^i_{t+\Delta t}(\omega_k) \leftarrow h_i(t+\Delta t, S_{t+\Delta t}(\omega_k), X_{t+\Delta t}(\omega_k))$
  end for
end for

Step 2: *Solve regularised regression (LASSO).*

Estimate A_t and hedge ratios $\phi_t = (\phi_t^i)_{i\in\mathcal{H}}$ by solving:

$$\min_{A_t\in\mathbb{R},\, \phi_t\in\mathbb{R}^J} \frac{1}{N}\sum_{k=1}^{N}\left(\Delta V_t(\omega_k) - A_t - \sum_{i\in\mathcal{H}}\phi_t^i \Delta H_t^i(\omega_k)\right)^2 + \alpha g_0 \sum_{i\in\mathcal{H}} c_t^i \left|\phi_t^i - \phi_{t-\Delta t}^i\right|$$

Step 3: *Update the cash position.*

$\psi_t = \Pi_t - \sum_{i\in\mathcal{H}}\phi_t^i H_t^i - \sum_{i\in\mathcal{H}} c_t^i \left|\phi_t^i - \phi_{t-\Delta t}^i\right|$

Return: A_t, $(\phi_t^i)_{i\in\mathcal{H}}$, ψ_t.

Remark 7 (Incorporating market impact)**.** *Transaction costs do not scale linearly with volume for large portfolios [137]. To take this into account, we can incorporate a model for market impact. Then s_t^i would depend on quantities such as average daily traded volume, volatility of the hedging instrument, value of the hedging instrument at time t, the rebalancing amount $|\phi_t^i - \phi_{t-\Delta t}^i|$, and more. If the price impact is proportional to $|\phi_t^i - \phi_{t-\Delta t}^i|^\delta$, then*

$$\Delta\Pi_t = \sum_{i\in\mathcal{H}}\phi_t^i\Delta H_t^i + r_t\left(\Pi_t - \sum_{i\in\mathcal{H}}\phi_t^i H_t^i - \overbrace{\sum_{i\in\mathcal{H}}\left|\phi_t^i - \phi_{t-\Delta t}^i\right|^{1+\delta} s_t^i}^{\text{Rebalancing cost}}\right)\Delta t, \tag{4.21}$$

where in this case s_t^i is the prefactor in the price impact model and should be expressed in USD. For options, the impact should be measured in terms of implied volatility. Incorporation of market impact as in (4.21) *leads to an alternative optimisation problem, where for $\lambda > 0$ we wish to solve:*

$$\inf_{A_t\in\mathbb{R},\phi_t\in\mathbb{R}^{d+1}} \frac{1}{N}\sum_{k=1}^{N}\left(\Delta V_t(\omega_k) - A_t - \sum_{i\in\mathcal{H}}\phi_t^i\Delta H_t^i(\omega_k)\right)^2 + \lambda g_0 \sum_{i\in\mathcal{H}} s_t^i\left|\phi_t^i - \phi_{t-\Delta t}^i\right|^{1+\delta}. \tag{4.22}$$

In case of linear impact ($\delta = 1$), (4.22) leads to a 'Ridge regression' problem [71] which has a closed-form solution.

However, the values of $\delta > 0$ do not lead to sparsity in (4.22). Automatic hedging instrument selection can be implemented by first performing a LASSO regression as in (4.20) at time $t = 0$ in order to determine the hedging instruments, and then using the selected instruments to solve (4.22). Alternatively, for $\delta > 0$, one could consider an 'Elastic Net' regression [65]:

$$\begin{aligned}\inf_{A_t\in\mathbb{R},\phi_t\in\mathbb{R}^{d+1}} & \underbrace{\frac{1}{N}\sum_{k=1}^{N}\left(\Delta V_t(\omega_k) - A_t - \sum_{i\in\mathcal{H}}\phi_t^i\Delta H_t^i(\omega_k)\right)^2}_{\text{Hedging error variance}} \\ & + \lambda g_0 \underbrace{\sum_{i\in\mathcal{H}} s_t^i\left|\phi_t^i - \phi_{t-\Delta t}^i\right|^{1+\delta}}_{\text{Market impact}} + \alpha g_0 \underbrace{\sum_{i\in\mathcal{H}} c_t^i\left|\phi_t^i - \phi_{t-\Delta t}^i\right|}_{\text{Rebalancing cost}}.\end{aligned} \tag{4.23}$$

4.3 Example: hedging volatility risk with VolGAN

The above methodology is general, but we will illustrate it with an example: hedging of option portfolios using VolGAN [134], as outlined in Chapter 3. We use raw VolGAN outputs, without scenario re-weighting [45], introduced in Chapter 2.

The portfolio we wish to hedge is a one-month long straddle with strikes $K = m_0 S_0$, for $m_0 \in \{0.75, 0.8, 0.9, 1.1, 1.2, 1.25\}$. The initial portfolio instrument set indexed by $\mathcal{P}$ consists of a $Call(K,T)$ and a $Put(K,T)$, with $K = m_0 S_0$ and $T = 1/12$.

In addition to data-driven hedging through VolGAN, we also consider delta hedging and delta-vega hedging. The hedging experiment is performed over non-overlapping periods. That is, a long straddle is entered and hedged until expiry, after which the new long straddle is entered and the exercise is repeated. This results in 52 non-overlapping one-month periods. One day is taken to be $\Delta t = 1/252$.

The initial hedging set indexed by $\mathcal{H}_0$ consists of one-month calls and puts whose initial moneyness values are in the set $\{0.9, 0.95, 0.975, 1, 1.025, 1.05, 1.1\}$, where $m < 1$ correspond to puts and $m \geq 1$ to calls. The hedging set indexed by $\mathcal{H}$ consists of all of the options in the initial hedging set that are not in the portfolio we wish to hedge, that is $\mathcal{H} = \mathcal{H}_0 \setminus \mathcal{P}$. The values α used for regularisation are obtained

through validation on new independent samples from VolGAN, as discussed in the next subsection.

Delta-vega hedging The hedging instruments are the underlying and an option. Denoting by κ_t^i the vega of the option $i \in \mathcal{P}$ in the initial portfolio we wish to hedge (long straddle) at time t, the overall vega of the portfolio is

$$\kappa_t^V = \sum_{i \in \mathcal{P}} \psi^i \kappa_t^i.$$

In order to obtain vega-neutrality, it is necessary to include an option H_t^1 with a vega of κ_t^H whose hedge ratio is determined by the ratio of sensitivities:

$$\phi_t^1 = \frac{\kappa_t^V}{\kappa_t^H}.$$

Delta-neutrality is obtained by adjusting the position in the underlying. Denoting by Δ_t^V and Δ_t^H the deltas of the portfolio and the option used for hedging, the hedge ratio ϕ_0 corresponding to the underlying is therefore:

$$\phi_t^0 = \Delta_t^V - \phi_t^1 \Delta_t^H.$$

We opt to hedge with an option initiated at-the-money, i.e. with a strike $K = S_0$. However, the choice of the hedging instrument is not unique. Opting for options with moneyness far away from one, or in general with low vega, may result in unstable hedge ratios ϕ_t^1.

Delta hedging The only hedging instrument is the underlying. The hedge ratio is the portfolio delta:

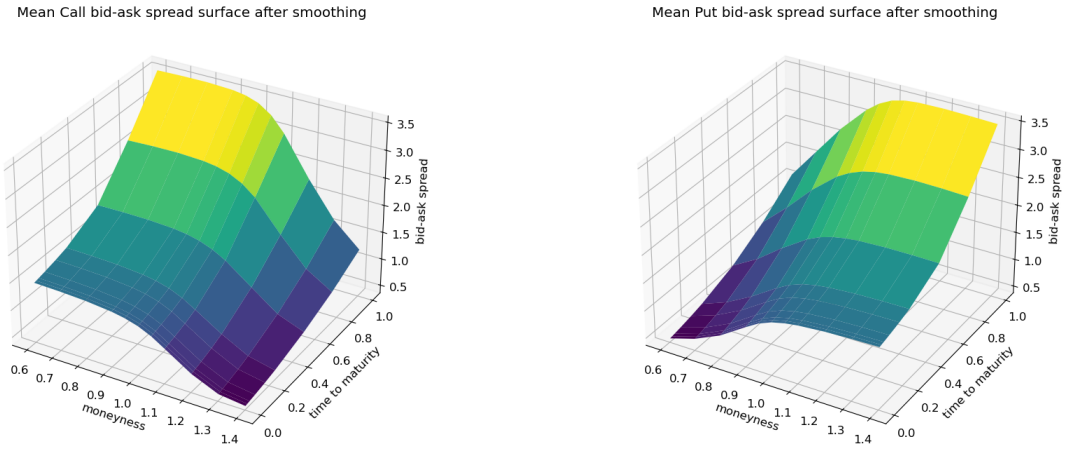
$$\phi_t^0 = \Delta_t^V.$$

4.3.1 Data

We use data on SPX options extracted from OptionMetrics, as in Chapter 3. The average bid-ask spread surfaces for calls and puts after smoothing are shown in Figure 4.1. On average, longer-dated in-the-money options have a higher bid-ask spread than out-of-the-money options with shorter times to expiry. We observe a skew in both the moneyness and time-to-maturity variables.

We compare the average at-the-money bid-ask-spread with the arbitrage penalty (2.9) in Figure 4.2, and note that the instances of non-zero arbitrage penalty coincide

with high bid-ask spread values. This observation is consistent with the notion of no-arbitrage, since the arbitrage penalty is calculated using implied volatilities of mid-prices.



(a) Calls. (b) Puts.

Figure 4.1: Average bid-ask spread for calls and puts.

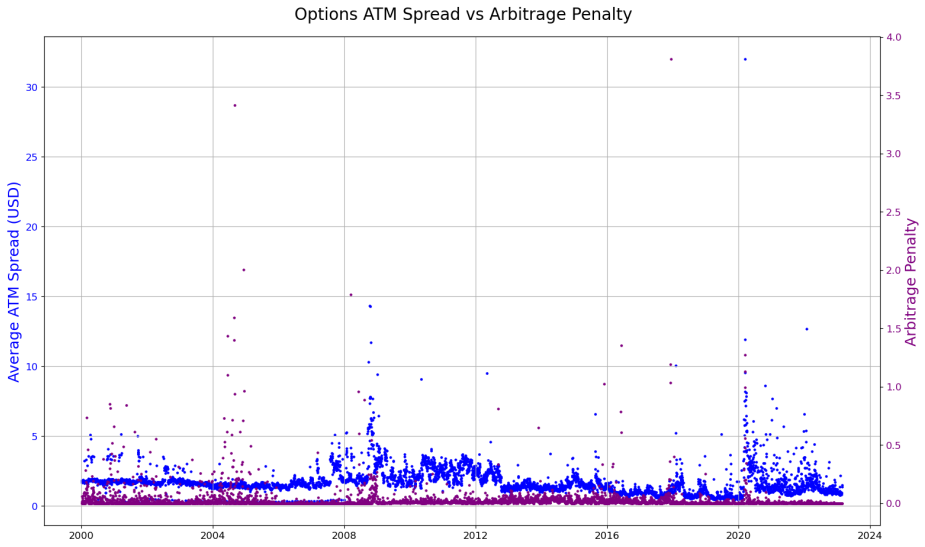


Figure 4.2: Average at-the-money bid-ask spread and the arbitrage penalty (2.9).

4.3.2 Choosing the regularisation parameter

High values of the regularisation parameter α would result in low rebalancing due to the high influence of the transaction costs, whereas low values of α would not provide sufficient regularisation and would lead to high transaction costs. That is, low values

of α might result in overfitting, and high values of α in underfitting. In order to set the appropriate value of α, in each one-month experiment, we perform a search at time $t = 0$ considering the potential α values in $\{0.01, 0.02, \ldots, 0.2\}$, which is a scale similar to that of the transaction-free hedging analysis of [134]. Let $\hat{A}_0(\alpha), (\hat{\phi}_0^i(\alpha))_{i\in\mathcal{H}}$ be the solutions of (4.20) given α, and let $\{\Delta V_0(\omega_j), (\Delta H_0^i(\omega_j))_{i\in\mathcal{H}}\}_{j=1,\ldots,1000}$ i.i.d. samples from VolGAN for each starting time $t = 0$, used for regression fitting. We choose α minimising the Akaike Information Criterion (AIC) [2] for the day on which a new position is entered ($t = 0$), calculated on new $M = 100$ i.i.d. samples from VolGAN $\{\Delta V_0(\omega_j), (\Delta H_0^i(\omega_j))_{i\in\mathcal{H}}\}_{j=N+1,\ldots,N+M}$, independent of the $N = 1000$ simulations on which the regression coefficients were estimated. The (relative) AIC value in the case of linear regression is:

$$AIC(\alpha) = M \log\left(\frac{RSS(\alpha)}{M}\right) + 2\left(1 + \sum_{i\in\mathcal{H}} 1_{\hat{\phi}_0^i(\alpha)\neq 0}\right), \tag{4.24}$$

where $1 + \sum_{i\in\mathcal{H}} 1_{\hat{\phi}_0^i(\alpha)\neq 0}$ is the number of parameters of the regression fit ($1_{\hat{\phi}_0^i(\alpha)\neq 0}$ is zero if $\hat{\phi}_0^i(\alpha) = 0$ and one otherwise), and the residual sum of squares $RSS(\alpha)$ is estimated on the new M samples from VolGAN:

$$RSS(\alpha) = \sum_{j=N+1}^{N+M} \left(\Delta V_t(\omega_j) - \hat{A}_0(\alpha) - \sum_{i\in\mathcal{H}} \hat{\phi}_0^i(\alpha)\Delta H_t^i(\omega_j)\right)^2.$$

That is, at each starting time $t = 0$ we choose α by minimising an AIC criterion, where the set of possible α is $\mathcal{A} = \{0.01, 0.02, \ldots, 0.2\}$. Estimating the AIC on independent VolGAN simulations prevents overfitting and provides more accurate estimates of regression fits. Since only VolGAN simulations are used at time $t = 0$, the choice of α is not anticipative, i.e. there is no look-ahead bias. Figure 4.3 shows that during periods of market turbulence, higher regularisation is preferred. The most common choice is $\alpha = 0.01$. Furthermore, we note that the values of m_0 closer to one result in more frequent recalibration of the regularisation parameter α. The fact that the highest value of the regularisation parameter is selected during periods of market turbulence suggests that a higher value of α would be more suitable than in those instances. However, a reduced search grid for α of $\{0.01, 0.05, 0.1, 0.5, 1\}$ results in a similar performance, as demonstrated in Section 4.4. We also note that fixing $\alpha = 0.25$ does not significantly alter the performance either.

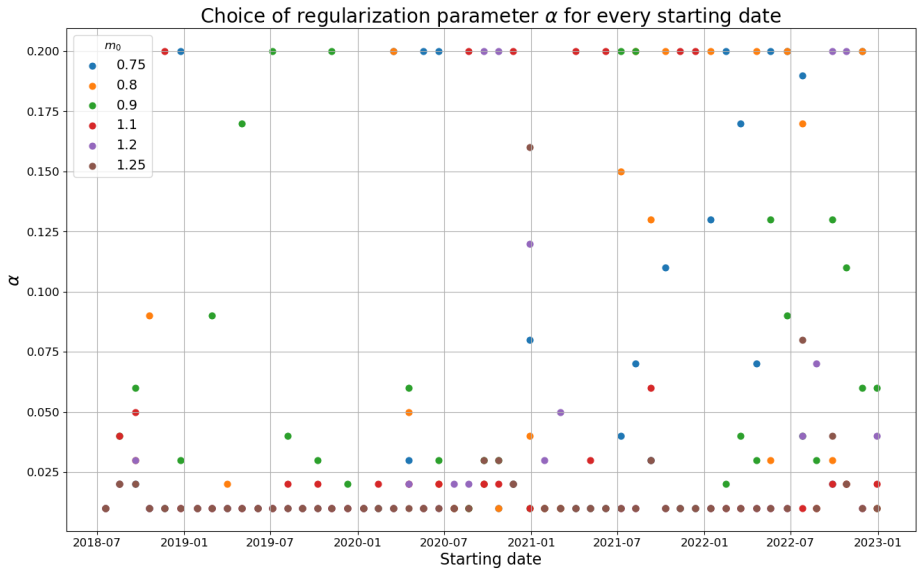


Figure 4.3: Choice of regularisation parameter α minimising the AIC (4.24) for each starting date $t = 0$ and for different values of m_0.

The value of V_t is the same as the gross value of the position since V_t is made up of a long call and a long put. Figure 4.4 shows that V_t is higher for values of m_0 further away from one. That is, the same value of α would result in a lower value of the regularisation term αg_0 in (4.20) for $m_0 \in \{0.9, 1.1\}$ compared to $m_0 \in \{0.75, 0.8, 1.2, 1.25\}$.

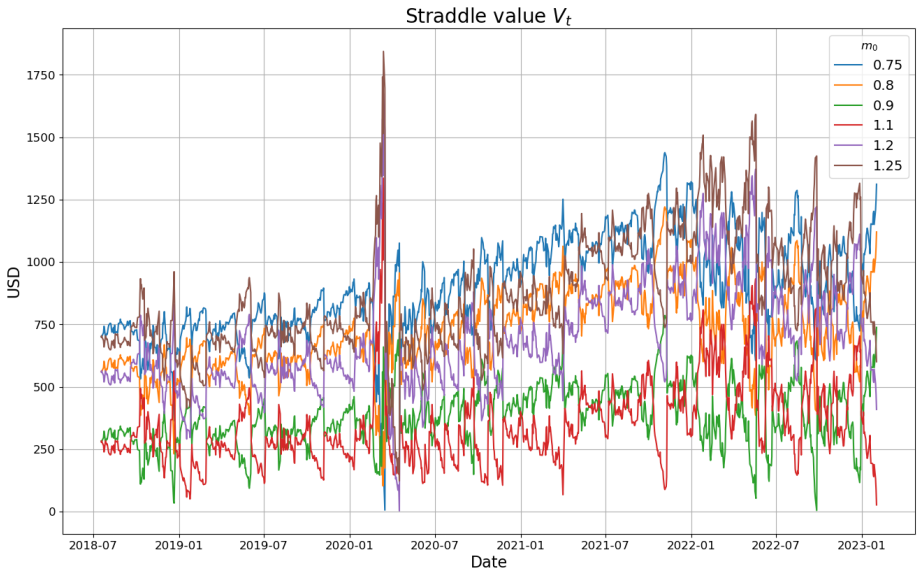


Figure 4.4: Value V_t of the long straddle position for different values of m_0.

4.3.3 Number of hedging instruments

Since incorporating transaction cost via an ℓ_1-penalty (4.20) leads to sparsity, we investigate the number of hedging instruments selected throughout the testing period. Unsurprisingly, the underlying is always selected. A breakdown of the number of times that each possible hedging instrument count was used for each starting value m_0 is available in Table 4.1. Data-driven hedging with VolGAN predominantly selected only the underlying as the hedging instruments for the straddles with m_0 further away from one. When $m_0 \in \{0.9, 1.1\}$ (the values closest to 1), options were selected alongside the underlying majority of the time.

	Number of instruments							
m_0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.75	1058	3	11	20	0	0	0	0
0.80	788	156	46	88	3	11	0	0
0.90	193	279	168	277	93	67	15	0
1.10	565	40	72	110	219	69	17	0
1.20	1077	2	7	0	0	1	3	2
1.25	1087	0	0	2	3	0	0	0

Table 4.1: Frequency of the number of hedging instruments selected for different m_0 values. The total number of days is $21 \times 52 = 1092$, with an additional day for unwinding. Position expiry dates are not included since the only adjustments in the hedging portfolio happen on days $0, \ldots, 20$.

Figure 4.5 shows the number of hedging instruments used for different starting values m_0. We note a spike in the number of selected hedging instruments at the start of 2019, during the start of the Covid-19 pandemic, and in the second half of 2022.

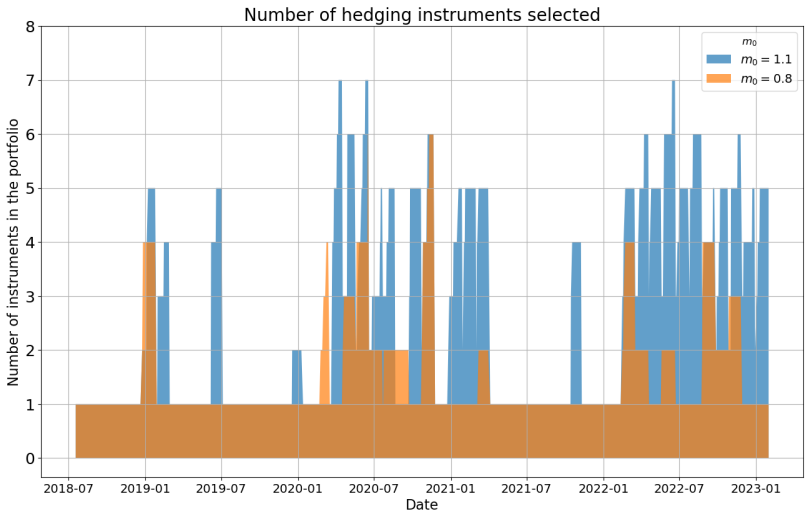


Figure 4.5: Number of hedging instruments selected for different values of m_0.

4.3.4 Tracking error statistics

We first compare the performance during the entire test set, and then without the initial Covid-19 shock. To exclude the Covid-19 pandemic effect, we remove 5 one-month periods starting from the position entered on the 13th Feb 2020. This results in exclusion of the dates 13th Feb 2020-21st Jul 2020.

Tracking error Z_t statistics under consideration are the mean, median, standard deviation, and Value-at-Risk, defined as

$$VaR_q(Z_t) = -F_{Z_t}^{-1}(q),$$

for $q = 5\%, 2.5\%$, and 1%, where $F_{Z_t}^{-1}$ is the quantile function of Z_t.

The tracking error statistics (for all values of m_0 pooled together) for delta hedging, delta-vega hedging, and VolGAN are given in Table 4.2. Outside of the Covid-19 pandemic, VolGAN results in the lowest standard deviation and 1% VaR. The corresponding tracking error distributions over the entire test set are shown in Figure 4.6. Usually, VolGAN is between delta-hedging and delta-vega hedging. We also note a significant reduction in standard deviation, and value-at-risk metrics compared to the unhedged position.

Covid-19 period	Method	Statistics					
		Mean	Median	Std	5% VaR	2.5% VaR	1% VaR
Included	Unhedged	0.16	0.29	60.58	78.95	102.07	155.29
	Delta	-1.23	-0.44	32.70	19.49	36.33	58.63
	Delta-vega	0.98	0.00	29.70	10.90	19.70	43.72
	Data-driven	0.55	-0.16	32.98	12.79	23.42	50.79
Excluded	Delta	-1.46	-0.36	8.40	13.22	22.84	36.66
	Delta-vega	-0.37	0.00	9.34	9.58	16.67	34.81
	Data-driven	-1.05	-0.18	8.15	10.55	17.32	33.85

Table 4.2: Tracking error statistics in USD for all values of m_0 over the entire test period.

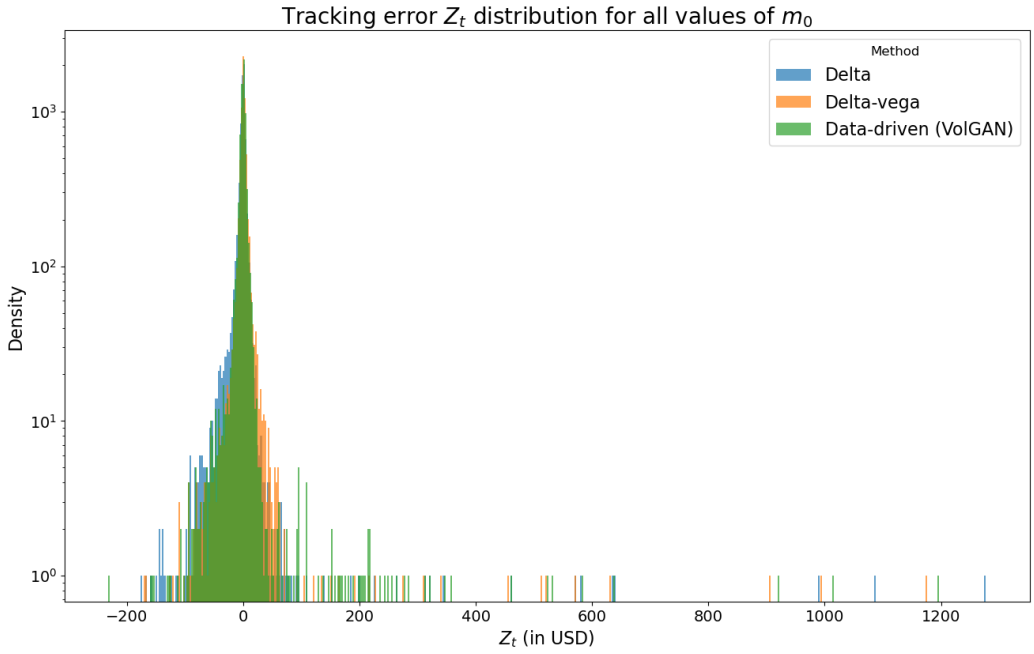


Figure 4.6: Distribution of tracking error Z_t for all values of m_0 pooled together, over the entire test set.

Comparing VolGAN hedge with delta hedge in Figure 4.7, and with delta-vega hedge in Figure 4.8, with the Covid-19 pandemic data excluded for better visualisation, we observe that all methods result in similar performance in the bulk of the experiments. In most cases where delta hedging results in a positive PnL, the same holds for VolGAN, and the points in the first quadrant appear to scatter symmetrically around $y = x$. However, there are many instances in which VolGAN results in a positive PnL, while delta hedging does not. In the scenarios in which both

approaches result in a negative PnL, the loss is usually more severe for delta-hedging. There is a bit more symmetry when comparing VolGAN with delta-vega hedging in Figure 4.8. This observation aligns with the statistics in Table 4.2, where outside of the Covid-19 pandemic, data-driven hedging leads to tracking errors statistics closer to delta-vega hedging than to delta hedging. As m_0 moves away from 1, differences in hedging approaches are less prominent: the points are tightly concentrated around $y = x$.

A detailed breakdown of performance for each individual value of m_0 is available in the Section 4.4.

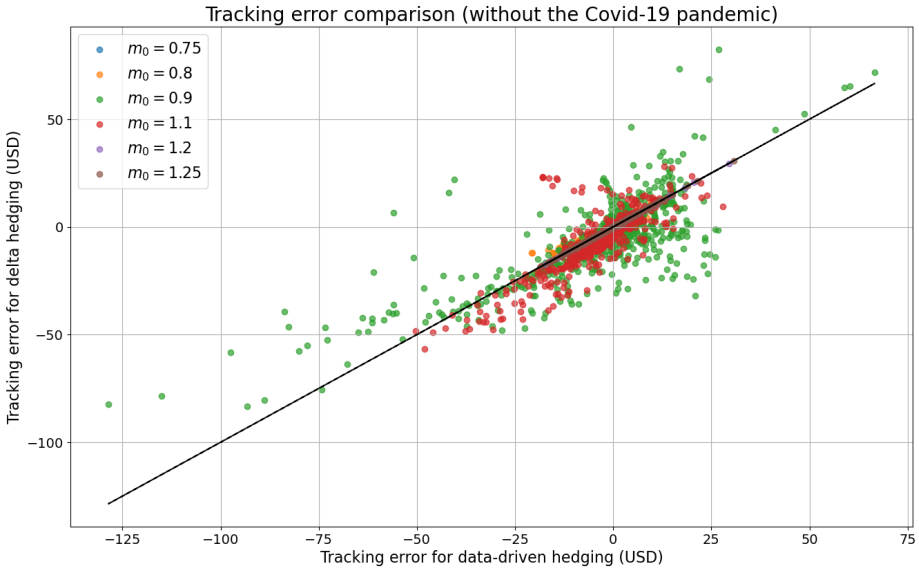


Figure 4.7: Tracking error: delta hedge vs VolGAN hedge. Solid black line: $y = x$. Covid-19 data excluded.

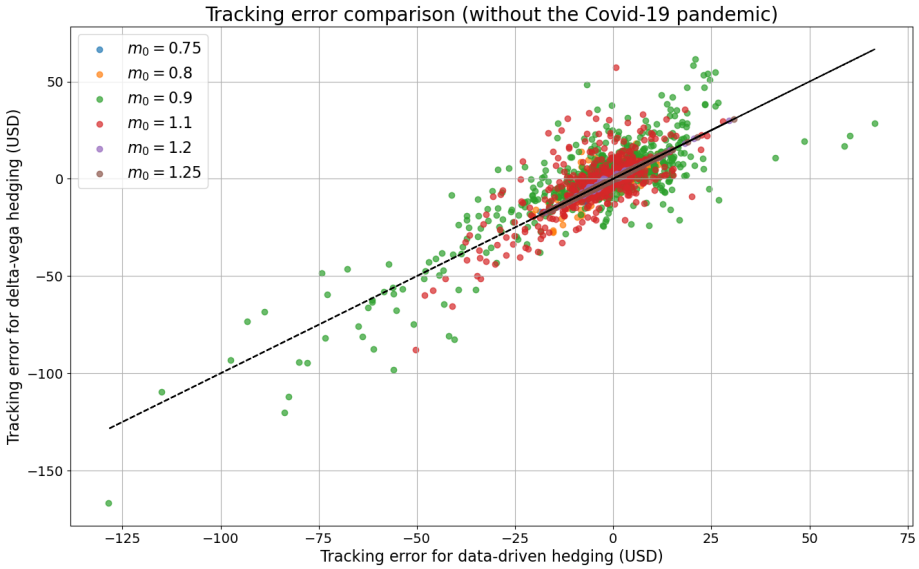


Figure 4.8: Tracking error: delta-vega hedge vs VolGAN hedge. Solid black line: $y = x$. Covid-19 data excluded.

Figures 4.9 and 4.10 compare the performance of the data-driven hedging strategy with delta hedging and delta-vega hedging, with the Covid-19 period included.

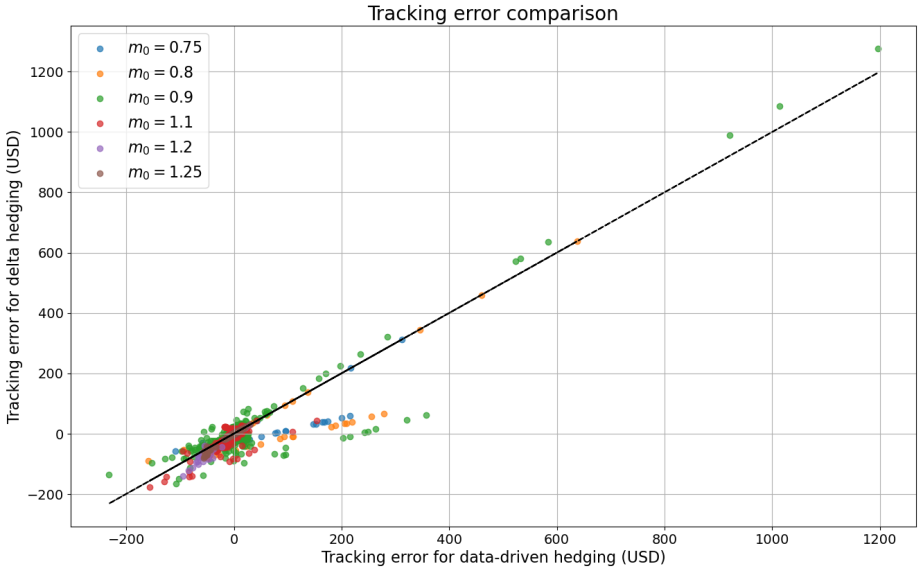


Figure 4.9: Tracking error: delta hedge vs VolGAN hedge. Solid black line: $y = x$.

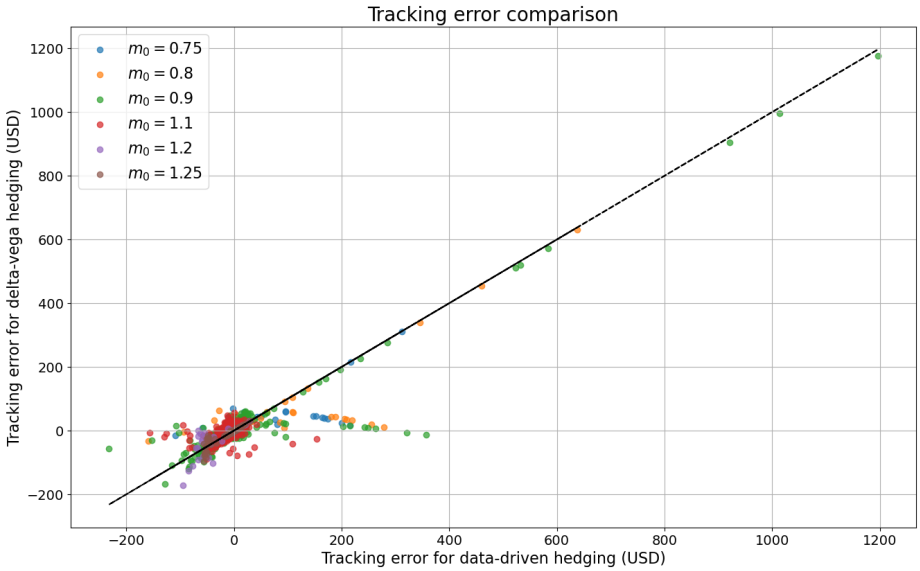


Figure 4.10: Tracking error: delta-vega hedge vs VolGAN hedge. Solid black line: $y = x$.

4.3.5 Delta and vega of the hedged position

In order to better understand the nature of the hedging strategy resulting from our method (4.20), we compare the vega and delta of the straddle V_t with the total vega and delta of the hedged position Z_t. As shown in Table 4.3, the delta exposure of the position is almost completely hedged by data-driven hedging. On the other hand, Table 4.4 shows that although the total vega of the hedged position Z_t is reduced compared to that of the unhedged position V_t, it is not zero. These observations show that our data-driven hedging approach (4.20) is not equivalent to delta-vega hedging. So, unlike what has been suggested in the recent literature [96], there is more to data-driven hedging than just "learning the Greeks".

	Mean		Median		95% Quantile		5% Quantile		Std Dev	
m_0	Z_t	V_t	Z_t	V_t	Z_t	V_t	Z_t	V_t	Z_t	V_t
0.75	0.003	0.994	0.000	1.000	0.005	1.000	-0.001	0.986	0.023	0.067
0.8	0.003	0.986	0.000	1.000	0.009	1.000	-0.016	0.958	0.040	0.097
0.9	-0.011	0.905	-0.011	0.979	0.051	1.000	-0.111	0.618	0.100	0.214
1.1	-0.006	-0.940	0.000	-0.997	0.035	-0.760	-0.049	-1.000	0.066	0.174
1.2	-0.000	-0.996	-0.000	-1.000	0.000	-0.998	-0.001	-1.000	0.016	0.039
1.25	0.000	-0.999	-0.000	-1.000	0.000	-1.000	-0.001	-1.000	0.006	0.009

Table 4.3: Delta statistics.

	Mean		Median		95% Quantile		Std Dev	
m_0	Z_t	V_t	Z_t	V_t	Z_t	V_t	Z_t	V_t
0.75	5.828	6.889	0.078	0.102	32.904	35.288	21.269	22.073
0.8	9.546	17.136	0.170	0.805	66.971	91.581	36.314	40.845
0.9	0.096	111.500	-4.397	37.221	179.155	452.507	89.315	150.315
1.1	26.165	66.529	2.807	7.817	170.733	326.120	68.777	108.969
1.2	3.302	3.637	0.000	0.000	5.399	5.637	23.437	25.145
1.25	1.076	1.079	0.000	0.000	0.239	0.239	10.013	10.022

Table 4.4: Vega statistics.

4.4 Hedging with VolGAN: robustness checks

To assess the stability and generalisability of our data-driven hedging methodology, we conduct a series of robustness checks examining the sensitivity of results to different values of m_0 and regularisation parameter α.

4.4.1 Performance for different values of m_0

Table 4.5 contains tracking error statistics over the entire data set, for each value of m_0. Most of the time, for $m_0 < 1$, data-driven hedging results in all tracking error statistics of interest between those corresponding to delta hedging and delta-vega hedging. The higher standard deviation in $m_0 = 0.75$ is due to the options selected as hedging instruments at the beginning of the Covid-19 pandemic, which, as illustrated in Figure 4.11, results in a more prominent volatility of the tracking error. As more options are selected for hedging, the performance of data-driven hedging is closer to that of delta-vega hedging than to pure delta hedging.

When $m_0 = 1.1$, hedging with options brings about a very clear reduction in all risk measures being considered. In this instance, delta hedging has the worst performance, delta-vega hedging has the lowest tracking error variance, but data-driven hedging results in the lowest Value-at-Risk. For $m_0 \in \{1.2, 1.25\}$, data-driven hedging produces tracking error of the lowest variance, with Value-at-Risk statistics that are usually in between those corresponding to the Black-Scholes benchmarks.

m_0	Method	Statistics					
		Mean	Median	Std	5% VaR	2.5% VaR	1% VaR
0.75	Delta hedging	0.17	-0.06	2.81	6.10	7.32	12.37
	Delta-vega hedging	1.38	0.11	12.89	5.07	6.54	9.09
	Data-driven hedging	1.48	0.00	19.23	5.66	7.01	9.04
0.80	Delta hedging	0.39	-0.32	27.20	8.69	12.23	18.39
	Delta-vega hedging	2.43	0.23	26.65	6.31	8.82	17.05
	Data-driven hedging	2.49	-0.16	32.66	7.78	10.31	15.30
0.90	Delta hedging	-0.35	-1.95	70.56	40.24	53.71	81.30
	Delta-vega hedging	4.90	0.55	63.76	26.88	56.35	81.00
	Data-driven hedging	4.02	-0.32	68.66	34.48	55.92	80.78
1.10	Delta hedging	-5.17	-0.98	17.00	33.66	46.66	76.69
	Delta-vega hedging	-1.20	-0.13	11.88	19.59	31.40	51.36
	Data-driven hedging	-2.82	-0.74	13.79	19.17	31.41	47.58
1.20	Delta hedging	-1.41	0.00	11.90	8.59	13.36	73.09
	Delta-vega hedging	-0.87	0.00	10.49	8.07	12.79	31.22
	Data-driven hedging	-1.04	-0.01	9.26	8.45	13.39	56.66
1.25	Delta hedging	-1.02	-0.001	9.33	8.61	13.88	54.51
	Delta-vega hedging	-0.79	0.00	8.47	8.42	13.88	37.12
	Data-driven hedging	-0.82	-0.01	8.04	8.60	13.87	48.52

Table 4.5: Performance metrics for different m_0 values over the entire test set (rounded to two decimal places). The initial positions $Z_0 = 0$ are included in the statistics.

The tracking errors Z_t that come from different hedging approaches as functions of time are shown in Figure 4.11, for various values of m_0. For better visualisation, Figure 4.12 displays the tracking values on a scale that is linear in the interval $[-50, 50]$ and logarithmic away from it. All three methods track the initial portfolio satisfactorily well, aside from the Covid-19 pandemic. The three methods are difficult to differentiate from each other during periods of calm. Table 4.6 highlights that the absolute value of the tracking error is usually less than 1% of the portfolio value, that is, the tracking portfolio and the portfolio we wish to hedge are usually within 1% of each other. Furthermore, Table 4.6 indicates that all methods perform worse for values of m_0 closer to one. The most volatility is visible during March-June 2020, and during 2022. The corresponding tracking error distributions are available in Figure 4.13. The bulk of the tracking error distribution is usually the tightest for data-driven hedging.

Method/m_0	0.75	0.80	0.90	1.10	1.20	1.25
Delta-hedging	95.98%	88.11%	43.62%	50.61%	84.53%	89.34%
Delta-vega hedging	94.49%	88.02%	49.04%	51.57%	85.05%	89.42%
VolGAN-based hedging	95.37%	89.25%	51.92%	54.63%	84.53%	89.34%

Table 4.6: Proportion of observations on which the tracking portfolio Π_t was within 1% of the initial straddle portfolio value V_t for different methods and m_0 values.

m_0	Method	Statistics					
		Mean	Median	Std	5% VaR	2.5% VaR	1% VaR
0.75	Delta hedging	-0.23	0.00	2.81	4.99	6.27	7.15
	Delta-vega hedging	0.09	0.04	2.92	4.89	6.47	8.57
	Data-driven hedging	-0.29	0.00	2.87	5.24	6.28	7.56
0.80	Delta hedging	-0.73	-0.17	3.45	6.60	8.01	11.09
	Delta-vega hedging	0.002	0.11	4.34	6.19	8.58	17.35
	Data-driven hedging	-0.71	-0.19	3.83	7.01	9.13	13.22
0.90	Delta hedging	-4.35	-1.77	15.58	34.46	42.10	51.00
	Delta-vega hedging	-0.88	0.40	18.52	27.61	56.87	81.74
	Data-driven hedging	-2.87	-0.39	16.30	32.88	51.28	73.20
1.10	Delta hedging	-3.15	-0.82	10.09	24.34	32.93	43.12
	Delta-vega hedging	-1.19	-0.17	10.47	17.59	27.81	40.52
	Data-driven hedging	-2.13	-0.72	7.84	16.40	24.08	33.99
1.20	Delta hedging	-0.17	-0.02	4.49	7.17	9.48	12.79
	Delta-vega hedging	-0.12	-0.003	4.50	7.22	9.38	12.80
	Data-driven hedging	-0.17	-0.07	4.49	7.14	9.46	12.78
1.25	Delta hedging	-0.14	-0.06	4.67	7.42	9.77	13.34
	Delta-vega hedging	-0.14	-0.05	4.67	7.42	9.78	13.36
	Data-driven hedging	-0.14	-0.07	4.67	7.41	9.78	13.32

Table 4.7: Performance metrics for different m_0 values without the start of the Covid-19 pandemic (rounded to two decimal places). The initial positions $Z_0 = 0$ are included in the statistics.

Given that we considered all 52 one-month intervals jointly, and that the start of the Covid-19 pandemic resulted in high tracking error variance, as well as that the tracking error was mainly positive during this period (as evidenced in Figure 4.11), it is important to repeat the analysis with 13th Feb 2020-21st Jul 2020 excluded. Once again, when $m_0 < 0$ data-driven hedging results in tracking error statistics in between of that corresponding to the Black-Scholes benchmark. As fewer options

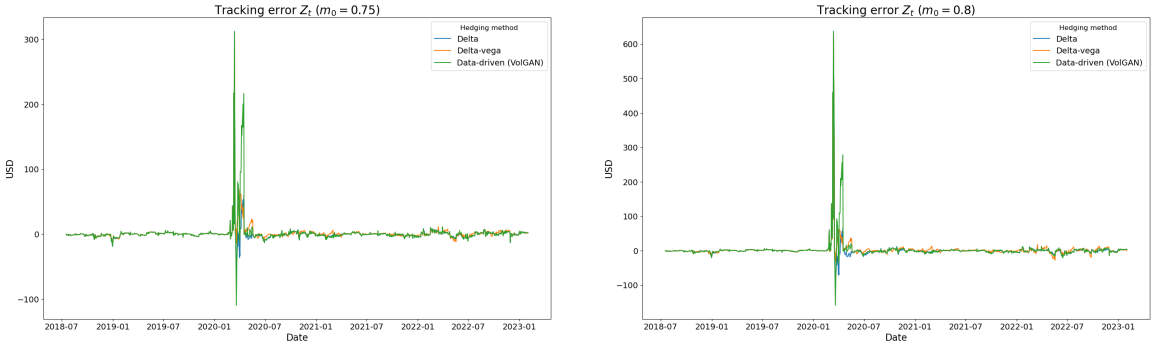
are selected for hedging, data-driven hedging results in performance more similar to that of delta hedging. All three methods obtain very similar performance to each other for values of $m_0 \in \{0.75, 1.2, 1.25\}$, which have the lowest total vega exposure (Table 4.4). Interestingly, all three examples result in the underlying being selected by data-driven hedging during the analysis period (Figure 4.5). The most significant difference is visible in the $m_0 = 1.1$ example, when data-driven hedging outperforms the benchmarks in both variance and Value-at-Risk statistics. Outside of the Covid-19 pandemic, data-driven hedging always results in lower variance and 1% Value-at-Risk than delta-vega hedging, even when more options are selected for hedging, resulting in higher rebalancing costs.

Figure 4.11 indicates that the majority of the tracking error variance after 2020 is due to the behaviour in 2022. During this time, data-driven hedging with VolGAN results in much more stable hedging performance compared to the Black-Scholes benchmarks.

4.4.2 Robustness with respect to regularisation parameter

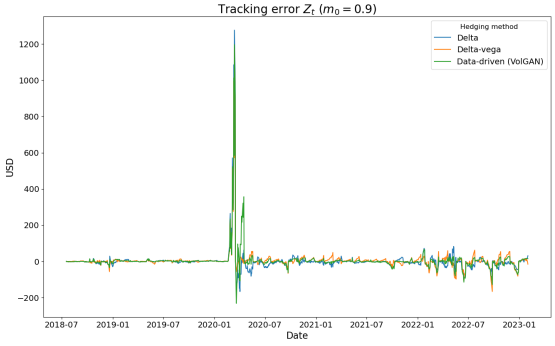
We investigate how the results of the data-driven hedging change with the change of the *search grid* $\mathcal{A}$ for the regularisation parameter α. Furthermore, we compare the performance for fixed α on three different scales. From Figure 4.14, we conclude that for $\alpha = 0.001$ overfitting occurs, whereas $\alpha = 5$ results in underfitting. However, $\alpha = 0.25$ leads to a similar tracking error to that corresponding to a dynamic choice of α via the AIC criterion.

Table 4.8 contains the information from Table 4.2, in addition to the results for data-driven hedging with potential values of the regularisation parameter in $\mathcal{A}_1 = \{0.01, 0.05, 0.1, 0.5, 1\}$. The tracking error statistics are very similar for both the search grids $\mathcal{A}_1$ and $\mathcal{A}_2 = \{0.01, 0.02, \ldots, 0.2\}$, indicating robustness to the regularisation parameter. However, it is important to note that much lower or higher values of α may result in overfitting and underfitting, as demonstrated with $\alpha = 0.001$ and $\alpha = 5$.



(a) $m_0 = 0.75$

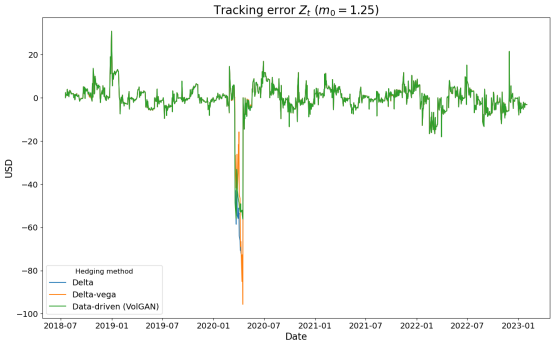
(b) $m_0 = 0.80$



(c) $m_0 = 0.90$

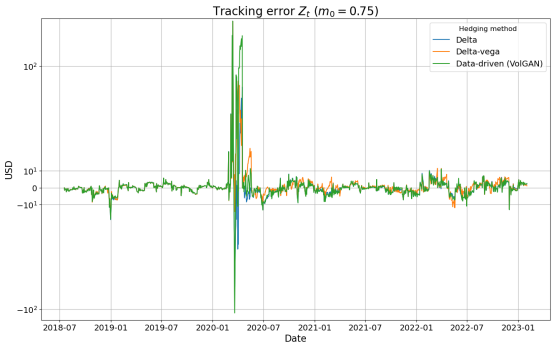
(d) $m_0 = 1.10$

(e) $m_0 = 1.20$

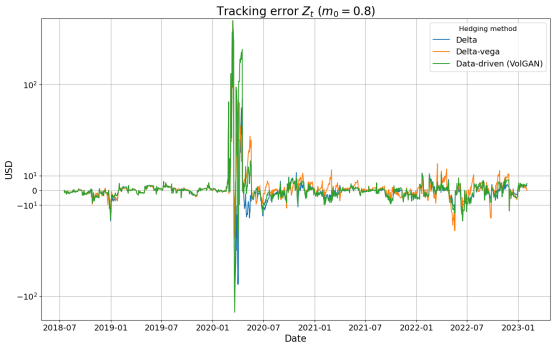


(f) $m_0 = 1.25$

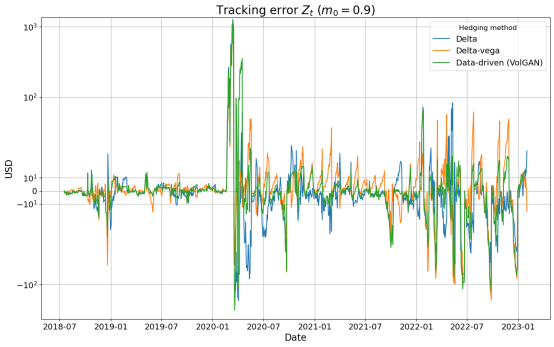
Figure 4.11: Tracking error Z_t as a function of time for different values of m_0. Once an initial one-month straddle is expired, a new one is entered.



(a) $m_0 = 0.75$

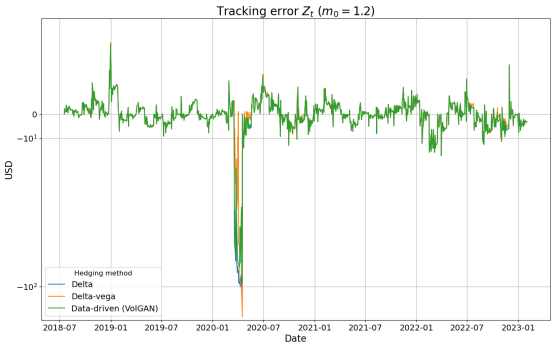


(b) $m_0 = 0.80$



(c) $m_0 = 0.90$

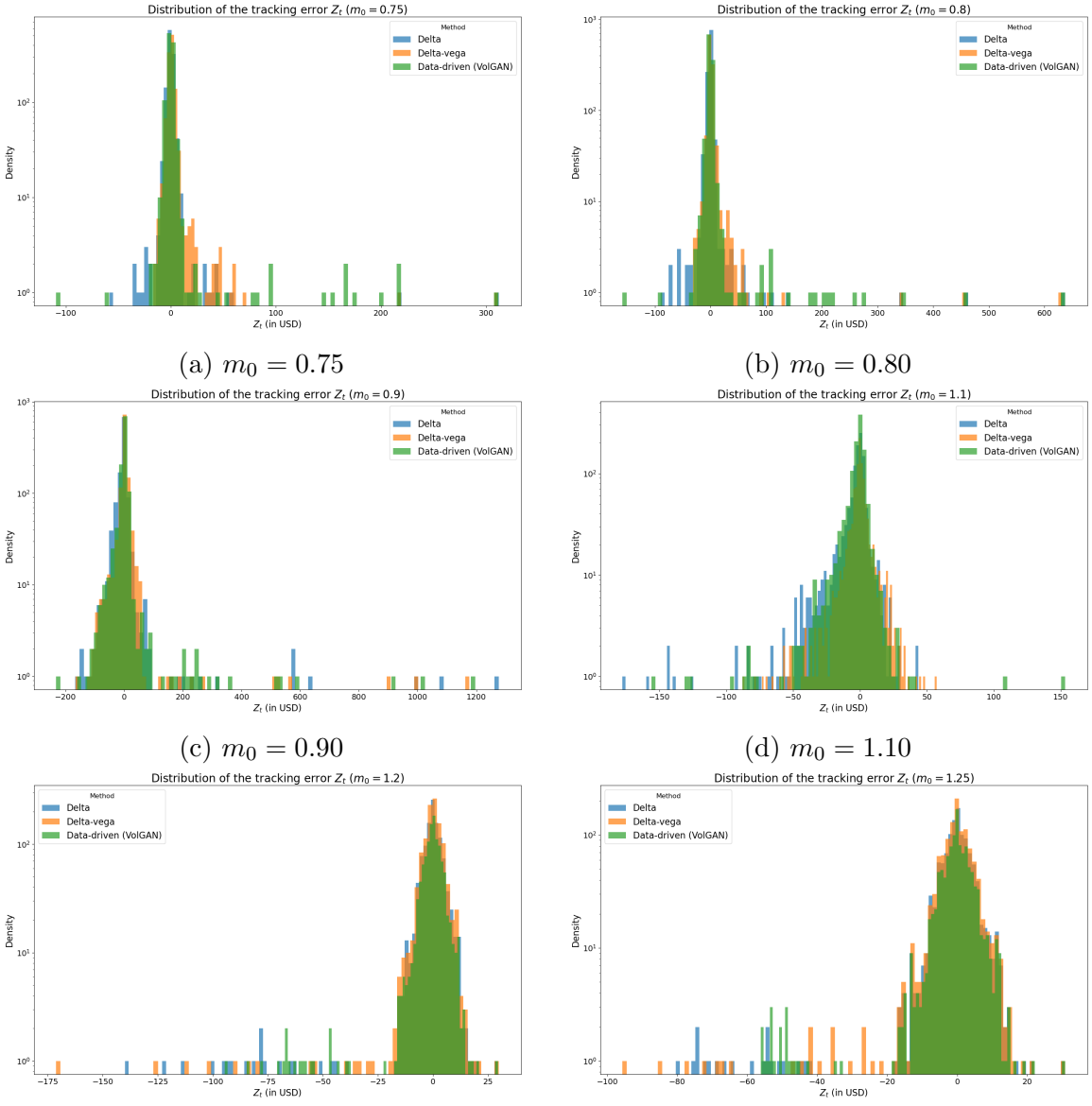
(d) $m_0 = 1.10$



(e) $m_0 = 1.20$

(f) $m_0 = 1.25$

Figure 4.12: Tracking error Z_t as a function of time for different values of m_0 on a scale which is linear in $[-50, 50]$, and logarithmic away from this interval. Once an initial one-month straddle is expired, a new one is entered.



(a) $m_0 = 0.75$

(b) $m_0 = 0.80$

(c) $m_0 = 0.90$

(d) $m_0 = 1.10$

(e) $m_0 = 1.20$

(f) $m_0 = 1.25$

Figure 4.13: Histogram of the tracking error Z_t for different values of m_0. Once an initial one-month straddle is expired, a new one is entered.

Covid-19	Method	Statistics					
		Mean	Median	Std	5% VaR	2.5% VaR	1% VaR
Included	Delta	-1.23	-0.44	32.70	19.49	36.33	58.63
	Delta-vega	0.98	0.00	29.70	10.90	19.70	43.72
	Data-driven ($\mathcal{A}_1$)	0.58	-0.13	33.01	13.01	24.44	50.45
	Data-driven ($\mathcal{A}_2$)	0.55	-0.16	32.98	12.79	23.42	50.79
Excluded	Delta	-1.46	-0.36	8.40	13.22	22.84	36.66
	Delta-vega	-0.37	0.00	9.34	9.58	16.67	34.81
	Data-driven ($\mathcal{A}_1$)	-1.02	-0.18	8.16	10.80	18.08	35.35
	Data-driven ($\mathcal{A}_2$)	-1.05	-0.18	8.15	10.55	17.32	33.85

Table 4.8: Performance metrics for all values of m_0 over the entire test set (rounded to two decimal places). The initial positions $Z_0 = 0$ are included in the statistics. Grid searches for α: $\mathcal{A}_1 = \{0.01, 0.05, 0.1, 0.5, 1\}$ and $\mathcal{A}_2 = \{0.01, 0.02, \ldots, 0.2\}$.

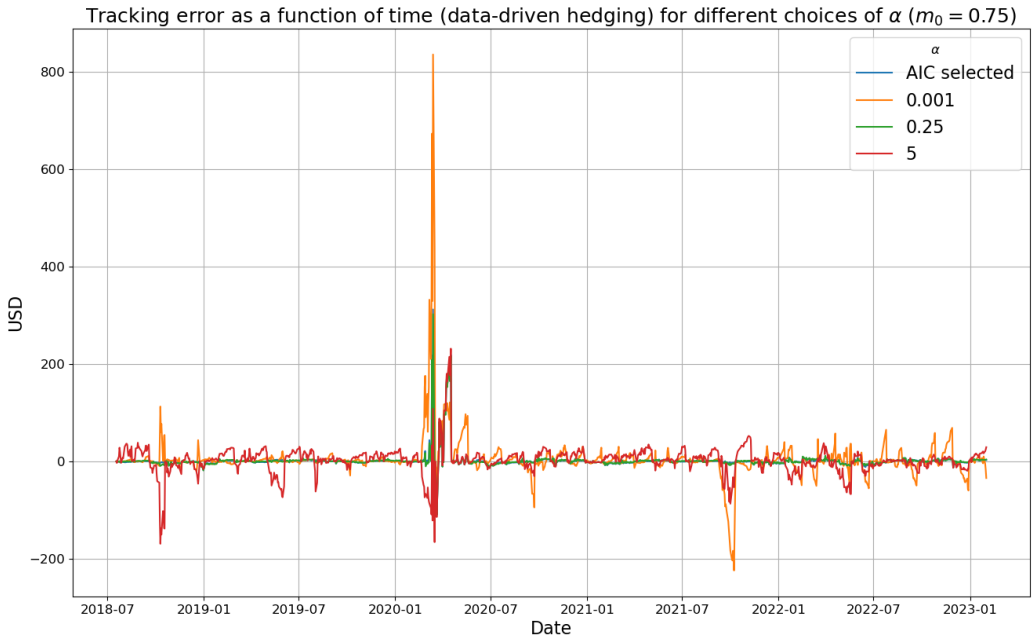


Figure 4.14: Tracking error as a function of time, $m_0 = 0.75$. Comparison between different values of α. Opting for very low or very high values of α results in overfitting and underfitting, respectively. Opting for fixed $\alpha = 0.25$ results in a similar performance to performing a grid search over $\mathcal{A}_1$ or $\mathcal{A}_2$. The "AIC selected" refers to searching over $\mathcal{A}_2 = \{0.01, 0.02, \ldots, 0.2\}$.

Chapter 5

FinGAN: forecasting and classifying financial time series via generative adversarial networks

5.1 Introduction

Following the use of generative models for risk management and hedging in Chapter 4, we now explore their potential in forecasting asset returns. While the previous chapter focused on conditional simulation to support hedging decisions, our objective in this chapter is to construct trading strategies using the forward-looking distributions produced by generative models.

First, we fix a frequency Δt, and focus on a single particular asset. Denote by $x_{t+\Delta t}$ its return between times t and $t + \Delta t$. Although X_t denoted non-tradable risk factors in Chapters 1-4, we now repurpose the notation x_t to refer to the asset return between t and $t + \Delta t$. This change provides a unified notation across different return types considered in this and the following chapter, including log-returns, excess log-returns, and market residuals.

In line with the generative modelling framework of Chapter 1, we investigate how a generative model G, particularly a GAN, conditioned on past returns $a_t = (x_t, x_{t-\Delta t}, \dots)$ and latent noise $Z \sim \mathbb{P}_Z$, can be applied in a trading context.

This chapter is based on [136], and presents Fin-GAN, a conditional generative adversarial network adapted to the forecasting setting. We place conditional GANs into a supervised learning setting by introducing a novel economics-based loss function for the generator. Fin-GAN outperforms a suite of benchmarks on daily stock market data in terms of Sharpe Ratios achieved, while producing distributional forecasts and uncertainty estimates. Furthermore, Fin-GAN alleviates issues around mode collapse.

Although machine learning (and, more specifically, deep learning) has become increasingly adopted in financial settings [1, 102] and in time series forecasting [24], training is typically done via a mean squared error objective, in a supervised learning manner, without adversarial training. The most similar loss to ours is perhaps QuantNet [86], who also opt for an objective based on trading strategy performance. However, the approach of [86] offers only pointwise estimates.

Outline. Section 5.2 introduces performance metrics, motivating the design of the custom loss. Section 5.3 defines the Fin-GAN loss and provides model details. Section 5.4 describes the data and implementation considerations. Section 5.5 introduces benchmark models. Finally, Section 5.6 is an empirical study on equity returns. Apart from a single-asset setting, Section 5.6 also studies Fin-GAN's performance in a universality [125] setting.

Code availability. The code used to implement Fin-GAN is publicly available at `https://github.com/milenavuletic/Fin-GAN`.

5.2 Performance measures

There are a number of ways to measure model performance. Based on applications, not all performance metrics are equally important. In order to compare one approach to another, it is crucial to understand which properties of the data we care about the most, in order to be able to construct appropriate performance metrics. Unlike in Chapter 3 (for VolGAN), we are not interested in analysing the statistical properties of the simulated samples, but specifically in the trading performance of Fin-GAN.

Opting for the anti-symmetric version of the sign function,

$$\text{sign}(x) = \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x < 0, \end{cases} \tag{5.1}$$

a trade at time t based on the expected sign of the forecast $\hat{x}_{t+\Delta t}$ will result in a PnL of

$$PnL_{t+\Delta t} = \mathbb{E}\left[\text{sign}\left(\hat{x}_{t+\Delta t}\right)|\mathcal{F}_t\right] x_{t+\Delta t.} \tag{5.2}$$

Of course, in the case of pointwise forecasts, $\mathbb{E}\left[\text{sign}\left(\hat{x}_{t+\Delta t}\right)|\mathcal{F}_t\right] = \text{sign}\left(\hat{x}_{t+\Delta t}\right)$. However, in case of a generative model G, the expected sign is given by $\mathbb{E}\left[\text{sign}\left(\hat{x}_{t+\Delta t}\right)|\mathcal{F}_t\right] = \mathbb{E}\left[\text{sign}\left(G(a_t, Z)\right)\right]$, with $Z \sim \mathbb{P}_Z$.

Traditionally, when the task at hand is time series forecasting, one opts for performance metrics such as MSE (mean squared error), RMSE(root mean squared error), and MAE (mean absolute error). To define MAE and RMSE, suppose that the real value we wish to forecast is $\boldsymbol{x} = (x_{t_1}, \ldots, x_{t_n})$ and that our forecast is $\hat{\boldsymbol{x}} = (\hat{x}_{t_1}, \ldots, \hat{x}_{t_n})$. This can be one generative model output given a single noise sample, or the mean or the mode of the generated distribution given the corresponding condition. Then, the mean absolute error and the root mean squared error are defined as

$$MAE(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{n}\sum_{i=1}^{n} |x_{t_i} - \hat{x}_{t_i}|, \qquad RMSE(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{t_i} - \hat{x}_{t_i})^2}. \quad (5.3)$$

As we wish to use the outputs of the generative model under consideration in a trading context, MAE and RMSE may not be as they are in traditional time-series settings. Hence, a metric based on profit and loss (PnL) is more suitable. While both MAE and RMSE can be very low, it could still be possible to misclassify the side/directionality of a large move in the underlying asset price, leading to a severe PnL loss. Hence, we incorporate the PnL as one of our main performance metrics.

At each point in time, we produce forecasts for $x_{t+\Delta t}$, and place trades based on these forecasts. This sequence of trades results in a time series of PnLs. To minimise risk and maximise profit, one aims for the PnL time series to be positive, and have low variance. The larger the average of the PnL series is, the higher the profitability of our strategy. The lower the variance of the PnL series, the lower the risk associated with the strategy. Altogether, this leads us to employ the *annualised Sharpe Ratio* [124] as our main performance measure. We remark that this is also the metric commonly used by portfolio managers to assess the risk-adjusted performance of their strategies.

In order to estimate the expected sign of the return forecast $\hat{x}_{t+\Delta t}$, we take $B = 1000$ i.i.d noise samples $\{z_j\}_{j\in\{1,\ldots,B\}}$ from $\mathbb{P}_Z$, for the same condition a_t. The estimates of probabilities of upwards and downwards moves are denoted by p_u and p_d, and are more precisely defined as

$$p_u^t = \frac{1}{B}\sum_{j=1}^{B} \mathbb{1}_{G(a_t,z_j)>0}, \;\; p_d^t = \frac{1}{B}\sum_{j=1}^{B} \mathbb{1}_{G(a_t,z_j)<0}. \quad (5.4)$$

Then, the PnL resulting from the generator, taking into account our certainty of the sign of the forecast, expressed in basis points is

$$wPnL_B^t = 10000(p_u^t - p_d^t)x_{t+\Delta t}. \quad (5.5)$$

This is the PnL of an equivalent strategy, consisting of taking an average bet of each of the generator's outcomes given appropriate conditions across different noise samples. Hence, we are penalising higher uncertainty more, and if we are not very certain about the outcome, we attain significantly lower potential loss. We refer to such an approach as the *weighted strategy*.

Since Sharpe Ratio is typically calculated via daily PnLs, we sum together the individual PnLs over one day to reach daily PnLs. Suppose that there are trades at times $t \in \mathbb{T}_i$ on a particular day i. Then, the resulting PnL for day i is

$$wPnL^{\mathbb{T}_i} = \sum_{t \in \mathbb{T}_i} wPnL_B^t. \tag{5.6}$$

The more symmetric distributions we learn, the closer $wPnL$ is to zero. We use the weighted strategy introduced above to compute the annualised Sharpe Ratio, which we use as our main performance metric. Given days $i \in \mathcal{D}$, we estimate the Sharpe Ratio SR as

$$PnL = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} wPnL^{\mathbb{T}_i}, \quad SR = SR = \sqrt{252} \frac{PnL}{\left(\frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} (wPnL^{\mathbb{T}_i} - PnL)^2 \right)^{1/2}}. \tag{5.7}$$

Annualised Sharpe Ratios of above 1 are considered to be *good*, above 2 are considered to be *very good* and those above 3 are referred to as *excellent*.

Remark 8. *Note that we do not take transaction costs into account.*

5.3 Fin-GAN loss function

The main contribution of our work lies in introducing a novel economics-driven loss function for the generator, which places GANs into a **supervised learning** setting. This approach allows us to move beyond traditionally employed methods for pointwise forecasting and move towards probabilistic forecasts, which offer uncertainty estimates. Furthermore, we are evaluating the performance based on the sign of the predictions, meaning that we are interested in **classification**, which our loss function is more suitable for.

5.3.1 Motivation

We are interested in correctly classifying returns/excess returns, especially when the magnitude of the price moves is large, rather than in producing forecasts close to

the realised value. The motivation is that one may produce a forecast which is close to the realised value of the time series, but has the opposite sign. In financial time series forecasting, *correctly estimating the sign when it matters the most* is often more important than forecasting close to the realised value. We aim to provide the generator with more information, such that it better replicates the sign of the data, which is the crucial component of the task. The main reasons and benefits of using the Fin-GAN loss function terms can be summarised as follows:

- shift the generated conditional distributions in the correct direction,
- render GANs more amenable to a classification task,
- provide effective regularisation,
- help the generator learn more, especially in the case of weak gradients,
- help evade mode collapse; by making the generator loss surface more complex, the generator is less likely to converge towards sharp local minima [53].

5.3.2 The loss function

In the remainder of the section, we define three loss function terms, namely PnL, MSE, and Sharpe Ratio based, which we use to train the generator.

PnL term The first and most obvious choice of a loss function term to included in the generator's training objective (to maximise) is the PnL. However, the sign function is not differentiable, rendering it impossible to perform backpropagation. In order to alleviate this issue, we propose a smooth approximation to the pPnL, which we denote as PnL^*, defined as

$$PnL^*(\theta_g) = \mathbb{E}\left[PnL_a^*(x_{t+\Delta t}, G(a_t, Z_t; \theta_g)\right], \tag{5.8}$$

where $x_{t+\Delta t}$ is a data sample (return between t and $t+\Delta t$), a_t is the corresponding input condition, Z_t is i.i.d. $\mathbb{P}_Z$, and PnL_a^* is a smooth approximation to the PnL for a particular forecast

$$PnL_a^*(x, \hat{x}) = \tanh(k_{tanh}\hat{x})x. \tag{5.9}$$

The hyperparameter k_{tanh} controls the accuracy of our approximation. As values of excess returns are usually small, it is desirable for k_{tanh} to be large enough to have a good approximation, but at the same time small enough to have strong gradients which the generator can learn from.

MSE term Even though we are mainly interested in the sign of our forecast, we would also like to have $\hat{x}_{t+\Delta t}$ close to $x_{t+\Delta t}$. In order to enforce this, we add an MSE term to the training objective

$$MSE(\theta_g) = \mathbb{E}\left[\left(x_{t+\Delta t} - G(a_t, Z_t; \theta_g)\right)^2\right]. \tag{5.10}$$

Sharpe Ratio term As our main measure of performance is the annualised Sharpe Ratio, we introduce an SR-based loss function term for the generator to maximise. However, due to the prohibitive computational cost, we did not implement the weighted strategy during training, and instead worked with point estimates, given one noise sample per condition window. We do not consider daily PnLs, as the consecutive data points in during training do not necessarily come in order. We hence define the Sharpe Ratio term to be maximised by the generator during training, denoted by SR^*

$$SR^*(\theta_g) = \frac{\mathbb{E}\left[PnL_a^*(x_{t+\Delta t}, G(a_t, Z_t; \theta_g)\right]}{\sqrt{Var\left[PnL_a^*(x_{t+\Delta t}, G(a_t, Z_t; \theta_g)\right]}}. \tag{5.11}$$

Standard deviation (of the PnL series) term Maximising the Sharpe Ratio means maximising the PnL and minimising the standard deviation of the PnLs. Hence, it is natural to consider a standard deviation term, which we define as

$$STD(\theta_g) = \sqrt{Var\left[PnL_a^*(x_{t+\Delta t}, G(a_t, Z_t; \theta_g)\right]}. \tag{5.12}$$

Since the loss term defined above refers to the standard deviation of approximate PnLs, it should only be included in the training objective if the PnL^* term is included, due to the hierarchy principle. The SR^* term and the PnL^* combined with the STD term convey the same information, so the three should not be included in the loss at the same time. However, although the goal of maximising SR^* is the same of simultaneously maximising PnL^* and minimising STD, the gradient norms are different, which may result in different behaviour during training. Note that we consider standard deviation, rather than variance due to the inclusion of standard deviation in the SR^* term.

An economics-driven loss function for the generator. Finally, we integrate all of the aforementioned loss function terms defined in Equations (5.8), (5.10) and (5.11), and arrive at the Fin-GAN loss for the generator

$$L^G(\theta_d, \theta_g) = J^{(G)}(\theta_d, \theta_g) - \alpha PnL^*(\theta_g) + \beta MSE(\theta_g) - \gamma SR^*(\theta_g) + \delta STD(\theta_g), \tag{5.13}$$

where $J^{(G)}$ is one of the standard GAN losses (zero-sum, cross-entropy or Wasserstein), and the hyperparameters α, β, γ are non-negative. In prior numerical experiments, the Wasserstein-GAN with gradient penalty version [5] suffered from explosion, hence we opted to use the binary cross entropy loss for $J^{(G)}$ (1.3). The discriminator is a classifier as in the conditional GAN setting, also trained via the binary cross entropy loss (1.2). We aim to minimise a usual generator loss ($J^{(G)}$), maximise the PnL-based loss function term, minimise the MSE term, and maximise the SR-based loss function term, or jointly maximise the PnL and minimise the STD term. Assuming ergodicity, we replace the expectations with (mini-batch) sample averages for training.

The loss function combinations which we investigate are PnL^*, $PnL^*\&STD$, $PnL^*\&MSE$, $PnL^*\&SR^*$, $PnL^*\&MSE\&STD$, $PnL^*\&MSE\&SR^*$, SR^*, $SR^*\&MSE$, MSE, BCE only. The standard deviation term refers to the PnL term, so due to the hierarchy principle, it is included only if the PnL term is. When it comes to Fin-GAN, there needs to be an economics-driven loss term included, hence for Fin-GAN we only consider the following options: PnL^*, $PnL^*\&STD$, $PnL^*\&MSE$, $PnL^*\&SR^*$, $PnL^*\&MSE\&STD$, $PnL^*\&MSE\&SR^*$, $SR^*, SR^*\&MSE$. In other words, we impose the following conditions on $\alpha, \beta, \gamma, \delta$:

- $\max(\alpha,\gamma,\delta) > 0$ (*at least one additional term other than MSE is included*).
- If $\beta > 0$ then $\max(\alpha, \gamma, \delta) > 0$ (*if the MSE term is included, then another term is included*).
- If $\delta > 0$, then $\alpha > 0$ (the STD term is included only if the PnL^* term is).
- $\min(\gamma, \delta) = 0$ (*SR^* and STD terms are never included at the same time*).

We refer to conditional GANs trained via the loss function defined by Equation (5.13) and the rules above as Fin-GANs.

5.3.3 Benefits and challenges of the Fin-GAN loss function

The novel loss function terms do indeed shift the generated distributions, help evade mode collapse, and improve Sharpe Ratio performance, which we demonstrate in an extensive set of numerical experiments. Including the new loss function terms introduces four new hyperparameters to be tuned, rendering the optimisation a more challenging problem. However, the *gradient norm matching*, introduced in in Chapter 3, mitigates this issue.

In addition, the loss surface becomes more complex, raising convergence challenges. In initial numerical experiments, when exploring a wider range of hyperparameters α, β, γ, the MSE and the Sharpe Ratio terms appeared to encourage the generator to produce very narrow distributions. It is not immediately obvious why the Sharpe Ratio term with a high γ coefficient would result in mode collapse, as it encourages the $PnLs$ to have a low standard deviation, and not the generated samples. However, including the PnL-based term helped alleviate the issue of mode collapse, which is related to convergence to sharp local minima [53]. The PnL term helps the generator escape such areas of the loss surface.

When *gradient norm matching* was used to tune the hyperparameters $\alpha, \beta, \gamma, \delta$, there was no mode collapse in Fin-GAN no matter the initialisation of network weights. However, we noticed that using Xavier initialisation [59] instead of He initialisation [66] helped reduce the number of iterations of ForGAN with mode collapse. With He initialisation, this occurred in 67% of the cases. Opting for normal Xavier initialisation resolved this issue.

The classical generator loss term, i.e. the feedback received from the discriminator, allows learning conditional probability distributions, instead of only pointwise forecasts. The remaining loss function terms promote sign consistency (enforced by the PnL and Sharpe Ratio terms), while at the same time targeting to remain close to the realised value (MSE).

If the hyperparameters $\alpha, \beta, \gamma, \delta$ are too large, the feedback of the discriminator becomes negligible, and we move towards the standard supervised learning setting, using a generator-only model. On the other hand, if $\alpha, \beta, \gamma, \delta$ are too small, we are not providing any extra information to the generator, thus remaining in a classical GAN setting. The additional information about the data communicated through the novel loss function terms is especially invaluable in the case of weak gradients coming from the discriminator/critic feedback, as the generator can still make progress in this case. This all together leads to a better classification of the data in terms of directional price movements, i.e. correctly classifying the sign of future time series values, alongside uncertainty estimates of the movement direction. The gradient norm matching procedure ensures that all loss terms are treated as equally important, and that the corresponding hyperparameters are neither too large, nor too small.

5.4 Data description and implementation considerations

In this section, we outline the main characteristics of the data sets we use, and further describe the ForGAN architecture, which we use in implementation. Next, we discuss the training setup, including the hyperparameter choice as well as optimisation. Finally, we demonstrate the general behaviour of the Fin-GAN model, which recurrently surfaced throughout the numerical experiments.

5.4.1 Data description

We consider daily stock **ETF-excess** returns and daily raw ETF returns, extracted from CRSP on Wharton Research Data Services. The time frame of interest is Jan 2000-Dec 2021. All time series have the same training-validation-testing split, namely $80-10-10$. That is, the training period is 3rd Jan 2000 - 9th Aug 2017, the validation period is 10th Aug 2017 - 22nd Oct 2019, and the testing period is 23rd Oct 2019 - 31st Dec 2021. These three time periods are very different from an economic perspective, and it is important to note that the test data includes the start of the Covid-19 pandemic, rendering the problem more challenging. We consider close-to-open and open-to-close returns sequentially, and refer to each one of them as one unit of time. In other words, one input time series alternates intraday open-to-close and overnight close-to-open returns. The input condition is $a_t = (x_t, \dots, x_{t-(L-1)\Delta t})$, with $L = 10$, corresponding to one full trading week in this context.

The prices are adjusted by division with the cumulative adjustment factor provided ($cfacpr$), and both the intraday and the overnight returns are capped at $\pm 15\%$, to alleviate the effect of potential outliers. To pre-process the data, we first create one time series consisting of consecutive close-to-open and open-to-close returns, then chronologically split the data into three disjoint shorter time series (training-validation-testing), and we appropriately further process the three resulting time series.

We consider ETF-excess returns for 22 different stocks across nine sectors, and **raw returns** of the nine sector ETFs. All stocks are current members of the S&P500 index. The tickers, along with their sector memberships, are shown in Table 5.1.

Assuming that asset A has price S_t^A at time t, its return from time t to $t + \Delta t$ is defined as

$$R^A_{t+\Delta t} = \log\left(\frac{S^A_{t+\Delta t}}{S^A_t}\right). \tag{5.14}$$

Sector name	ETF ticker	Stock tickers
Consumer Discretionary	XLY	AMZN, HD, NKE
Consumer	XLP	CL, EL, KO, PEP
Energy	XLE	APA, OXY
Financial	XLF	WFC, GS, BLK
Health Care	XLV	PFE, HUM
Industrial	XLI	FDX, GD
Technology	XKL	IBM, TER
Materials	XLB	ECL, IP
Utilities	XLU	DTE, WEC

Table 5.1: Tickers and ETFs used for numerical experiments.

If the sector to which asset A belongs, has the corresponding ETF I whose return at time t is R_t^I, then the excess return of the asset A from time t to $t + \Delta t$ is defined as

$$re_{t+\Delta t}^A = R_{t+\Delta t}^A - R_{t+\Delta t}^I. \tag{5.15}$$

Then, in the case of a stock A the return we wish to forecast is $x_{t+\Delta t} = re_{t+\Delta t}^A$. Similarly, for an ETF I, we forecast raw returns $x_{t+\Delta t} = R_{t+\Delta t}^I$.

At time t, we enter a position based on the expected directionality of $x_{t+\Delta t}$. In case of ETF-excess returns, a positive outlook on $x_{t+\Delta t}$ would correspond to long stock and short ETF, the same nominal amount.

5.4.2 Architecture

For the single-stock/ETF numerical experiments, the ForGAN architecture [85] with LSTM cells [70] was used, as shown in Figure 5.1. Due to the small size of the data sets used, the corresponding layer dimensions and the noise dimension are all set to 8. The generator uses ReLU as activation function, while the discriminator uses the sigmoid.

Although we use the ForGAN architecture to study the performance of FinGAN, an additional area of interest would be to explore how one might combine the Fin-GAN loss with architectures such as COT-GAN [143] and Time-GAN [145].

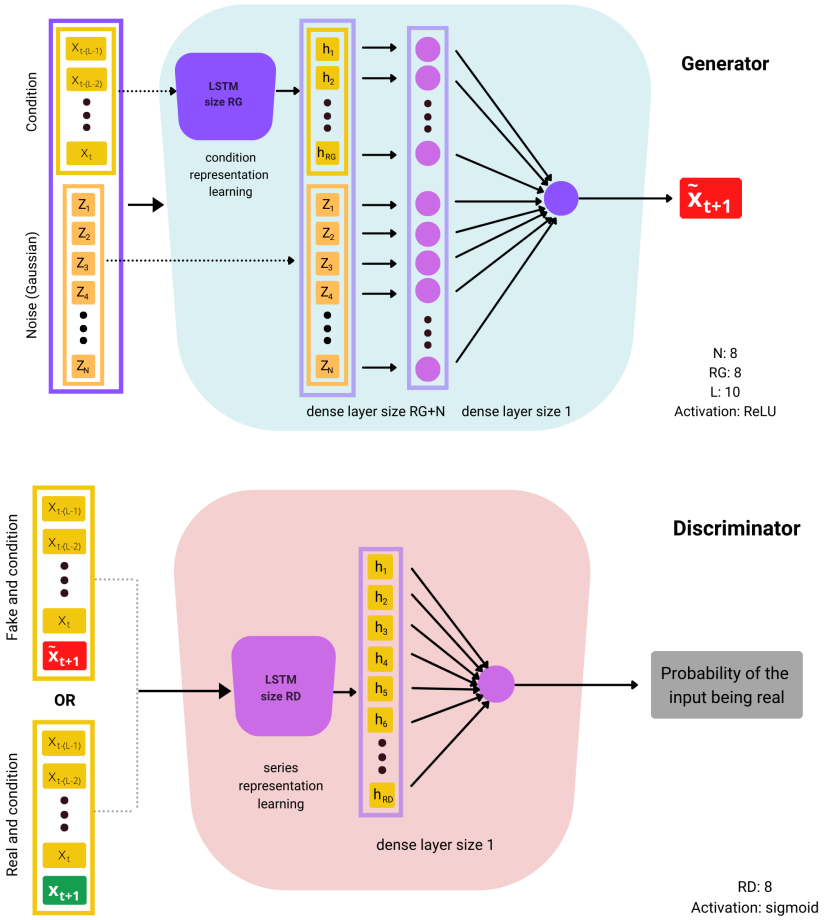


Figure 5.1: ForGAN architecture using an LSTM cell.

5.4.3 Training

Avoiding further hyperparameter tuning, the optimiser used throughout is RMSProp [68], and normal Xavier initialisation [59] is used for the weights. The discriminator and the generator are trained *using an alternating direction method*, having one discriminator update per one generator update. We use reference batch normalisation, picking the first 100 samples from the training data serving as the reference batch, to not be anticipative. Due to time dependence, LSTM cell state and hidden state are both set to zero at every call, aiming to learn one-step transitions. Mini-batch size is 100. The learning rates of both networks are set to 0.0001. We train for 25 epochs at the start, using the BCE loss only in order to implement the *gradient norm matching*, and find the values for $\alpha, \beta, \gamma, \delta$ coefficients. We then *branch away* from the generator and the discriminator by training via every suitable loss combination for 100 more epochs. At the end of the training stage, we calculate Sharpe Ratio on the

validation set, and choose the loss function combination, and the optimised generator which maximises it. Due to differences in economic periods for validation and testing, it may not be the case that the networks which would perform well on the test set are chosen. However, this approach allows us to ensure good performance, regardless of periods of crisis. The code is implemented in PyTorch. The Fin-GAN algorithm is summarised in Algorithm 2.

Remark 9. *One may explore a wider range of hyperparameters α, β, γ, δ, learning rates lr_g, lr_d, hidden dimensions RG, RD, noise dimension N, as well as completely different architectures, by evaluating Sharpe Ratio on the validation set, and opting for the hyperparameter/architecture maximising it.*

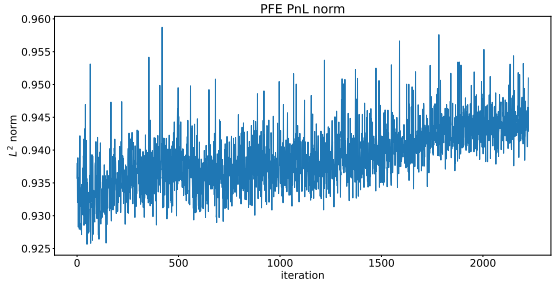
Algorithm 2 Fin-GAN algorithm

Input: Hidden dimensions of the generator and discriminator; noise dimension; target size; condition window; discriminator and generator learning rates; training data; validation data; testing data; number of epochs for gradient matching n_{grad}; number of training epochs n_{epochs}; minibatch size n_{batch}; mode collapse threshold ϵ; number of samples for the weighted strategy B;

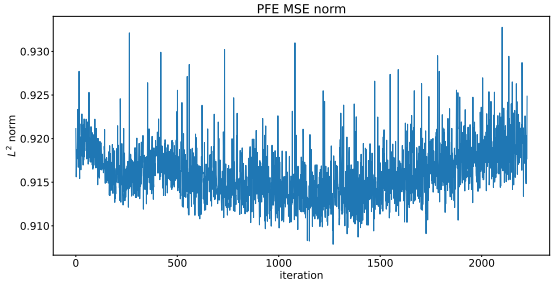
Step 0: Take the first n_{batch} items from the training data set as the reference batch for data normalisation. Initialise the generator *gen* and the discriminator *disc* networks.
Step 1: *Matching the gradient norms to find $\alpha, \beta, \gamma, \delta$.*
  for n_{grad} epochs **do**
    Split the training data into the minibatches.
    for number of minibatches **do**
      Calculate norms of gradients of the BCE, PnL^*, MSE, SR^*, STD terms with respect to θ_g.
      Label them as $grad_0, grad_\alpha, grad_\beta, grad_\gamma, grad_\delta$.
      Update *disc* and then *gen* parameters via RMSProp and the BCE loss.
    end for
  end for
  $\alpha \leftarrow \text{mean}(grad_0/grad_\alpha)$; $\beta \leftarrow \text{mean}(grad_0/grad_\beta)$;
  $\gamma \leftarrow \text{mean}(grad_0/grad_\gamma)$; $\delta \leftarrow \text{mean}(grad_0/grad_\delta)$
Step 2: *Training and validation.*
Label the possible loss function combinations (PnL^*, $PnL^*\&STD$, $PnL^*\&MSE$, $PnL^*\&SR^*$, $PnL^*\&MSE\&STD$, $PnL^*\&MSE\&SR^*$, SR^*, $SR^*\&MSE$) as L_i, for $i = 1, \ldots, 8$ respectively.
  for $i = 1, \ldots, 8$ **do**
    $gen_i \leftarrow gen$; $disc_i \leftarrow disc$.
    for n_{epochs} **do**
      Split the training data into the minibatches.
      for number of minibatches **do**
        Update *disc* via RMSProp and $J^{(D)}$.
        Update *gen* via RMSProp and L_i.
      end for
    end for
    Take B samples from gen_i for each day in the validation set.
    $SR^i_{val} \leftarrow$ Sharpe Ratio on the validation set.
  end for
  $i^* \leftarrow \text{argmax}\{SR^i_{val} : i = 1, \ldots 8\}$; $gen^* \leftarrow gen_{i^*}$.
Step 3: *Evaluation on the test set.*
  For each time t in the test set, take B i.i.d. outputs from gen_{i^*}, $\hat{r}^i_t$, $i = 1, \ldots, B$.
  Point estimate for each time $\tilde{r}_t \leftarrow \text{mean}(\hat{r}^i_t)$.
  $MAE \leftarrow MAE(r_t, \tilde{r}_t)$; $RMSE \leftarrow RMSE(r_t, \tilde{r}_t)$;
  Implement the strategy, pair up days into two.
  Calculate the annualised Sharpe Ratio SR and the mean daily PnL PnL.
  if $std(\{\hat{r}^i_0\}_{i=1,\ldots,B}) < \epsilon$ or $std(\{\tilde{r}_t\}_{t \in \text{test set}}) < \epsilon$ **then**
    Mode collapse.
  else
    No mode collapse.
  end if

5.4.4 Gradient stability

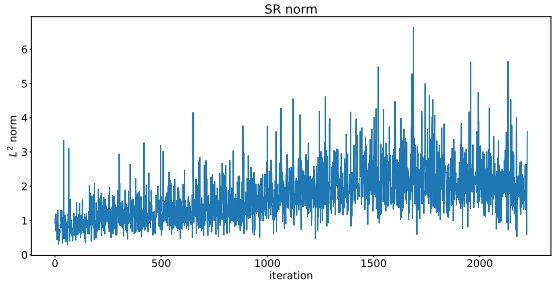
Before incorporating the additional loss function terms, we first investigate the behaviour of the gradient norms during training, in order to check for exploding and vanishing gradients. We train ForGAN, performing updates on the BCE loss using RMSProp [68], for 25 epochs, the period used for *gradient norm matching*. In Figure 5.2, we display sample gradient norms of each term Equation (5.13). The data used in this example is the daily excess stock returns time series of PFE. We observe that there are no vanishing or explosion phenomena, and that the gradient norms are on a similar scale. This also holds true for other tickers used in our experiments.



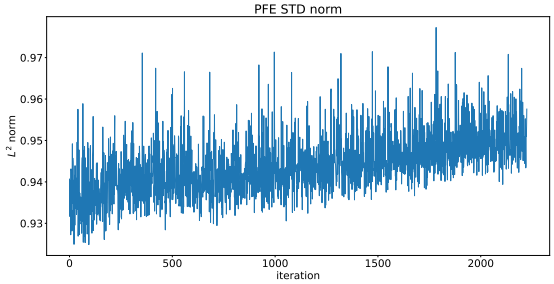
(a) PnL term.



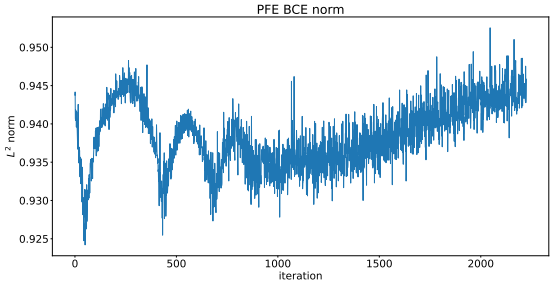
(b) MSE term.



(c) SR term.



(d) STD term.



(e) BCE term.

Figure 5.2: Sample generator gradient norms during training of different terms (PnL, MSE, SR, STD, BCE) with respect to θ_g. Updates were performed using the BCE loss only.

5.4.5 Generated distributions

Samples of generated distributions when training using different Fin-GAN loss function term combinations are displayed in Figure 5.3. All of the distributions shown are generated using the same training data (excess returns of PFE), and the same condition in the test set (they have the same target). Throughout the numerical experiments, training via just the BCE loss on its own would produce more symmetrical distributions, whereas the incorporated Fin-GAN loss function terms would help shift the forecast distributions. We also investigate the generated means and compare them with the true distributions of the target in Figure 5.4. We note that in this particular example, the PnL and STD combination has a similar behaviour to the SR term on its own. This is not surprising as both convey the same information, although they have different gradients. Both options have the biggest distribution-shifting effect in this example. We note that apart from the PnL with STD, and SR loss terms, all of the generated means resemble the true target distribution. Furthermore, including the MSE term prompted the distribution of the means to be closer to that of the true returns.

Figure 5.3 allows us to illustrate the weighted strategy (based on the expected sign of the forecast) that GANs are able to employ due to uncertainty estimates. In this particular forecast, the SR, as well as PnL and STD, would result in long ETF and short stock with weight one, whereas the PnL, MSE and SR combination would have a similar trade of a slightly lower weight, and the other combinations would result in very small trades due to uncertainty about the sign of the forecast. Although we are considering the Fin-GAN loss terms only in Figures 5.3 and 5.4, it is important to note that the average correlation between the BCE loss and the BCE loss with the MSE term added to it is 99.7%.

We observe that it is beneficial to train on a combination of the loss terms, as they are able to produce shifts in generated distributions and evade mode collapse. The benefit of jointly using the Fin-GAN loss function terms is backed up by the Sharpe Ratio performance, which we analyse in Section 5.6.

It is important to highlight that the configuration chosen as optimal during the validation stage will vary from instance to instance. Figures 5.3 and 5.4 are illustrative examples of resulting distributions obtained by training via different objectives.

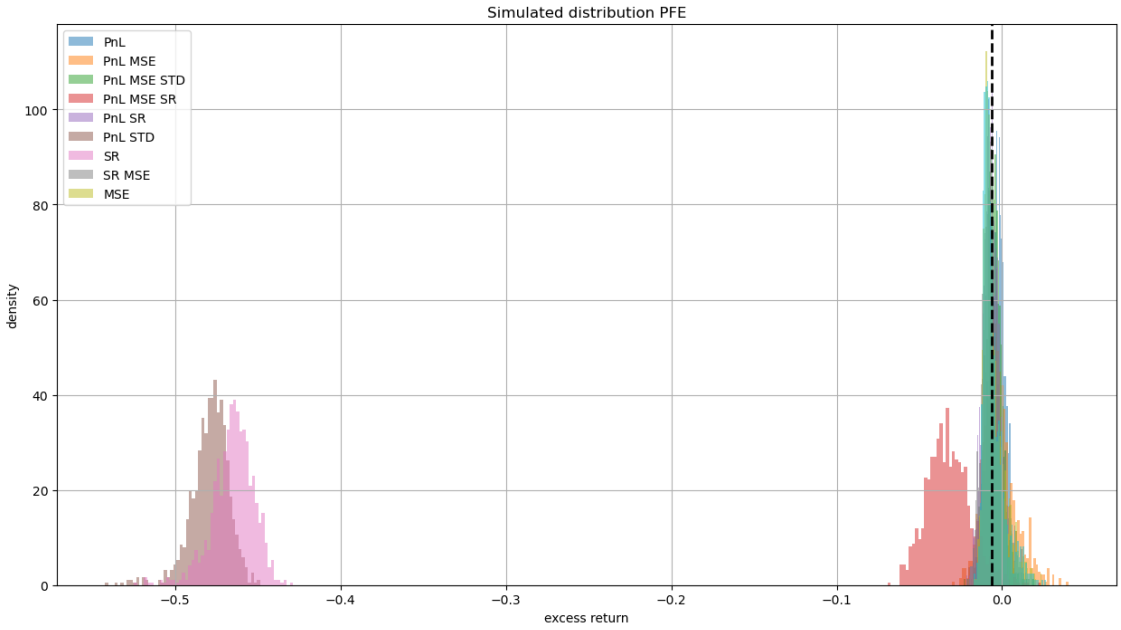


Figure 5.3: An illustration of how the Fin-GAN loss function terms can shift generated distributions. All the distributions shown are generated using the same condition window. The black vertical line is the true value, the target. The data used are the ETF-excess returns of PFE.

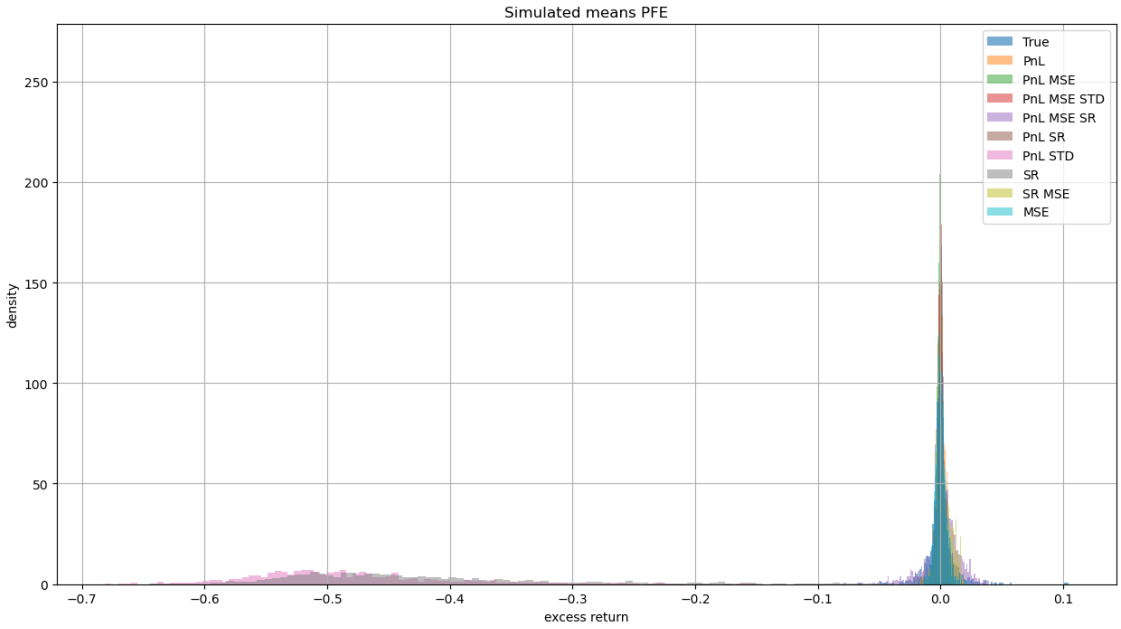


Figure 5.4: Illustration of generated out-of-sample (one-step-ahead) means on the test set, obtained by training on different loss function combinations. Training data: PFE excess returns. The loss function with the best Sharpe Ratio performance on the validation set is PnL-MSE-SR, for this particular instance.

5.5 Benchmark algorithms

Apart from the standard ForGAN (GAN with the ForGAN architecture trained via the BCE loss), we compare our Fin-GAN model with more standard supervised learning approaches to time series forecasting: ARIMA and LSTM. For completeness, and to demonstrate that the task at hand is non-trivial, we further include PnL and Sharpe Ratio values of long-only strategies, where the expected sign is $+1$ for each observation. All comparisons have the same training-validation-testing split as those used in the Fin-GAN experiments.

ARIMA. We first recall the *auto regressive integrated moving average* (ARIMA), a very classical time series model [131]. The parameters p, d, q in ARIMA(p, d, q) correspond to the autoregressive, differencing, and moving average parts, respectively. The differencing coefficient d indicates how many times we need to difference our initial time series in order to reach a time series X_t which has no unit roots. To determine d, we perform the augmented Dickey-Fuller (ADF) test [49]. In our case, all of the time series had p-values smaller than 10^{-6}, indicating stationarity. This is not surprising, since we are working with returns and excess returns, which are already differenced log-prices time series. Therefore, in our setting, we set $d = 0$ throughout, hence working with ARMA(p, q) models of the form

$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}, \tag{5.16}$$

where $\epsilon_1, \ldots, \epsilon_t$ are white noise terms, θ_i are the moving average parameters, and α_j are the autoregressive parameters. In order to determine the most suitable p and q, we plot the autocorrelation (ACF) and partial autocorrelation (PACF), which indicate that $p, q \in \{0, 1, 2\}$. We fit ARMA(p, q) with $(p, q) \in \{(1, 0), (1, 1), (2, 0), (2, 1), (2, 2)\}$ and choose the model with the lowest Akaike Information Criterion (AIC).

LSTM. Long short-term memory networks, or LSTMs [70], belong to the class of recurrent neural networks. They help with the vanishing/exploding gradients problem, and are particularly well-suited for working with time series. At every call time, there is the input, the previous cell state and the previous hidden state (the previous LSTM output). These are then processed by the forget gate, the input gate, and the output gate, in order to create the new cell state and the new hidden state, the output. An illustration of the LSTM cell architecture is shown in Figure 5.5.

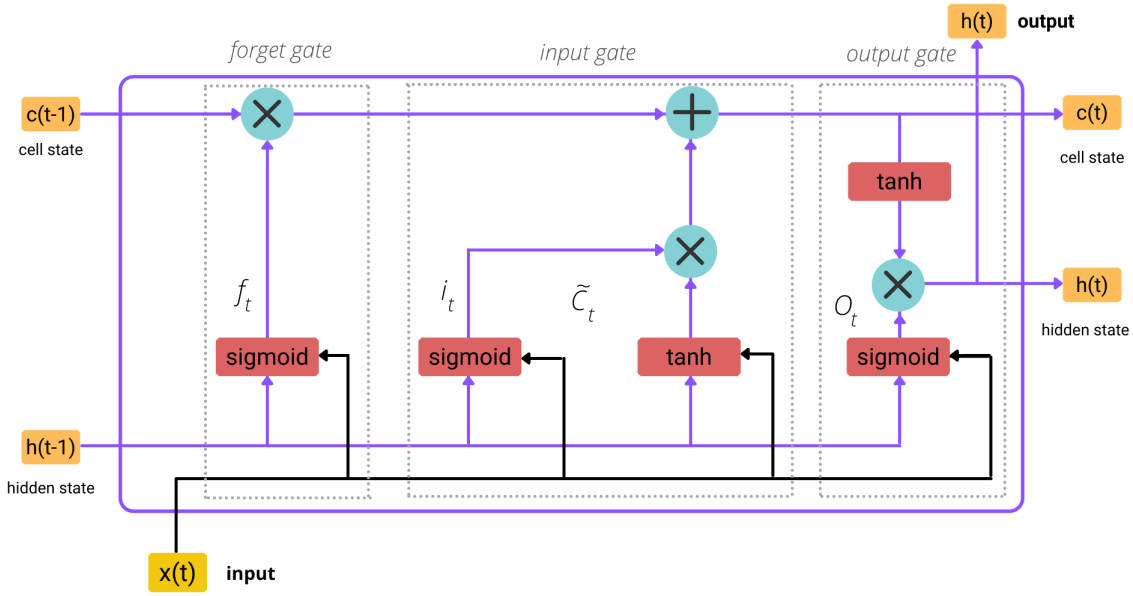


Figure 5.5: Illustration of an LSTM cell.

For training, we use the same setup as for Fin-GAN. That is, we use RMSProp as the optimiser of choice, train for 125 epochs in total. We are in a regression setting now, so the LSTM is trained to minimise the MSE loss. The rate is once again 0.0001.

LSTM-Fin. We also train LSTM on the appropriate combinations of the (baseline) MSE loss, PnL^*, SR^*, and STD loss terms, in order to have a better comparison with Fin-GAN. We use the same methodology as we did in the Fin-GAN setting, also performing gradient norm matching to determine the values of the hyperparameters corresponding to the newly included loss terms. That is, we consider the loss function (5.17), including and excluding the PnL^*, SR^*, and STD loss terms via the same rules that apply to Fin-GAN, determining the values of the hyperparameters α, γ, δ using the same gradient norm matching procedure, and determining the final loss function combination to be used for testing by evaluating the Sharpe Ratio on the validation set.

$$L^{Fin}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = MSE(\boldsymbol{x}, \hat{\boldsymbol{x}}) - \alpha PnL^*(\boldsymbol{x}, \hat{\boldsymbol{x}}) - \gamma SR^*(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \delta STD(\boldsymbol{x}, \hat{\boldsymbol{x}}). \tag{5.17}$$

Long-only strategies. We also remark on the results of long stock and short ETF (for the stock data); and long-only ETF (for the ETF data), which corresponds to constantly forecasting $+1$ for the sign.

5.6 Empirical results

In this section, we first consider a single-ticker setting. That is, we train Fin-GAN and the benchmarks on ETF-excess returns/raw returns of a particular stock/ETF. We then explore the notion of universality, in the spirit of [125], on three different sets of stocks. We pool the data across the tickers used in the single-ticker setting, and train Fin-GAN as if the data was coming from the same source. We repeat this methodology on stocks belonging to the same sector (Consumer Staples), with and without the XLP raw returns used in the training stage. In the universal setting, we further investigate the performance of Fin-GAN on stocks not previously seen by the model during training.

Remark 10. *In this and the following chapter, transaction costs and bid–ask spreads are not considered. Hence, these simulations represent idealised, frictionless strategies. In practice, spreads would affect the trade size and frequency. For example, rebalancing might occur only if the absolute expected return exceeds the bid–ask spread, or if there is a significant change in the expected forecast sign. While this simplification does not alter the validity of our methodology, it would reduce absolute performance levels in real-world implementation.*

5.6.1 Single stock & ETF settings

We first examine the summarised performance of the models under consideration: Fin-GAN, ForGAN (trained via the BCE loss), LSTM, LSTM-Fin, ARIMA, and the long-only approach. The statistics of interest are the annualised Sharpe Ratio, mean daily PnL, MAE, and RMSE. All statistics are summarised by computing the mean and the median across the 31 data sets (22 stocks and 9 ETFs) used in the numerical experiments. Additionally, we report on the *portfolio Sharpe Ratio* by first summing the daily Pnls across all tickers in the universe, and then computing the Sharpe Ratio of the resulting daily PnL time series. These summary statistics are shown in Table 5.2.

We remark that the highest mean, median, and portfolio Sharpe Ratios are achieved by Fin-GAN, followed by LSTM. The median Sharpe Ratio achieved by Fin-GAN is close to twice that of LSTM. Furthermore, we note that despite the Sharpe Ratio being the highest when using the Fin-GAN approach, the highest mean PnLs are achieved by the LSTM, and the highest median PnLs by ARIMA. This altogether demonstrates the significant reduction in variance and higher consistency of the generated PnL by Fin-GAN, compared to the other models. It is also a consequence of placing smaller

	Fin-GAN	**ForGAN**	**LSTM**	**LSTM-Fin**	**ARIMA**	**Long-only**
Mean SR	**0.540**	0.033	0.467	0.341	0.206	0.182
Median SR	**0.413**	-0.092	0.214	0.170	0.204	0.194
Portfolio SR	**2.107**	0.172	2.087	0.942	0.612	0.618
Mean PnL	2.978	0.25	**4.123**	2.361	2.059	2.350
Median PnL	1.890	-0.673	1.959	1.735	**2.245**	1.975
Mean MAE	0.044	0.052	0.007	0.007	**0.007**	–
Median MAE	0.008	0.009	**0.007**	0.007	0.007	–
Mean RMSE	0.049	0.056	0.012	0.012	**0.012**	–
Median RMSE	0.012	0.014	0.011	0.011	**0.011**	–

Table 5.2: Summary of performance metrics over the models across the stocks and ETFs. SR refers to the annualised Sharpe Ratio, and PnL refers to the mean daily PnL. MAE and RMSE represent the mean absolute error and the mean root squared error, respectively. Highlighted are the best-performing results according to each metric.

trades compared to other strategies due to the ability to leverage the uncertainty estimates and implement the weighted strategy. We stress the beneficial effect of the novel loss function terms on performance, evidenced by the mean, median, and portfolio Sharpe Ratios achieved by Fin-GAN being significantly higher than those achieved by ForGAN trained on the BCE loss only.

We further note that ARIMA attains the best performance in terms of RMSE and achieves the best mean MAE, while LSTM achieves the lowest median MAE. We also remark that the non-GAN models have their MAE and RMSE summary statistics on the same scale, while the mean MAE and RMSE are significantly higher for the GAN models. However, the medians are on a similar scale to the other models, and errors are reduced when moving from ForGAN to Fin-GAN. Investigating RMSE and MAE in Figures 5.11 and 5.10, respectively, we find that the main cause of such high mean MAE and RMSE achieved by Fin-GAN is performance on IBM. However, Fin-GAN achieves higher Sharpe Ratio (Figure 5.6) and PnL (Figure 5.7) than the other models in this case. Similarly, even though ARIMA attains values close to the realised ones, it does so on the opposite side of the real line, when the movements in the underlying are large, resulting in lower PnL and lower Sharpe Ratios. We note that the summary statistics achieved by the non-deep learning approaches, ARIMA and long-only, are similar, and that they achieve better performance than ForGAN trained via the BCE loss (1.2)-(1.3). The long-only column indicates that the task at hand is non-trivial, given that the only ticker on which a Sharpe Ratio above one is achieved is XLK. This is not surprising given the continued increase in stocks belonging to the technology sector during the Covid-19 pandemic. Furthermore, the

positive effect that the economics-driven loss function terms had in the GAN setting is not visible in the case of LSTMs.

The above experiments have shown that Fin-GAN outperforms, on average, ForGAN, LSTM, ARIMA and long-only benchmarks. We now further investigate the behaviour of Sharpe Ratios at the individual ticker level. The annualised Sharpe Ratio performance across the models and tickers is displayed in Figure 5.6. We observe that the Fin-GAN model outperforms all other benchmarks, being able to achieve very competitive Sharpe Ratios, especially in light of the fact that we are dealing with single-instrument portfolios. Sharpe Ratios achieved by Fin-GAN are more stable than those of LSTM, and the only ticker with Sharpe Ratio below -1 is XLY (Consumer Discretionary). This is not surprising, as this sector ETF had a crash at the start of the Covid-19 pandemic, and was consistently growing during the time used for validation. On the data sets on which LSTM achieves Sharpe Ratios above 1 or 2, Fin-GAN does so as well, achieving similar Sharpe Ratios, or a group higher in the case of CL. Altogether, we observe clear benefits from using Fin-GAN when compared to the other methods, in terms of Sharpe Ratio performance. The breakdown of the Fin-GAN cumulative PnL performance by ticker is displayed in Figure 5.8. We remark that the Covid pandemic had a different effect on different stocks, and that performance in March-June 2020 is the main cause of volatility in the attained Pnls.

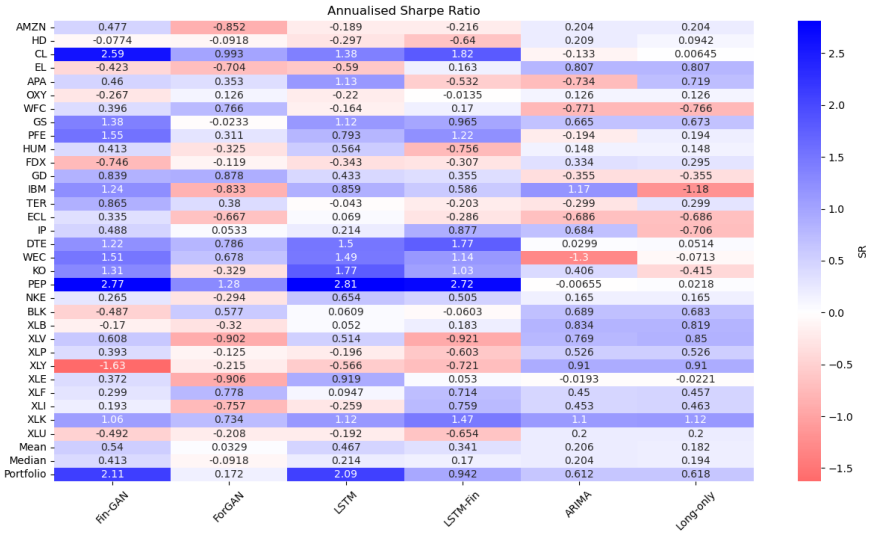


Figure 5.6: SR (annualised Sharpe Ratio) obtained by different methods on the test set.

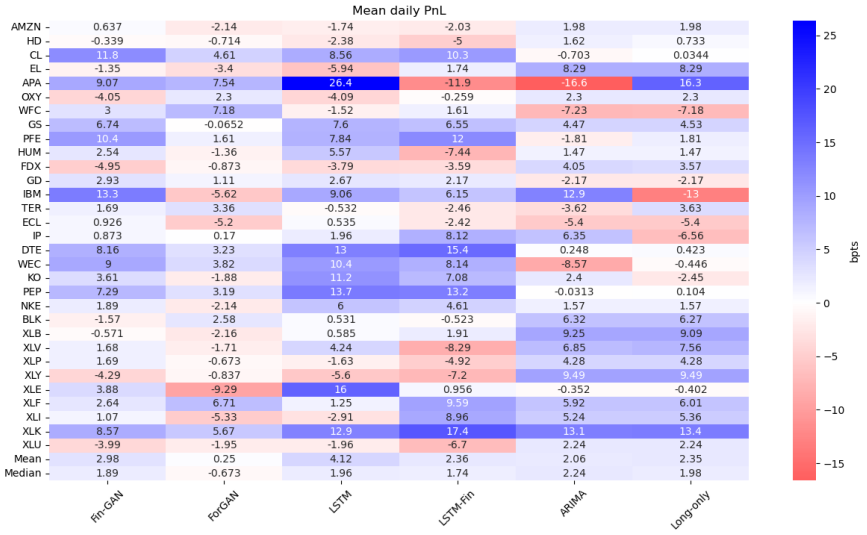


Figure 5.7: Mean daily PnL in basis points obtained by different methods.

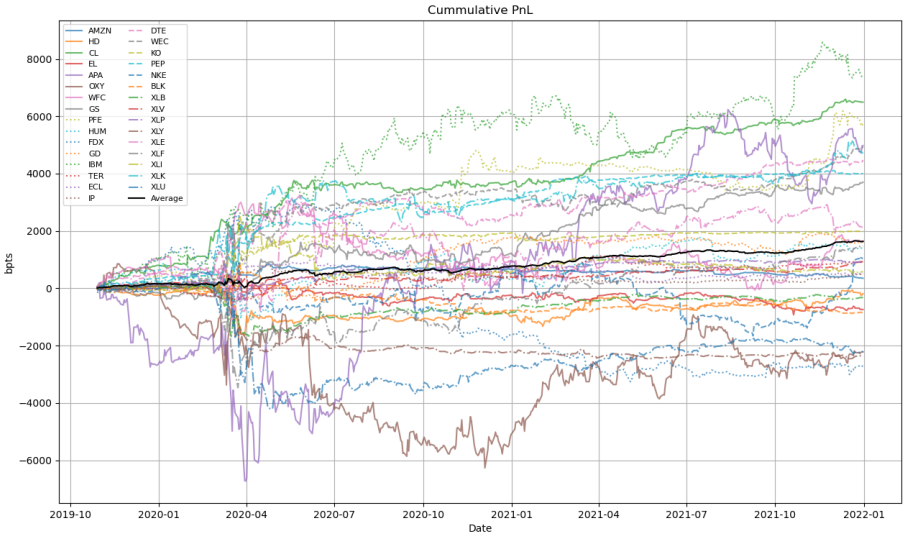


Figure 5.8: Cumulative PnL across tickers achieved by different loss function combinations of Fin-GAN. The portfolio PnL is the average PnL displayed in black, multiplied by the number of instruments. A comparison of the overall portfolio performance across the benchmarks (and the Fin-GAN loss function combinations) is shown in Figure 5.9.

In terms of the overall performance, Fin-GAN and LSTM outperform all the other methods under consideration. Cumulative PnL plots of the corresponding portfolios

are shown in Figure 5.9. The deep learning models recover from the Covid-19 shock much faster than ARIMA and long-only do, and we note that most of the volatility in the generated PnLs is stemming from the pandemic. In Figure 5.9 we display the cumulative PnLs achieved by different Fin-GAN loss combinations, including MSE with BCE term only, which on average has a 99.7% correlation with ForGAN. It is evident that the overall performance is increased by using validation, rather than using the same loss combination from the start for every data set. The LSTM has a higher portfolio PnL, but the path is more volatile, especially from March 2020 until June 2020, resulting in a lower Sharpe Ratio than Fin-GAN.

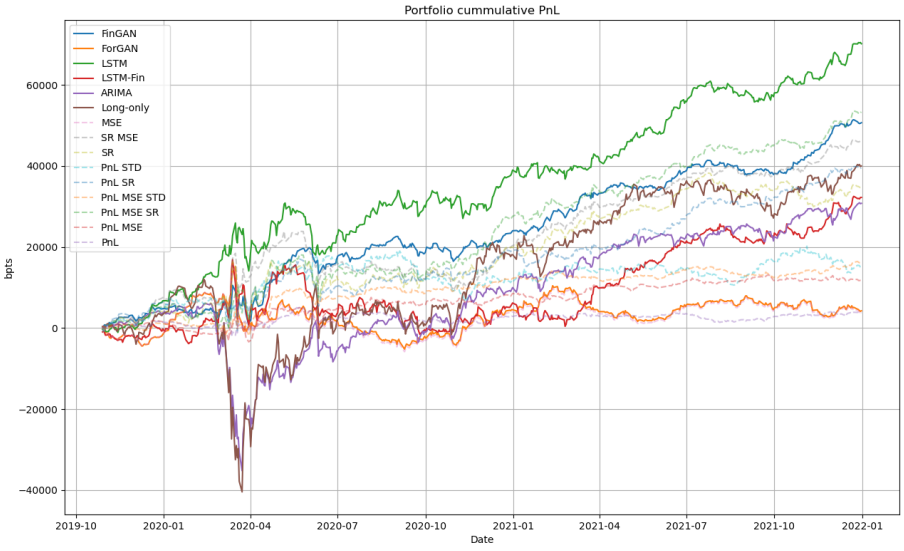


Figure 5.9: Portfolio cumulative PnL of different models. Dashed lines correspond to PnL paths generated by the appropriate Fin-GAN loss function combinations (including MSE alone).

Concerning the mean daily PnL performance displayed in Figure 5.7, we note that ARIMA and the long-only approach have a tendency of producing mean daily PnLs on the same scale, while LSTM attains the highest PnLs on average. However, the PnLs achieved by LSTM fluctuate significantly from ticker to ticker. The mean daily PnLs achieved by the GAN approaches have a much lower standard deviation (4.85 for Fin-GAN, 4.00 for ForGAN) than LSTM (7.48), LSTM-Fin (7.53), ARIMA (6.26), and long-only (5.97), showcasing the benefits of being able to leverage the uncertainty estimates for developing a weighted strategy with dynamic sizing.

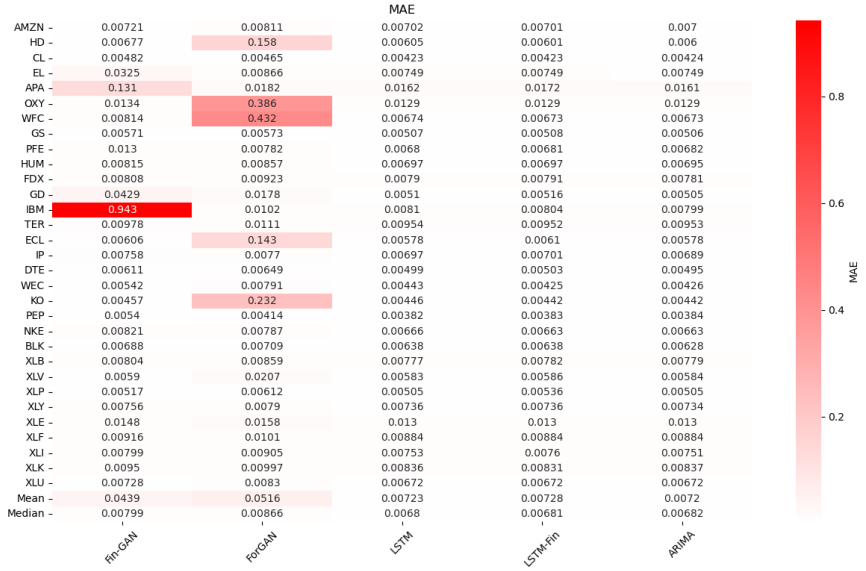


Figure 5.10: MAE obtained by different methods.

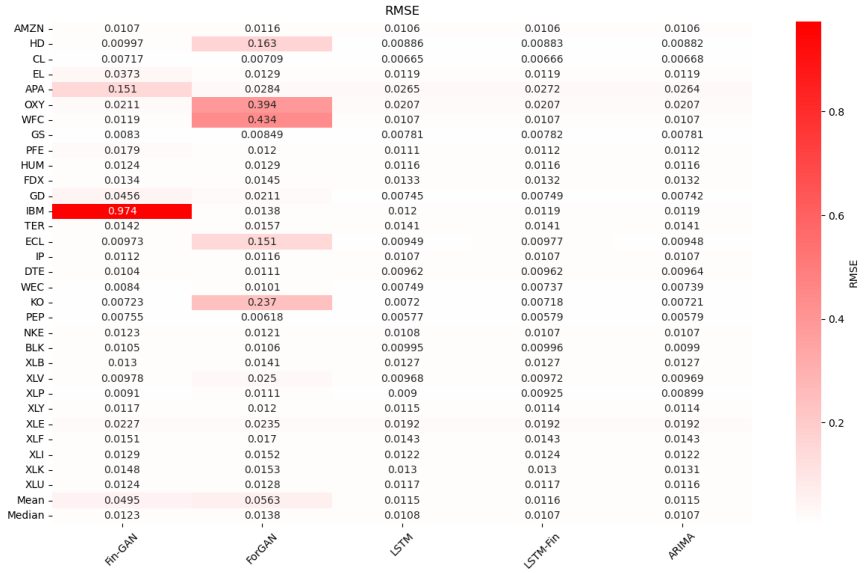


Figure 5.11: RMSE obtained by different methods.

Next, we comment on the breakdown of the chosen loss function combinations in the Fin-GAN performance. As shown in Figure 5.3, it can be beneficial to utilise the introduced loss function terms together, both for shifting distribution purposes, achieving a better approximation to the data distribution, and for avoiding mode

collapse. We remark that the inclusion of the MSE term is very prominent, and that interestingly, the PnL and Sharpe Ratio combination was never chosen. The most common choice was Sharpe Ratio with MSE, in 29% of the cases. The breakdown of the number of times each of the combinations was used is shown in Table 5.3.

Combination	Count
PnL	3
SR	3
PnL & MSE	5
PnL & SR	0
SR & MSE	9
PnL & MSE & SR	4
PnL & STD	5
PnL & STD & MSE	2

Table 5.3: Number of chosen (highest SR on the validation set) Fin-GAN loss function term combinations across the data.

We compare the Sharpe Ratio performance of the different Fin-GAN loss function combinations on the test set, and show the results in Figure 5.12. Due to the differences in validation and test set, sometimes a sub-optimal loss function combination is chosen during the validation stage. However, we note that performing validation in order to choose which terms to include in the training objective improves the Sharpe Ratio performance of the model.

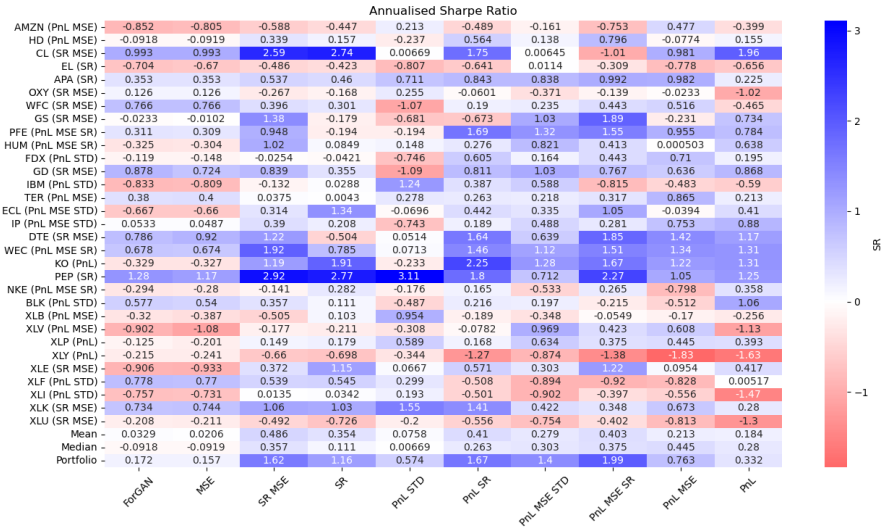


Figure 5.12: SR (annualised Sharpe Ratio) obtained by different loss combinations of Fin-GAN on the test set. The chosen loss combination is reported in parentheses, for each ticker.

As previously discussed, SR and the combination of PnL and STD (PnL & STD) terms convey the same information, but have different gradients, and may result in different forecasts. Hence, we investigate the average Pearson correlation of the obtained out-of-sample PnLs (test set) by different loss term combinations. It is not surprising that there is a high correlation between combinations with the MSE term included, and the corresponding ones with the MSE term excluded. The correlation between the SR term and PnL & STD term is 33.5%, indicating that the two learn very different distributions, while once the MSE term is included, the correlation increases significantly.

An important finding is that the correlation between the MSE term with the BCE loss and the BCE loss is 99.7% on average, implying that ForGAN trained via the binary cross entropy loss is already aiming to produce outputs close to the real values. This behaviour could be simply explained by the structure of the data and the task at hand: since there is a *true target* for each condition, that is, the empirical conditional distribution is point mass $p_{data}(x_{t+\Delta t}|a_t) = \delta(x_{t+\Delta t})$ if L is large enough. Hence, it is not surprising that the forecast values close to the target would receive high scores from the discriminator, encouraging mode collapse in a classical GAN setup. This further supports the decision to use ForGAN as the baseline, its suitability for probabilistic forecasting, and the necessity to customise it to a financial setting.

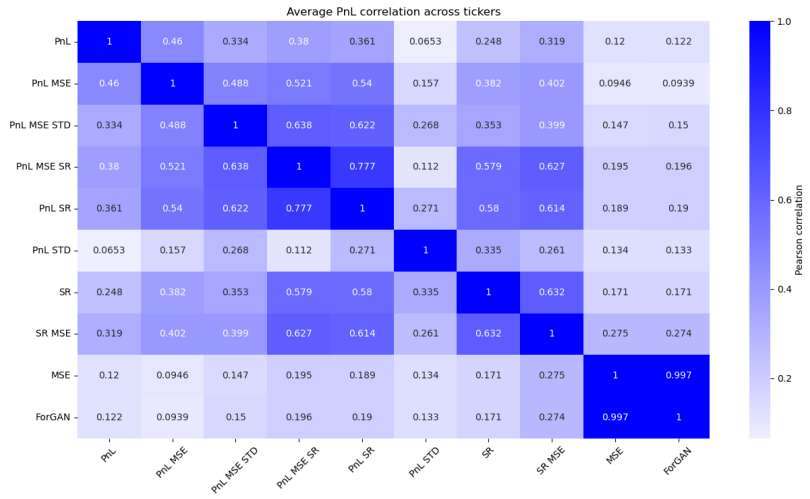


Figure 5.13: Mean out-of-sample (test) correlation of PnLs across different tickers of the Fin-GAN loss term combinations.

5.6.2 Universality

We test for universality in our approach, in the spirit of [125]. However, due to computational cost, we are only able to perform numerical experiments on a small universe of stocks and ETFs, the 31 previously used. We do not attempt to learn from the cross-asset correlations, but rather consider each time series individually. That is, we employ the single-asset model, where we combine (pool) the data from all 31 tickers. We test the performance of the model on all stocks used for training, and on four additional unseen stocks. The time periods for the training, validation, and test sets across different tickers, as well as the Fin-GAN methodology remain the same. A summary of the Sharpe Ratio performance of the small universal model is shown in the heatmap displayed in Figure 5.14. The *Single Stock* column refers to the highest Sharpe Ratio achieved by the single-stock model when training on a particular ticker only. Stocks CCL, EBAY, TROW, and CERN have not been seen by the model during the training stage. We note that it is possible to achieve competitive Sharpe Ratios even on unseen stocks. For example, Sharpe Ratios achieved on THROW data are often above 1, despite the fact that the universal model has never seen the EBAY data. Figure 5.14 indicates that some of the stocks included in the training of the universal model benefit from the pooled training with other stocks, but not all. This is most visible for XLY, where the achieved Sharpe Ratios are either significantly less negative, or even positive. The overall performance not increasing could be due to our universe of stocks being small, and not having strong correlations. One would expect that cross-asset interactions are more informative when stocks belong to the same sector.

We repeat the same analysis using stocks from the Consumer (XLP) sector. We perform one set of numerical experiments with XLP data (raw returns) included in the training data, and another set of experiments without it. The Sharpe Ratio performance without the XLP data included is shown in the heatmap displayed in Figure 5.15, while same performance when the XLP data is included is shown in Figure 5.16. Stocks unseen by the model are SYY and TSN. Similarly to the previously discussed universal model, Sharpe Ratios achieved on the unseen data can be very competitive, see for example TSN. When XLP data is included, the model chosen by validation is PnL & SR, achieving a good overall portfolio SR of 1.43 on the test set. Although not all stocks have a positive Sharpe Ratio, this combination achieves good and even very good SRs, including the Sharpe Ratio of 1.55 on unseen TSN data. An important observation is that the Sharpe Ratio on XLP data, in this case, is 1.11, compared to 0.393 in the single-stock setting. Other ETFs might as well benefit

from being trained alongside their constituents. When XLP data is removed from the pool of stocks used for training, the model of choice is SR. The portfolio Sharpe Ratio reduces to 1.18, but remains competitive. Comparing the heatmaps in Figures 5.15 and 5.16, we conclude that there is a clear benefit from including the ETF data in the training process of Fin-GAN, but not of ForGAN. Training alongside the XLP data results in more stable Sharpe Ratios compared to the model without it.

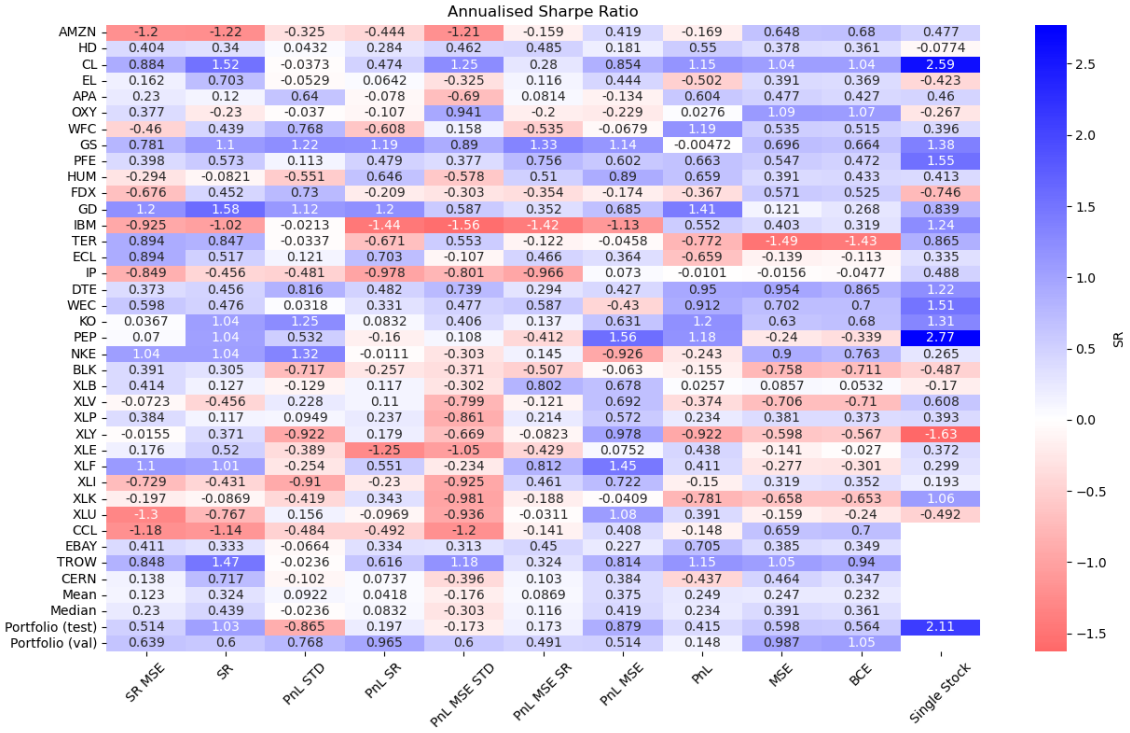


Figure 5.14: Summary of Sharpe Ratio performance for individual stocks in the universal model. Each column represents a different combination of the loss function terms. The *Single Stock* column shows the best Fin-GAN performance when trained on a particular stock/etf. CCL, EBAY, TROW and CERN have not been seen by the model during the training stage.

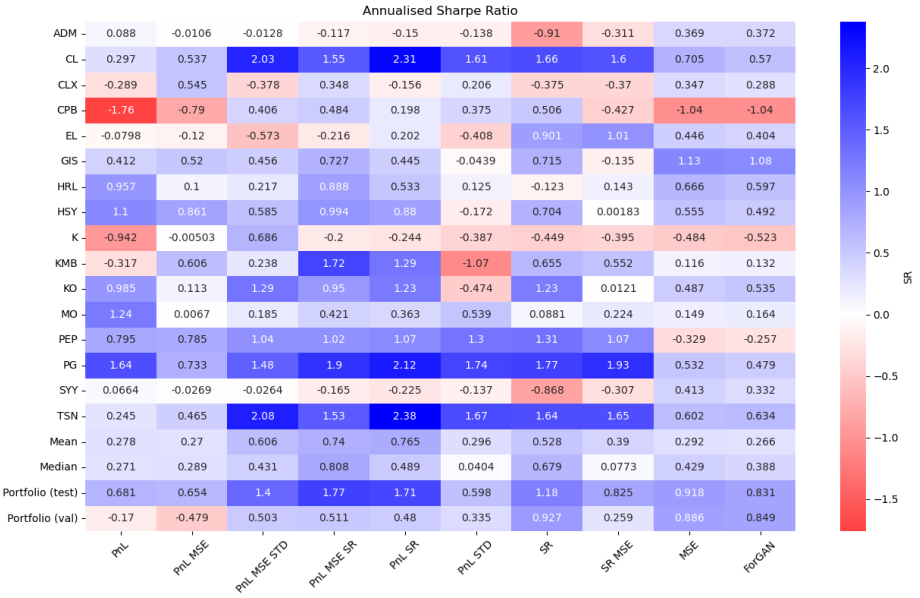


Figure 5.15: Summary of Sharpe Ratio performance on stocks constituents of the XLP sector. Each column represents a different combination of loss function term. SYY and TSN have not been seen by the model during the training stage.

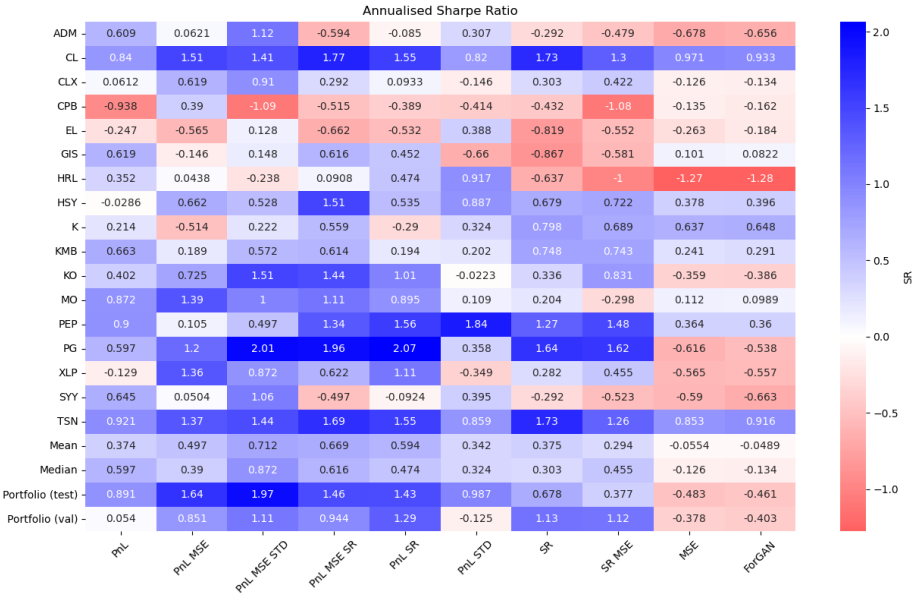


Figure 5.16: Summary of Sharpe Ratio performance on the stocks belonging to the XLP sector, with XLP data included in the training data. Each column represents a different combination of loss function terms. SYY and TSN have not been seen by the model during the training stage.

Chapter 6

Graph-based ensemble generative modelling for multi-asset forecasting

6.1 Introduction

Building on Chapter 5, which introduced Fin-GAN as a generative model for directional forecasting, we now investigate how to scale such models using a graph-based ensemble framework that accounts for cross-asset relationships. This chapter is based on the methodology developed in [135], where we propose a hybrid framework that combines independent training with performance-based cross-asset generalisation. This involves a three-step process:

1. Train individual generative models on each asset independently.

2. Construct a directed graph whose nodes represent these trained generators, and edges encode relationships between the assets themselves, or the generator capabilities. An edge from i to j represents the flow of information. In our setting, there exists an edge from i to j if the generator trained on asset j is evaluated on data i, i.e. it is used to forecast asset i.

3. A meta-generator for each asset is created by introducing a mixture of probability densities given by the out-neighbouring generators evaluated on the asset under consideration. Since each of the generators encapsulates a probability density of the forecast, the overall forecast probability distribution is given as a linear combination of the probability distributions implied by the neighbouring generators.

Training per-asset and selectively sharing the knowledge based on generalisation performance offers a compromise between pooling all data, and isolating individual models. Furthermore, evaluating each generator on the neighbouring data allows quantifying how well one asset's model generalises to another.

Traditional ensemble learning methods, such as bagging, boosting [16, 32, 33, 80], and random forests [105, 95], combine outputs of pointwise models to improve generalisation. Such approaches have shown to outperform individual models in financial tasks [130]. In these methods, outputs of multiple models are combined in a linear manner. In contrast, we focus on the mixture of the generated probability density distributions, akin to Gaussian Mixture Models [112], enabling richer strategies based on uncertainty-aware statistics. The fact that the meta-generator is based on a directed graph implies that such an approach is capable of capturing asymmetric relationships.

To overcome the issues around homogeneity in universal models, hybrid approaches where some features are shared, and some are stock-specific, have been developed. For example, Fused Encoder Networks [110] enable selective parameter sharing across assets. [13] train to maximise financial performance and group similar stocks together. Other methods [78, 27] pre-train neural networks on assets that are highly correlated or more volatile than the target asset, implicitly encoding domain similarity. Another approach, a Two-Stage Multi-Task algorithm [77], is to first train on pooled data without any regularisation, and then refine model parameters on individual securities, by including an ℓ_2-penalty between the specialised parameter and the global parameter initially learned. There also exist ensemble-based transfer learning frameworks, such as TrEnOS-ELMK [144], which dynamically adjusts weights, but cannot handle probabilistic outputs/generative approaches.

QuantNet [86] also leverages transfer learning for trading signal improvement, trained to maximise financial performance. A special architecture is designed to handle different markets, and encapsulate features present in all, while retaining market-specific information. In their setup, each market has its own encoder-decoder pair, with a shared transfer layer processing the latent representation across all markets. While effective, this architecture is tightly coupled and does not accommodate probabilistic forecasts. Our ensemble framework allows independent training, probabilistic outputs, and interpretable, performance-driven combination weights.

Our method differs from all of the above, and from traditional transfer learning, in the way that we evaluate generalisation capabilities of independently trained generators

across assets to construct a graph encoding transfer learning potential, weights of which are learned through an optimisation procedure based on financial performance.

Our approach shares a number of similarities to the methodology proposed in [141], which decouples asset-specific models and cross-asset relationships, by considering a linear combination of asset-specific forecasts, with the weights given by a kernel estimated at latent representations for each asset. Although both our and the approach of [141] consider a linear combination of asset-specific models, they are estimated on different data. In the setting of [141], each model is evaluated on the data it is trained on, in order to reach a point forecast for a particular asset. On the other hand, we evaluate all asset-specific models on the data we wish to forecast, and learn the weights in an optimisation approach. Although the latent space in [141] would be analogous to the learned weights by the generator, we do not consider them directly, and our approach allows for bi-directional relationships. Similarly, a vector autoregression (VAR) [131] would evaluate each of the generators only on the data it is trained on, and would be more similar to [141], than to our methodology. If we consider all three approaches in a hierarchical manner, VAR and [141] would obtain N first-order pointwise forecasts for each of the stocks. On the other hand, our approach would result in N^2 first-order distributional forecasts.

To learn the graph weights, we formulate a profit-maximisation problem, and restate it as a LASSO regression [129], inducing sparsity. This results in interpretable and computationally efficient weights. In instances where the universe of assets is prohibitively large, sparsity may be induced prior to weight estimation by incorporating domain knowledge (e.g., sector membership or supply chain information), in order to avoid the $O(N^2)$ approach of evaluating all generators on all assets. While our primary objective is not portfolio optimisation, the probabilistic outputs from our meta-generators can be integrated with portfolio construction frameworks such as mean-variance optimisation [98] or risk-aware strategies [146]. The sparse, directed graph structure further supports the construction of uncorrelated signals across assets.

We implement our meta-generator methodology with Fin-GAN [136] as the generative model. We train it on 193 of the S&P500 constituents, individually, i.e. we train one Fin-GAN per asset. By evaluating each generator on every asset, we identify which assets have higher transfer learning potential. Our findings suggest that some stocks perform consistently well, despite the training data stemming from diverse sectors and exhibiting low correlation. We evaluate our LASSO-based approach against Ridge regression [71], PnL and Sharpe Ratio-induced graphs, while also comparing it with the baseline scenarios in which each stock is evaluated solely on its own generator.

Lastly, we benchmark the aforementioned approaches against the setup where the weights of the meta-graph are implied by the correlation structure of the returns. Our results demonstrate that the LASSO-based approach significantly outperforms these benchmarks, efficiently leveraging cross-sectional information to enhance performance. Additionally, we show that soft classification results in higher Sharpe Ratios, compared to trading based solely on the direction, which fails to account for intensity, i.e. the magnitude of the forecast.

In sum, this chapter contributes a scalable and interpretable method for cross-asset generative ensemble learning, with applications in directional forecasting, distributional prediction, and risk-aware decision-making. Our method departs from purely universal or purely asset-specific modelling, and instead creates a new middle ground: one that adapts to the structure of information transfer in financial markets.

Outline. We derive our methodology in Section 6.2, and demonstrate its performance on Fin-GAN in Section 6.3. Section 6.4 studies the resulting graph.

6.2 Methodology

6.2.1 Multi-asset forecasting

Suppose we are interested in forecasting returns of N assets at a frequency Δt. Let x_t^i be the returns of the asset i between times t and $t - \Delta t$, and let a_t^i be the market information on asset i at time t, used as features to forecast the next value of the returns time series, $x_{t+\Delta t}^i$, the same notation as in Chapters 1 and 5.

Suppose we have a function g_i (generative model) trained on the historical data of asset i, taking as input a_t^i and i.i.d. (typically Gaussian) noise z_t^i, and outputting a forecast of $x_{t+\Delta t}^i$. That is,

$$g_i(a_t^i, z_t^i) = \hat{x}_{t+\Delta t}^i(z_t^i) \tag{6.1}$$

is a sample from the estimated distribution of $x_{t+\Delta t}^i$ given a_t.

If N is large, there might not be enough data samples to learn a single function g taking inputs $a_1, \ldots, a_N$, noise Z, and outputting forecasts for all assets. Our aim is to scale up generative models built for forecasting returns (such as Fin-GAN [136]) to a multi-asset setting, while circumventing this issue.

6.2.2 Meta-generators

We are interested in efficiently leveraging cross-sectional information to derive more robust performance. Given that we have N generators g_1, g_2,...,g_N, with generator i trained on data of the asset i, we can use the information that each generator will have when evaluated on the data it is not necessarily traded on. Let $w_{i,j}$ be the edge from i to j, corresponding to relationship between the data i and the generator j. An illustration is provided in Figure 6.1.

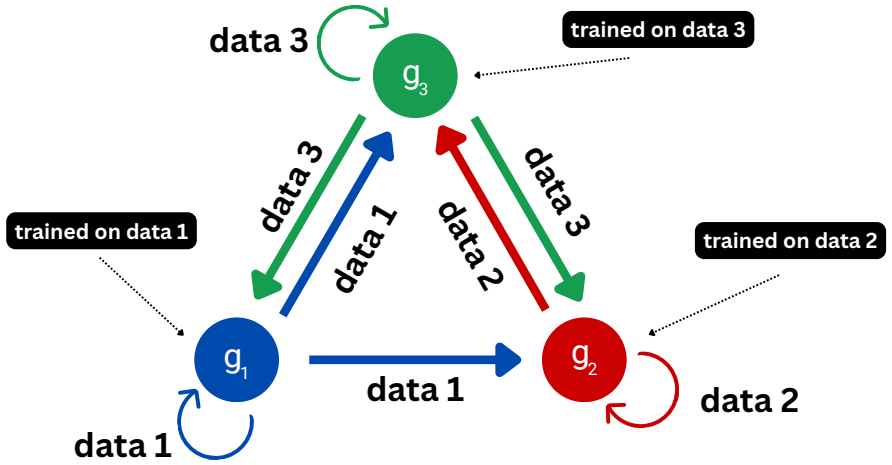


Figure 6.1: Meta-graph illustration. Each node i represents a generator g_i trained on data of asset i. An edge from i to j encodes the effectiveness of the generator j at forecasting data i.

There are two main ways in which *ensemble* learning can be performed on such a graph to create a meta-generator for each asset.

1. An output for the meta-graph G_i for asset i at time t, and noise Z is given by a linear combination of each $g_j(a_t^i, Z)$, with weights $w_{i,j}$.

2. The probability density of the meta-graph G_i for asset i at time t is given as a mixture of densities of $\text{sign}(w_{i,j})g_j(a_t^i, Z)$, with the mixture weights $|w_{i,j}|$.

Working with probability densities rather than the raw outputs offers a plethora of benefits, due to the fact that the expectation under the probability distribution intrinsic to the meta-generator is a superposition of the individual expectations. Denote by $\mathbb{P}_t^{i,j}$ is the distribution intrinsic to the generator g_j (reflected if $w_{i,j} < 0$), conditional on a_t^i (for asset i). Consider

$$X^{i,j}(a_t^i) = \text{sign}(w_{i,j})g_j(a_t^i, Z), \quad Z \sim \mathcal{N}(0,1),$$

so that $\mathbb{P}_t^{i,j}$ the probability distribution of $X^{i,j}(a_t^i)$. That is,

$$X_t^{i,j}(a_t^i) \sim \mathbb{P}_t^{i,j}.$$

Denote by $\mathbb{P}_t^i$ the mixture of densities

$$\mathbb{P}_t^i = \sum_{j=1}^{N} |w_{i,j}| \mathbb{P}_t^{i,j}, \quad \sum_{j=1}^{N} |w_{i,j}| = 1. \tag{6.2}$$

In fact, $\mathbb{P}_t^i$ is the resulting probability distribution of the meta-generator $G_i(a_t^i, Z)$

$$G_i(a_t^i, Z) \sim \mathbb{P}_t^i. \tag{6.3}$$

Then, sampling from $\mathbb{P}^i$ is equivalent to sampling from $\mathbb{P}^{i,j}$ with probability $|w_{i,j}|$. In other words, for fixed data i, first a generator j is selected with probability $|w_{i,j}|$, and then a sample $\text{sign}(w_{i,j}) g_j(a_t, Z)$ is obtained.

Opting for a mixture of densities rather than superimposing direct outputs with fixed noise results in the linearisation of expectations. However, care needs to be taken with respect to the sign, which is included in the weight in the case of anti-symmetric functions.

Although we perform our analysis by opting for expected sign as the trade size, our setting is compatible with any trade size which is given as a conditional expectation of an anti-symmetric function. Denote by

$$s_i(a) = \mathbb{E}\left[\text{sign}(g_i(a, Z))\right] \tag{6.4}$$

the trade given by the forecast of the generator i given data a and noise Z, with the usual choice $Z \sim \mathcal{N}(0, 1)$. Now, for a 'node' i and weights $\{w_{i,j}\}$ such that $\sum_{j=1}^{N} |w_{i,j}| = 1$, define the aggregated bet size by

$$h_i(\cdot) = \sum_{j=1}^{N} w_{i,j} s_j(\cdot), \tag{6.5}$$

which is the bet size of the meta-generator G_i for asset i. Hence, only the trade sizes obtained from the generators need to be considered, and not all possible generated scenarios, thus optimising memory.

6.2.3 Learning the weights

The adjacency matrix given by the weights could be learned via machine learning techniques, such as Graph Neural Networks [119]. However, such an approach would be computationally heavy, and lack interpretability. An alternative is to learn the weights via a PnL-maximisation approach. We study two approaches

- regularised PnL maximisation;
- casting the PnL maximisation problem as a linear regression problem.

The first approach would lead to solutions with potentially undesirable properties, where either most (or all) generators will be incorporated into the final vote for a particular asset, or only one. However, the second approach, in the case of LASSO regression, leads to an automatic neighbour selection based on transferability of information and financial performance, and induces sparsity, while still allowing for robustness stemming from multiple generators.

6.2.4 PnL maximisation

We are interested in optimising a *global* strategy. For simplicity, the initial investment nominal is the same across all assets. That is, each portfolio weight corresponds to a percentage of the same dollar value. Further contributions can be made by considering a more complex portfolio optimisation. We aim to maximise

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[h_i(a_t^i)X_{t+\Delta t}^i\right], \tag{6.6}$$

where $(a_t^i, X_{t+\Delta t}^i)$ are the representations of the condition window and the return of asset i as random variables (from time t to $t+\Delta t$).

Remark 11. *Equation* (6.6) *is equivalent to the expected return of the generated strategy for an asset i, with i sampled uniformly at random from* $\{1,\dots,N\}$.

Assuming ergodicity, we replace the expectation in (6.6) with an average over a long time period $\mathbb{T}$, which is either the training or the validation set, or both combined:

$$PnL(\mathbb{T},\boldsymbol{w}) = \frac{1}{N|\mathbb{T}|}\sum_{i=1}^{N}\sum_{t\in\mathbb{T}}h_i(a_t^i)x_{t+\Delta t}^i = \frac{1}{N|\mathbb{T}|}\sum_{t\in\mathbb{T}}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{i,j}s_j(a_t^i)x_{t+\Delta t}^i. \tag{6.7}$$

As we average the PnLs across assets, and we do not require $w_{i,j} = w_{j,i}$, we focus on each asset individually. Without the factor $\frac{1}{N|\mathbb{T}|}$, the expression in Equation (6.7) corresponds to the profit and loss of a trading strategy in which, at time t, the notional amount $|h_i(a_t^i)|$ is invested in asset i. Over the time period $\mathbb{T}$, the average PnL obtained by trading asset i using information from the generators $\{g_j\}_{j=1,\dots,N}$ and weights $\boldsymbol{w} = \{w_{i,j}\}_{i,j\in\{1,\dots,N\}}$ is

$$\frac{1}{|\mathbb{T}|}\sum_{t\in\mathbb{T}}\sum_{j=1}^{N} w_{i,j}s_j(a_t^i)x_{t+\Delta t}^i, \tag{6.8}$$

where $|\mathbb{T}|$ is the size of the time window $\mathbb{T}$. Maximising the PnL for each asset i is equivalent to the following optimisation problem

$$\begin{aligned} &\max_{\boldsymbol{w}_i\in\mathbb{R}^N} \sum_{t\in\mathbb{T}}\sum_{j=1}^{N} w_{i,j}s_j(a_t^i)x_{t+\Delta t}^i \\ &\text{s.t.} \sum_{j=1}^{N} |w_{i,j}| = 1, \end{aligned} \tag{6.9}$$

where $\boldsymbol{w}_i$ is the $i-$th row of the weight matrix $\{w_{i,j}\}_{i,j\in\{1,\dots,N\}}$.

To simplify notation, for each $i \in \{1,\dots,N\}$ and fixed $\mathbb{T}$, define the cumulative PnL over a period $\mathbb{T}$ as

$$c_{i,j} = \sum_{t\in\mathbb{T}} s_j(a_t^i)x_{t+\Delta t}^i. \tag{6.10}$$

The optimisation problem (6.9) has a closed-form solution. Unfortunately, its solution is not robust, since only a single generator is selected if there exists a generator producing a non-zero average PnL, i.e. if there is a $j \in \{1,\dots,N\}$ such that $c_{i,j} \neq 0$.

Proposition 3. *Fix $i \in \{1,\dots,N\}$ and suppose that there exists $j \in \{1,\dots,N\}$ such that $c_{i,j} \neq 0$. Let $\mathcal{J}^* = \{j \in \{1,\dots,N\} : |c_{i,j}| \geq |c_k^i| \quad \forall k \in \{1,\dots,N\}\}$ be the index set of the generators with the strongest signals for asset i.*

1. *If $|\mathcal{J}^*| = 1$, i.e. if there is a unique maxima of $|c_{i,j}|$,*

$$j^* = argmax_j \left|c_{i,j}\right|,$$

the solution to (6.9) *is given by*

$$w_{i,j} = \begin{cases} 0, & \text{for } j \neq j^*, \\ \text{sign}\left(c_{i,j}\right), & \text{for } j = j^*. \end{cases} \tag{6.11}$$

2. *If $|\mathcal{J}^*| > 1$, then the set of solutions is the convex hull of the signed unit vectors supported on the set of the strongest generators $\mathcal{J}^*$:*

$$w_{i,j} = \begin{cases} \alpha_j \operatorname{sign}(c_{i,j}) & \text{if } j \in \mathcal{J}^*, \\ 0 & \text{otherwise,} \end{cases} \quad \text{with } \alpha_j \geq 0, \quad \sum_{j \in \mathcal{J}^*} \alpha_j = 1.$$

Proof. For fixed $i \in \{1, \ldots, N\}$ the problem (6.9) is

$$\begin{aligned} &\max_{\boldsymbol{w}_i \in \mathbb{R}^N} \sum_{j=1}^{N} w_{i,j} c_{i,j}, \\ &\text{s.t.} \sum_{j=1}^{N} |w_{i,j}| = 1. \end{aligned} \tag{6.12}$$

The problem (6.12) is convex and coordinate separable, so we can study contributions from each coordinate j to the objective $\sum_{j=1}^{N} w_{i,j} c_{i,j}$.

Firstly, notice that the contribution from coordinate j is non-negative when $\operatorname{sign}(w_{i,j}) = \operatorname{sign}(c_{i,j})$. If $c_{i,j}$ and the weight $w_{i,j}$ are of the opposite sign, then $w_{i,j} c_{i,j} < 0$, so $\sum_{k=1}^{N} w_{i,k} c_k^i \leq \sum_{k \neq j} w_{i,k} c_{i,k}$. Hence, the maximum value of (6.12) is achieved for $\operatorname{sign}(w_{i,j}) = \operatorname{sign}(c_{i,j})$. If $c_{i,j} = 0$, then $w_{i,j} c_{i,j} = 0$, and $\sum_{k=1}^{N} w_{i,k} c_{i,k} = \sum_{k \neq j} w_{i,k} c_{i,k}$. Let k be such that $c_{i,k} \neq 0$. Then, as $w_{i,k} c_{i,k} > 0$,

$$\operatorname{sign}(c_{i,k}) |w_{i,j}| c_{i,k} \sum_{l \neq j} w_{i,l} c_{i,l} \geq \sum_{l=1}^{N} w_{i,l} c_{i,l}.$$

Hence, since $\sum_{l=1}^{N} |w_{i,l}| = 1$, when $c_{i,j} = 0$ it is optimal to set $w_{i,j} = 0$.

We can now assume that $\operatorname{sign}(w_{i,j}) = \operatorname{sign}(c_{i,j})$, and introduce a new variable

$$u_j = w_{i,j} \operatorname{sign}(c_{i,j}) = w_{i,j} \operatorname{sign}(w_{i,j}) = |w_{i,j}| \geq 0.$$

Since $\operatorname{sign}(w_{i,j}) = \operatorname{sign}(c_{i,j})$, and $\operatorname{sign}(c_j)^2 = 1$ when $c_j \neq 0$,

$$w_{i,j} = u_j \operatorname{sign}(c_{i,j})$$

As $\operatorname{sign}(0) = 0$, and $w_{i,j} = 0$ when $c_{i,j} = 0$, the above equalities also hold for $c_{i,j} = 0$.

We now re-write (6.12) in terms of u_j:

$$\begin{aligned} &\max_{u \in \mathbb{R}^N} \sum_{j=1}^{N} u_j |c_{i,j}|, \\ &\text{s.t.} \sum_{j=1}^{N} u_j = 1, \\ &\text{and } \forall j \in \{1, \ldots, N\} \quad u_j \geq 0. \end{aligned} \tag{6.13}$$

Suppose that $|\mathcal{J}^*| = 1$ and let $j^* = \text{argmax}_j |c_{i,j}|$. Then, $|c_{i,j^*}| \geq |c_{i,j}|$ for all j, and since $u_j \geq 0$, then $u_j |c_{i,j^*}| \geq u_j |c_{i,j}|$. Hence, since $\sum_{j=1}^N u_j = 1$,

$$|c_{i,j^*}| = \sum_j u_j |c_{i,j^*}| \geq \sum_j u_j |c_{i,j}|.$$

Therefore, the optimal solution is given by $u_j = 0$ for $j \neq j^*$, and $u_{j^*} = 1$. This directly translates to

$$w_{i,j} = \begin{cases} 0, & \text{for } j \neq j^*, \\ \text{sign}\left(\sum_{t \in \mathbb{T}} s_{j^*}(a_t^i) x_{t+\Delta t}^i\right), & \text{for } j = j^*. \end{cases}$$

However, when $|\mathcal{J}^*| \geq 0$, let $\alpha \in \mathbb{R}^N$ such that

$$\alpha_j = 0 \quad \forall j \notin \mathcal{J}^*,$$

$$\alpha_j \geq 0 \quad \forall j \in \mathcal{J}^*,$$

and

$$\sum_{j=1}^N \alpha_j = 1.$$

Then, for any $w_{i,\cdot} \in \mathbb{R}^N$,

$$\forall j \in \mathcal{J}^* \quad \text{sign}(c_{i,j}) c_{i,j} = \sum_{k=1}^N (\alpha_k \text{sign}(c_{i,k})) c_{i,k} \geq \sum_{k=1}^N w_{i,k} c_{i,k},$$

Hence, although (6.12) does not have a unique solution, the values of $w_{i,\cdot} \in \mathbb{R}^N$ which maximise $\sum_{j=1}^N w_{i,j} c_{i,j}$ subject to $\sum_{j=1}^N |w_{i,j}| = 1$ are given by

$$w_{i,j} = \begin{cases} 0, & \text{for } j \notin \mathcal{J}^*, \\ \alpha_j \text{sign}\left(c_{i,j}\right), & \text{for } j \in \mathcal{J}^*, \end{cases} \qquad \alpha_j \geq 0 \quad \forall j \in \mathcal{J}^*, \qquad \sum_{j \in \mathcal{J}^*} \alpha_j = 1. \quad (6.14)$$

□

The solution to (6.9) does not offer any diversification and may result in overfitting. Hence we consider other possible formulations.

We now show that the unconstrained relaxation of problem (6.9) using the ℓ_1 penalty has the same solution as the original problem when there exists a dominant generator. However, it still remains non-robust due to the absence of diversification.

To transform the problem into an unconstrained formulation, introduce auxiliary variables $\lambda_{i,j} \in \mathbb{R}$, and define

$$w_{i,j} = \frac{\lambda_{i,j}}{\sum_{k=1}^N |\lambda_{i,k}|}.$$

Clearly, the vector $\boldsymbol{\lambda}_i$ is only identifiable up to a positive scalar multiple. Substituting this into the original objective, the cumulative PnL becomes

$$c_{i,j} = \sum_{j=1}^{N} w_{i,j} s_j(a_t^i) x_{t+\Delta t}^i = \frac{1}{\sum_{k=1}^{N} |\lambda_{i,k}|} \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i.$$

Thus, the constrained problem (6.9) becomes:

$$\max_{\boldsymbol{\lambda}_i \in \mathbb{R}^N} \frac{1}{\sum_{k=1}^{N} |\lambda_{i,k}|} \sum_{t \in \mathbb{T}} \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i. \tag{6.15}$$

This objective is homogeneous of degree zero, non-convex, and non-differentiable due to the ℓ_1-norm in the denominator. To render the problem tractable, we consider the regularised objective

$$\max_{\boldsymbol{\lambda}_i \in \mathbb{R}^N} \frac{1}{T} \sum_{t \in \mathbb{T}} \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i - \eta \sum_{k=1}^{N} |\lambda_{i,k}|. \tag{6.16}$$

where $\eta > 0$ is a regularisation parameter that penalises the ℓ_1-norm of $\boldsymbol{\lambda}_i$. Despite being unconstrained, the problem remains non-smooth but is coordinate-separable.

Lemma 1 (Signal thresholding)**.** *The optimisation problem* (6.16) *admits only sparse solutions.*

1. *If $|c_{i,j}| \leq \eta$, then $\lambda_{i,j} = 0$ in all optimal solutions.*
2. *If there exists $j \in 1, \ldots, N$ such that $|c_{i,j}| > \eta$, then $\sup_{\boldsymbol{\lambda}_i} f(\boldsymbol{\lambda}_i) = +\infty$, i.e., the objective in* (6.16) *is unbounded above.*
3. *Otherwise, if $|c_{i,j}| \leq \eta$ for all j, the unique optimal solution is $\boldsymbol{\lambda}_i = 0$, and the supremum is zero.*

Proof. The objective function can be written as

$$\max_{\boldsymbol{\lambda}_i} f(\boldsymbol{\lambda}_i) = \boldsymbol{\lambda}_i^T c - \eta ||\boldsymbol{\lambda}_i||_1.$$

It is coordinate-separable, so we consider each coordinate independently. Hence, we consider

$$\max_{\lambda_{i,j}} (c_j \text{sgn}(\lambda_{i,j}) - \eta) |\lambda_{i,j}|$$

Define $g_j(\lambda_{i,j}) = c_{i,j}\lambda_{i,j} - \eta|\lambda_{i,j}|$. This function is unbounded above if and only if $|c_{i,j}| > \eta$, in which case $g_j(\lambda_{i,j}) \to +\infty$ as $\lambda_{i,j} \to \infty$ if $c_{i,j} > 0$ and as $\lambda_{i,j} \to -\infty$ if $c_{i,j} < 0$. If $|c_{i,j}| \leq \eta$, then $g_j(\lambda_{i,j}) \leq 0$ for all $\lambda_{i,j}$, and the maximum is achieved at $\lambda_{i,j} = 0$. The result follows by combining the coordinate-wise arguments. □

Lemma 2 (Uniqueness of the normalised solution)**.** *The relaxed problem* (6.16) *has a unique solution for the normalised weights $w_{i,j}$ if and only if there exists a unique index $j^* \in 1, \ldots, N$ such that $|c_{i,j^*}| > \eta$.*

Proof. From Lemma 1, only coordinates j with $|c_{i,j}| > \eta$ can have non-zero $\lambda_{i,j}$ in an optimal solution. If multiple such indices exist, say $j \in \mathcal{J}$ with $|\mathcal{J}| > 1$, then the optimal $\boldsymbol{\lambda}_i$ lies in an unbounded cone supported on $\mathcal{J}$. Hence, there are infinitely many optimal solutions for $\boldsymbol{\lambda}_i$ differing by direction and scale. However, the normalised vector $\boldsymbol{w}_i$ depends only on the relative magnitudes and signs of the non-zero $\lambda_{i,j}$. Therefore, unless a unique coordinate j dominates (i.e., $|c_{i,j}| > \eta$ and $|c_{i,k}| \leq \eta$ for all $k \neq j$), the resulting $\boldsymbol{w}_i$ is not uniquely defined. If such a unique j^* exists, then $\lambda_{i,j^*} \to \infty \cdot \operatorname{sign}(c_{i,j^*})$, all other coordinates vanish, and

$$w_{i,j} = \begin{cases} 0, & \text{for } j \neq j^*, \\ \operatorname{sign}(c_{i,j}), & \text{for } j = j^*, \end{cases}$$

which is unique. □

Note that in the case of non-unique solutions to (6.9) and (6.16), the sets of maximisers are not the same. Without the ℓ_1-relaxation, only the generators j with $|c_{i,j}| \geq |c_{i,k}|$ for all $k \in \{1, \ldots, N\}$ are considered. However, in the case of (6.16), all j with $|c_{i,j}| \geq \eta$ are included.

The only way that the relaxed problem (6.16) can have a unique solution is if the threshold η lies strictly between the strongest and second strongest signal provided by the generators, and if there is a unique generator j with the highest $|c_{i,j}|$. In this instance, the unique solution coincides with that of the unconstrained problem (6.15).

However, we can relax the problem (6.16) further by replacing the regularisation term with an ℓ_2-penalty. This results in an the alternative formulation, given by

$$\max_{\boldsymbol{\lambda}_i \in \mathbb{R}^N} \sum_{t \in \mathbb{T}} \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i - \eta \sum_{j=1}^{N} (\lambda_{i,j})^2. \tag{6.17}$$

The solution to (6.17) is unique and intuitive: the optimal weight $w_{i,j}$ is proportional to the PnL that the generator g_j achieves on the data of asset i. However, this formulation has non-zero weights for all generators, making the solution dense rather than sparse.

Proposition 4. *The weights that maximise* (6.17) *are given by*

$$w_{i,j}^* = \frac{\sum_{t \in \mathbb{T}} s_j(a_t^i) x_{t+\Delta t}^i}{\sum_{k=1}^{N} \left| \sum_{t \in \mathbb{T}} s_k(a_t^i) x_{t+\Delta t}^i \right|}. \tag{6.18}$$

Proof. The objective in (6.17) is strictly concave and differentiable. Taking the derivative with respect to $\lambda_{i,j}$ and setting it to zero gives:

$$\frac{\partial}{\partial \lambda_{i,j}} \left(\sum_{t \in \mathbb{T}} \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i - \eta \sum_{j=1}^{N} \lambda_{i,j}^2 \right) = \sum_{t \in \mathbb{T}} s_j(a_t^i) x_{t+\Delta t}^i - 2\eta \lambda_{i,j} = 0.$$

Solving for $\lambda_{i,j}$ leads to

$$\lambda_{i,j}^* = \frac{1}{2\eta} \sum_{t \in \mathbb{T}} s_j(a_t^i) x_{t+\Delta t}^i.$$

To convert this to normalised weights, we compute:

$$w_{i,j}^* = \frac{\lambda_{i,j}^*}{\sum_{k=1}^{N} |\lambda_{i,k}^*|} = \frac{\sum_{t \in \mathbb{T}} s_j(a_t^i) x_{t+\Delta t}^i}{\sum_{k=1}^{N} \left| \sum_{t \in \mathbb{T}} s_k(a_t^i) x_{t+\Delta t}^i \right|}.$$

□

6.2.5 Regression

Instead of maximising the PnL in (6.16), one can minimise the distance to the highest possible PnL, i.e., aim to match the sign of the return (directionality). Such an approach results in an objective of the form

$$\min_{\boldsymbol{\lambda}_i \in \mathbb{R}^N} \frac{1}{2|\mathbb{T}|} \sum_{t \in \mathbb{T}} \left(|x_{t+\Delta t}^i| - \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i \right)^2. \tag{6.19}$$

However, since the same data a_t^i is used to evaluate all generators j, OLS regression in (6.19) might not be stable due to potential singularity of the Gram matrix. Furthermore, we wish to induce sparsity, in order to reduce the computational cost and prevent overfitting.

We regularise by including an ℓ_1 penalty to induce sparsity, resulting in a LASSO regression [129, 65] of the form

$$\min_{\boldsymbol{\lambda}_i \in \mathbb{R}^N} \frac{1}{2|\mathbb{T}|} \sum_{t \in \mathbb{T}} \left(|x_{t+\Delta t}^i| - \sum_{j=1}^{N} \lambda_{i,j} s_j(a_t^i) x_{t+\Delta t}^i \right)^2 + \eta \sum_{k=1}^{N} |\lambda_{i,k}|. \tag{6.20}$$

The resulting weights are then normalised as

$$w_{i,j} = \frac{\lambda_{i,j}}{\sum_{k=1}^{N} |\lambda_{i,k}|}.$$

Such normalisation merely rescales the optimal strategy derived from (6.20). Python libraries such as `scikit-learn` can be employed to solve (6.20). Moreover, due

to the ℓ_1-norm penalty, this formulation naturally performs automatic 'neighbour' selection. LASSO regression induces sparsity, which reduces computational intensity at inference time. Since fewer generators are selected, variance of the meta-generator PnL is typically reduced, potentially improving portfolio performance. However, if a particular generator is over-used, the variance of final outputs across assets might still be high.

To enhance computational efficiency, most weights $w_{i,j}$ can be manually set to zero using domain knowledge. Other constraints can be straightforwardly incorporated by modifying (6.20).

Although LASSO regression (6.20) offers the most desirable properties under all formulations considered so far, as another benchmark we also consider an ℓ_2-regularised version of (6.19), i.e. Ridge regression [71]. Such formulation leads to

$$\min_{\boldsymbol{\lambda}_i} \frac{1}{2|\mathbb{T}|} \sum_{t\in\mathbb{T}} \left(|x^i_{t+\Delta t}| - \sum_{j=1}^{N} \lambda_{i,j} s_j(a^i_t) x^i_{t+\Delta t} \right)^2 + \eta \sum_{k=1}^{N} (\lambda_{i,k})^2. \tag{6.21}$$

Unlike LASSO (6.20), Ridge regression (6.21) admits a closed-form solution. However, it does not yield sparse weights, potentially resulting in highly correlated PnLs across different assets, since all generators are used for all assets. At inference time, Ridge regression requires evaluation of all generators on all assets.

In both LASSO and Ridge settings, additional constraints may be imposed to reflect interactions between rows of the weight matrix $\boldsymbol{w}$, effectively enabling joint portfolio optimisation.

Overall, the LASSO formulation (6.20) offers the greatest flexibility: sparsity leads to better memory and compute efficiency, faster inference, and reduced correlation, while maintaining a diversified signal aggregation strategy. The algorithm used to develop meta-generators via LASSO regression is summarised in Algorithm 3.

Remark 12. *The factor* $\frac{1}{2}$ *is included to conform to the convention used in the* `scikit-learn` *package.*

Algorithm 3 Meta-generator construction and weight learning

Input:

- Market data $\{a_t^i, x_{t+\Delta t}^i\}_{t\in\mathbb{T}}$ for each asset $i \in \{1, \dots, N\}$;
- Training/validation split $\mathbb{T} = \mathbb{T}_t \cup \mathbb{T}_v$;
- Number of samples B for expected sign estimation;
- Regularisation parameter η.

Output: Meta-graph edge weights $\{w_{i,j}\}_{i,j\in\{1,\dots,N\}}$

Step 1: *Train individual generators.*

for each asset i **do**

Train generator g_i on $\{(a_t^i, x_{t+\Delta t}^i)\}_{t\in\mathbb{T}_t}$

end for

Step 2: *Cross-evaluate generators on validation data.*

for each asset i **do**

for each generator g_j (as allowed) **do**

Sample B i.i.d. noise terms $\{Z_t^k\}_{k=1}^B$ from $\mathcal{N}(0,1)$

Compute expected sign estimate:

$$s_j(a_t^i) = \frac{1}{B}\sum_{k=1}^{B}\left[\mathbf{1}_{g_j(a_t^i, Z_t^k)>0} - \mathbf{1}_{g_j(a_t^i, Z_t^k)<0}\right]$$

end for

end for

Step 3: *Assign weights and define distributions.*

Assign weight $w_{i,j}$ to each asset-generator pair

Define $\mathbb{P}_t^{i,j}$ as the distribution of $\text{sign}(w_{i,j})g_j(a_t^i, Z)$

Step 4: *Construct meta-generator.*

Define mixture distribution:

$$\mathbb{P}_t^i = \sum_{j=1}^{N} |w_{i,j}| \mathbb{P}_t^{i,j}$$

To sample: choose j with probability $|w_{i,j}|$, then evaluate $\text{sign}(w_{i,j})g_j(a_t^i, Z)$

Step 5: *Learn weights via LASSO regression.*

For each i, solve:

$$\min_{\boldsymbol{\lambda}_i} \frac{1}{2|\mathbb{T}_v|}\sum_{t\in\mathbb{T}_v}\left(|x_{t+\Delta t}^i| - \sum_{j=1}^{N}\lambda_{i,j}s_j(a_t^i)x_{t+\Delta t}^i\right)^2 + \eta\sum_{j=1}^{N}|\lambda_{i,j}|$$

Step 6: *Normalise the weights.*

Set $w_{i,j} = \dfrac{\lambda_{i,j}}{\sum_{j=1}^{N}|\lambda_{i,j}|}$

Return: $\{w_{i,j}\}_{i,j\in\{1,\dots,N\}}$

6.3 Extending Fin-GAN to a multi-asset setting

We apply our methodology to Fin-GAN [136], developed in Chapter 5. We train each Fin-GAN generator current S& P500 members for which we have data available from the start of 2000, which is 193 of them. The data and the set-up is the same as in Chapter 5. The sector ETFs tickers and the corresponding stock tickers are listed in Table 6.1. We compute the expected sign given 100 samples.

Sector	Ticker	Stock tickers
Consumer Discretionary	XLY	AMZN, AZO, BBY, BWA, DHI, DRI, HAS, HD, LOW, MAR, MCD, MHK, NWL, NKE, PHM, PVH, RL, RCL, TJX, VFC, WHR, YUM
Consumer Staples	XLP	ADM, MO, CPB, CHD, CLX, KO, CL, EL, GIS, HSY, HRL, K, KMB, KR, PEP, PG, SYY, TSN, WMT
Energy	XLE	APA, DVN, EOG, XOM, HAL, MRO, OXY, OKE, PXD, SLB, VLO, WMB
Financial	XLF	AFL, AIG, ALL, AXP, BAC, BLK, BK, BRO, COF, CMA, RE, FDS, BEN, AJG, GS, HIG, LNC, MMC, PNC, PGR, RJF, STT, SIVB, WFC, ZION
Health Care	XLV	ABT, A, AMGN, BAX, BDX, TECH, BSX, BMY, CI, COO, CVS, DHR, HUM, INCY, JNJ, LH, LLY, MCK, MDT, MRK, MTD, PKI, PFE, RMD, STE, SYK, TFX, TMO, UNH, UHS, WAT, WST
Industrials	XLI	MMM, ALK, AME, BA, CAT, CSX, DE, DOV, ETN, EMR, EFX, FDX, GE, GD, GWW, HON, IEX, ITW, LMT, MAS, NDSN, NSC, NOC, PCAR, PH, PWR, RSG, RHI, ROK, ROL, ROP, SNA, LUV, SWK, TDY, TXT, UNP, UPS, URI, WAB
Technology	XLK	AKAM, AMD, APH, ADI, AMAT, ADSK, GLW, IBM, MU, TER, TYL, WDC
Materials	XLB	APD, ALB, AVY, EMN, ECL, FMC, IP, NEM, NUE, PPG, SEE, SHW, VMC
Utilities	XLU	AES, AEE, AEP, ATO, CMS, ED, D, DTE, DUK, EIX, ETR, FE, PNW, PPL, PEG, SRE, SO, WEC

Table 6.1: ETF tickers and their corresponding stock tickers

We investigate the approaches discussed in Section 6.2.4, as well as some additional benchmarks.

- **Identity**: corresponds to the adjacency matrix being the identity, i.e. there is no information sharing between different generatos.
- **Correlation-based**: the adjacency matrix is implied by the correlation matrix of the asset returns, calculated over the training and the validation set. Since each row is normalised such that the absolute values sum to one, the resulting adjacency matrix is not the correlation itself.
- **SR-based**: each weight $w_{i,j}$ is proportional to the Sharpe Ratio achieved by generator j on data i over the validation set.
- **PnL-thr**: the weights $w_{i,j}$ are proportional to the PnL achieved by generator j on data i over the validation set, for *strong enough* PnLs. That is, the weight $w_{i,j}$ is non-zero if the absolute cumulative PnL is above a certain threshold η.
- **PnL-max**: the weights are given by the solution of Proposition 3. That is, this represents the linear case (maximising PnL), with the ℓ_1-penalty (6.16), in the unique solution case. For each data set (asset), only the generator with the strongest signal is used.
- **PnL-based**: weights given by Proposition 4, i.e. the weights maximising the PnL with the ℓ_2-penalty (6.17) are proportional to the PnLs achieved on the validation set.

- **LASSO**: weights given by LASSO regression (6.20).
- **Ridge**: weights given by Ridge regression (6.21).

For all of the methods requiring hyperparameter tuning, i.e. PnL-thr, LASSO, and Ridge, we performed a small hyperparameter search and chose the value which achieved the highest overall Sharpe Ratio on the validation set, which in this instance would correspond to an in-sample fit.

In the PnL-thr case, the thresholding parameters under consideration were $\eta \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$. For the value of $\eta = 0.5$, there were nodes i for which all $w_{i,j}$ were set to zero. The thresholding parameter η for which the best performance over the validation set was achieved (in terms of Sharpe Ratio) was $\eta = 0.2$, which is the value we used. High penalty values would also induce too much sparsity in the LASSO case, so the search gird we opted for was $\eta \in \{0.001,0.0001, 0.00001, 0.000001\}$. The selected value was 0.000001. Similarly, the grid search for Ridge regression was $\eta \in \{0.001,0.0001, 0.00001, 0.000001\}$, with the optimal hyperparameter choice being once again 0.000001. We first compare financial performance of the methods under consideration, and then we study the best-performing resulting graph, which is based on LASSO regression (6.20). In the LASSO instance, if all weights are set to zero by the algorithm, then we set $w_{i,i} = 1$ for the asset i in question. However, with $\eta = 0.000001$, this never occured.

Remark 13. *It is possible to consider Markowitz-inspired weights [98] instead of the PnL-based methodology, corresponding to the ℓ_2-regularisation. Instead of $\eta \sum_{k=1}^{N} (\lambda_{i,k})^2$ in* (6.17)*, one could consider $\eta \lambda_i^T \Sigma_i \lambda_i^2$, where Σ_i is the covariance matrix of the corresponding PnLs produced by different generators for the same asset i. However, since the generators are evaluated on the same data, the correlation between their outputs was high, resulting in singular Σ_i.*

6.3.1 Performance analysis

Before comparing the performance over the test set, we first check whether the relationship between the performance of the generators on each of the data sets aligns with the correlation between the returns of the assets corresponding to the generator and to the data under consideration. Figure 6.2 shows that the correlation matrix of returns (over the training and validation set) indicates strong correlation between the ETF-excess returns of tickers corresponding to the same industry sector. However, the same is not true for the Sharpe Ratio performance in Figure 6.3 on the validation

set, where each row corresponds to forecasting data, and each column to a generator trained on specific data. Interestingly, the Sharpe Ratio matrix is not diagonal and there is clear indication that some tickers are easier to forecast than others, or have higher transfer learning potential. In the universality setting discussed in Chapter 5, we have shown that pooling data from the same sector results in good performance on unseen data from that particular sector. However, in this setting, we are observing transfer learning potential. In fact, 126 out of 193 tickers have positive Sharpe Ratios (validation set) on at least 50% of the generators, showing a promising starting point for a meta-generator.

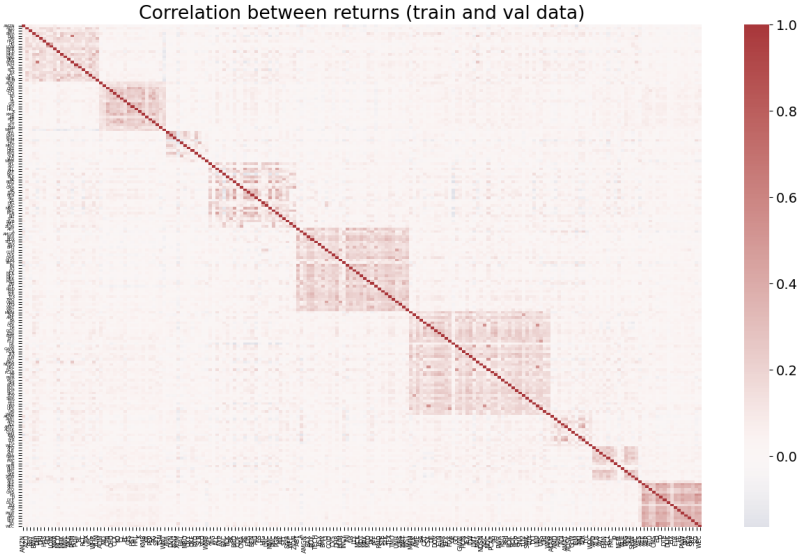


Figure 6.2: Pearson correlation coefficient between the ETF-excess returns over the training and validation set. Unsurprisingly, we observe clusters corresponding to industry sectors. The tickers are sorted alphabetically first by sector, and then by the ticker.

The most prominent transfer learning case is TDY, for which the median Sharpe Ratio achieved is 1.29. Training a generator on 147 out of 193 data sets results in good performance on TDY data (XLI sector). Despite the highest median Sharpe Ratio being achieved on TDY data, the highest number of well-performing generators is RMD (XLV), where 163 different generators result in positive PnLs. The median SR in this case is 0.84. TMO (XLV) and SHW (XLB) also have a higher number of positive generators compared to TDY (149 and 153, respectively).

On the other hand, there are tickers for which most of the generators result in negative performance. The lowest number of positive Sharpe Ratios is achieved for CHD (XLP), where only 41 of the Sharpe Ratios are positive (and 152 are negative),

with the median SR being -0.74. It also has the lowest median Sharpe Ratio out of all tickers.

We also note that evaluating on data from the Utilities Sector (XLU) usually results in strong signals. On the other hand, there are certain generators for which mainly negative Sharpe Ratios occur. The worst overall generator in terms of number of negative Sharpe Ratios is ALB (XLB), where 131 tickers have negative performance, and where the median SR is -0.3. The strongest generator is the one trained on AEP (XLU), with 136 positive SRs, and the median of 0.32.

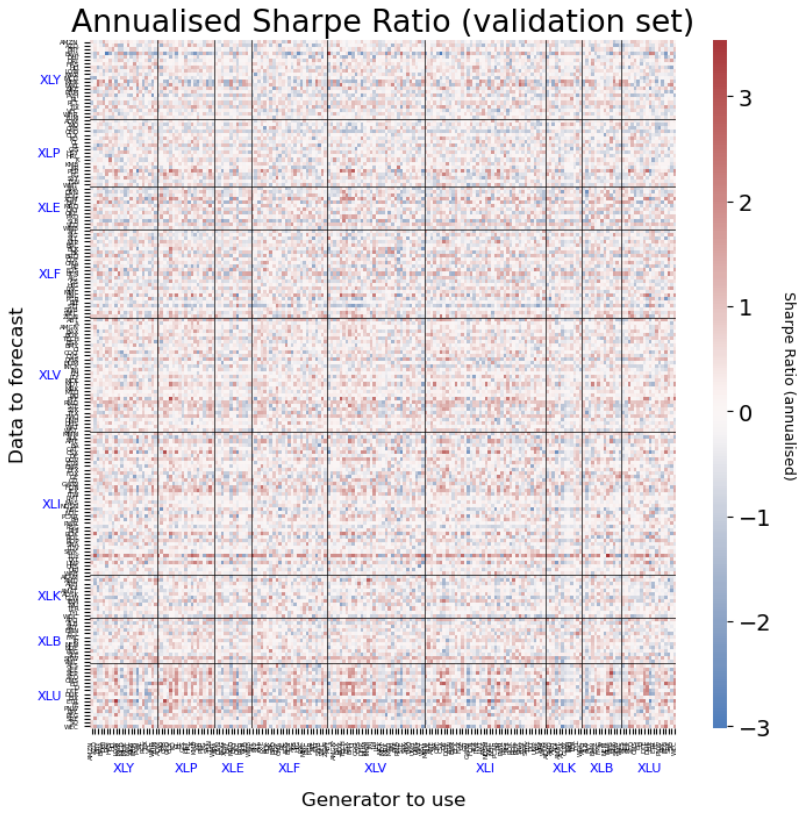
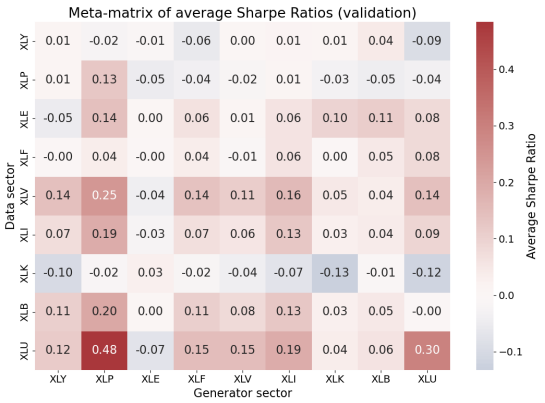
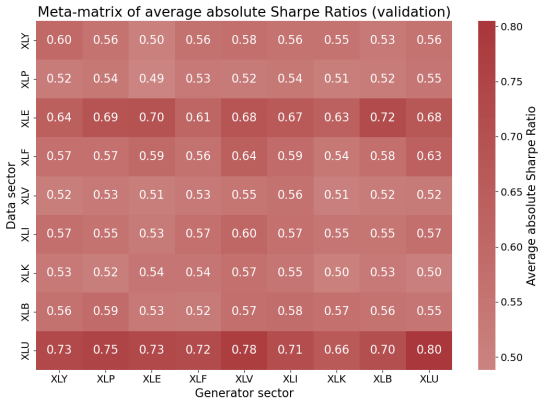


Figure 6.3: Annualised Sharpe Ratio over the validation set. Each row represents the data to forecast, and each column represents a generator trained on a specific data set.

By considering the meta-matrix of the Sharpe Ratio (on the validation set) matrix from Figure 6.3, we confirm in Figure 6.4 that the highest Share Ratios are achieved on data from the XLU sector, both in terms of average values (Figure 6.4a) and average absolute values (Figure 6.4b). We observe that the highest average Sharpe Ratios are produced by generators trained on XLP sector data (Consumer Staples), evaluated on the XLU constituents (Utilities). The most difficult sector to forecast over the validation set was XLK (Technology).



(a) Cumulative (portfolio) bet size for each method as a function of time.



(b) Cumulative (portfolio) absolute bet size for each method as a function of time.

Figure 6.4: Sharpe Ratio (validation set) meta-matrix for average and average absolute values.

The correlation matrix and the Sharpe Ratio matrix displayed in Figures 6.2 and 6.3, respectively, are used to construct the weights $w_{i,j}$ in the *Correlation-based* and *SR-based* approaches. We compare the Sharpe Ratio performance achieved by different methods on the test set in Table 6.2, and note that the best portfolio performance by far is achieved by *LASSO* (6.20). There is a significant improvement in performance when a meta-generator is constructed, and information is shared, compared to the baseline case where each data is forecast via its own generator only (the *Identity* method). Ridge regression (6.21) leads to significantly worse portfolio performance compared to LASSO, due to the lack of sparsity. The *SR-based*, *PnL-thr*, and *PnL-max* approaches result in similar overall performance, whereas *PnL-based* has a slightly lower annualised Sharpe Ratio. Opting for the weights constructed from the correlation matrix of the data leads to the second-highest portfolio Sharpe Ratio, which is still significantly below the one achieved by *LASSO*.

In Figure 6.5 we investigate the similarities and differences between the LASSO and baseline performance. There is an improvement for 116 out of 193 tickers, and the average difference between the LASSO performance and Identity in terms of Sharpe Ratio is 0.16, as indicated in Table 6.2. There are 33 stocks for which the baseline SR was positive (test set), but opting for the LASSO-based graph method resulted in negative performance. Furthermore, in 16 instances negative performance was further amplified. The most improved sectors are XLU, where 89% have a better Sharpe Ratio, and XLI, where 70 % of tickers have increased performance. On the other hand, 58% of XLK tickers have worse perfromance than they did, and 52% for XLF.

However, on XLF, the performance change was more significant since forecasting more of the tickers resulted either in more extremely negative Sharpe Ratios, or in ruining positive performance. Some of the best performers for the baseline, *Identity*, such as NCS, AVB, PVH, and ALH, had negative performance for *LASSO*.

Type	Portfolio SR	Mean SR	SR Std	Num.Pos. SR
Identity	2.00	0.16	0.74	104
Correlation-based	2.27	0.41	0.75	139
SR-based	2.22	0.45	0.79	142
PnL-thr	2.32	0.45	0.85	**143**
PnL-max	2.30	0.31	0.86	125
PnL-based	1.93	**0.49**	0.77	136
LASSO	**4.39**	0.32	0.72	125
Ridge	1.51	0.18	**0.70**	121

Table 6.2: Comparison of annualised Sharpe Ratios on the test set. The mean and standard deviation reflect the distribution of individual Sharpe Ratios computed for each of the 193 assets analysed. The final column shows the number of positive Sharpe Ratios across the tickers.

The method achieving the highest average Sharpe Ratio across all tickers are based on PnL-metrics, and not *LASSO*. This is due to the fact that *LASSO* results in lower PnL correlation compared to *SR-based*, as indicated in Table 6.3. The highest correlations are observed in the *PnL-based* instance. Furthermore, sparsity in *LASSO* reduces the significance of a particular generator, whereas in *PnL-based* methodology, certain generators can hold significant weight (as demonstrated in Figure 6.3), resulting in related outputs.

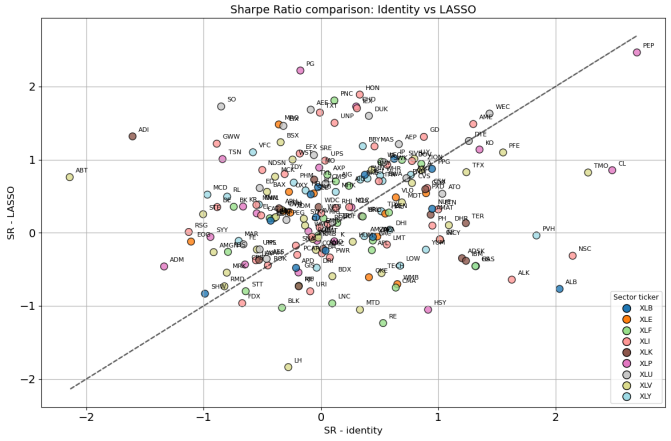


Figure 6.5: LASSO Sharpe Ratio vs Identity Sharpe Ratio. Comparison on the test set.

Method	Mean	Mean Abs	Std	Min	Max
Identity	0.001	0.068	0.090	-0.541	0.543
Correlation-based	0.015	0.058	0.075	-0.363	0.512
SR-based	0.011	0.068	0.090	-0.487	0.687
PnL-thr	0.010	0.062	0.084	-0.488	0.686
PnL-max	0.003	0.058	0.074	-0.403	0.432
PnL-based	0.012	0.074	0.096	-0.507	0.688
LASSO	0.0003	0.063	0.085	-0.502	0.612
Ridge	0.003	0.095	0.129	-0.603	0.663

Table 6.3: PnL correlation statistics for each method. Correlation between PnLs of asset i and j are calculated only once, for $i \neq j$. The smallest values are in blue and the highest are in red.

The Sharpe Ratio distributions for *Identity*, *Correlation-based*, *SR-based*, *PnL-max*, and *LASSO* methods are displayed in Figure 6.6. All methods significantly shift the initial (*Identity*) Sharpe Ratio distribution towards positive values. Only *PnL-thr*, *PnL-based* and *SR-based* approaches are able to achieve an annualised Sharpe Ratio above 3, all on PEP.

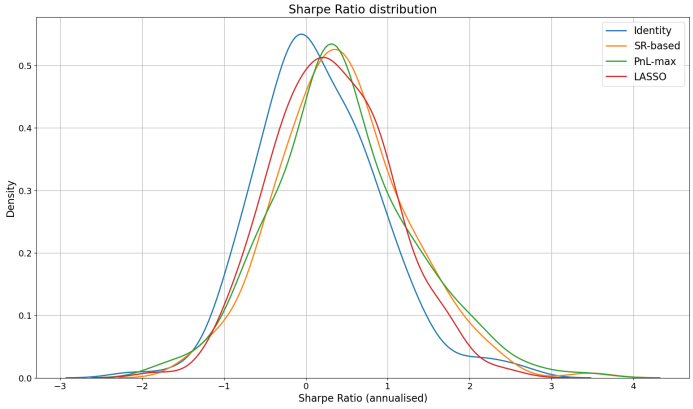
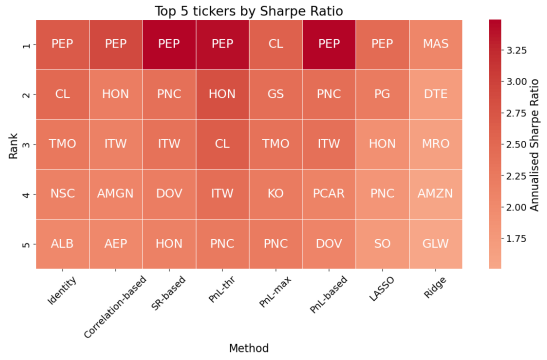


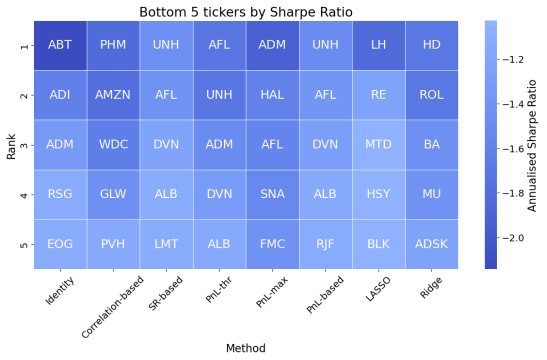
Figure 6.6: Distribution of the annualised Sharpe Ratio over the validation set, across 193 tickers.

In Figure 6.7a we investigate which five tickers result in the highest Sharpe Ratios by each method considered. We observe that different tickers result in the best performance across methods, but that there are data sets on which all methods perform well. Tickers such as PEP (6), PNC (5), HON (4), ITW (4), CL (3), DOV (2) and TMO (2) are common choices for top 5, indicating that performance on them remains similar across different graph constructions.

Similarly, we investigate which tickers result in the worst performance in Figure 6.7b. The worst tickers for the *Identity* method are not in the bottom 5 tickers for



(a) Top 5 tickers as measured by Sharpe Ratio, for each method. Strategies for PEP and HON are usually performing in the top 5.



(b) Bottom 5 tickers as measured by Sharpe Ratio, for each method. AFL, ADM, ALB, ABT, and BLK appear to be particularly hard to forecast.

Figure 6.7: Top and Bottom 5 tickers by Sharpe Ratio across methods.

other methods, apart from ADM. Interestingly enough, ALB is the fifth best for *Identity*, but it is in the worst five in the *PnL-thr*, *PnL-based*, and *SR-based* instances. Tickers which are usually labelled as the most difficult to forecast are AFL (4), ADM (3), UNH (3), DVN (3), and ALB (3).

From Figures 6.7a and 6.7b, we note that there are instances where certain tickers are easy to forecast by some graph-based methods, but difficult by others. In Figure 6.8, we investigate the similarity between the outputs of different methods by considering the Pearson correlation between portfolio PnLs produced by different methods. Unsurprisingly, since they are based on similar metrics, Figure 6.8 shows that the *SR-based*, *PnL-based* and *PnL-thr* graphs produce PnLs which are almost perfectly correlated. There is a cluster of high correlation between methods based on creating a graph driven by financial performance on the validation set. All methods have very low correlation with the baseline setting in which the graph adjacency matrix is the identity. Importantly, *Lasso* and *Ridge* regression lead to very different PnLs, whose correlation is around 20%. Ridge regression results in low correlation with all other methods, whereas *LASSO* produces portfolios correlated around 30% with those produced by graphs based on financial performance.

We show the overall cumulative PnL for different methods in Figure 6.9. As a reminder, this would be the profit and loss of a trading strategy in which $h_i(a_t^i)$ in USD is invested in stock i at time t, and the opposite trade in the corresponding ETF (same monetary amount) for hedging purposes. Figure 6.11a shows that all methods other than LASSO and Ridge mainly forecast that the individual stocks will outperform their corresponding sector ETF. Not only that, but the overall absolute

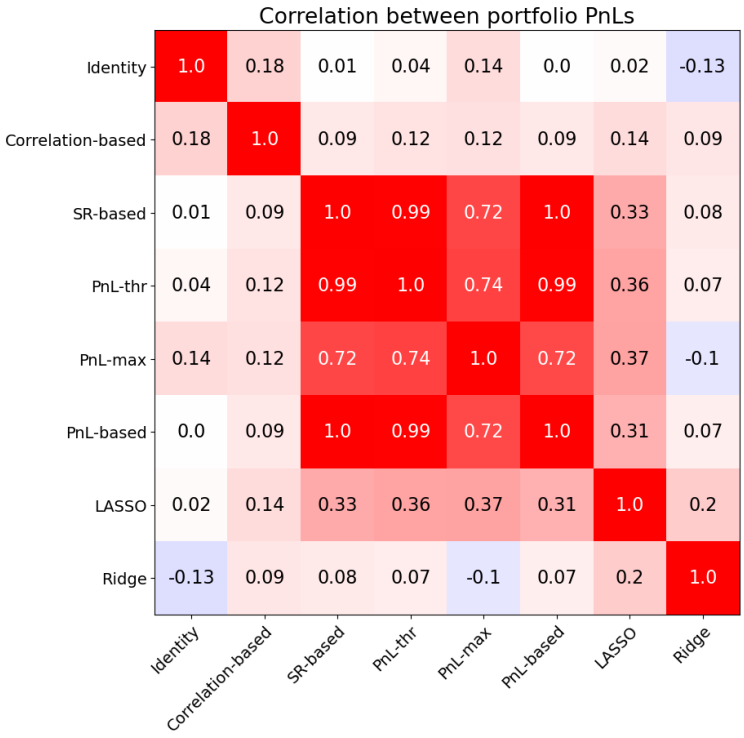


Figure 6.8: Pearson correlation between portfolio PnLs achieved by different methods (test set), rounded to two decimal places. A more precise estimate of the Pearson correlation between *SR-based* and *PnL-max* PnLs is 0.9986.

exposure (sum of absolute bet sizes) is significantly lower for the regression-based methods. However, the cumulative absolute bet size for LASSO is very similar to that implied by the return correlation matrix. For a better comparison, Figure 6.10 shows the cumulative PnL for every strategy, rescaled by the overall exposure to the bet size. Up until mid 2020, *SR-based*, *PnL-thr*, *PnL-based*, and *LASSO* methods attain very similar performance. From October 2020, the LASSO-implied strategy starts to perform significantly better than the other benchmarks.

Figures 6.9 and 6.10 illustrate the similarity of the methods based on financial performance over the validation set, as evidenced by their correlation (Figure 6.8). *PnL-thr* and *PnL-max* in absolute terms result in very similar cumulative PnL, but since *PnL-max* offers no diversification and opts for the *strongest signals*, its bet exposure is the highest (Figure 6.11b). The *SR-based* and *PnL-based* methods, unsurprisingly, produce cumulative PnLs which appear to be parallel to each other most of the time.

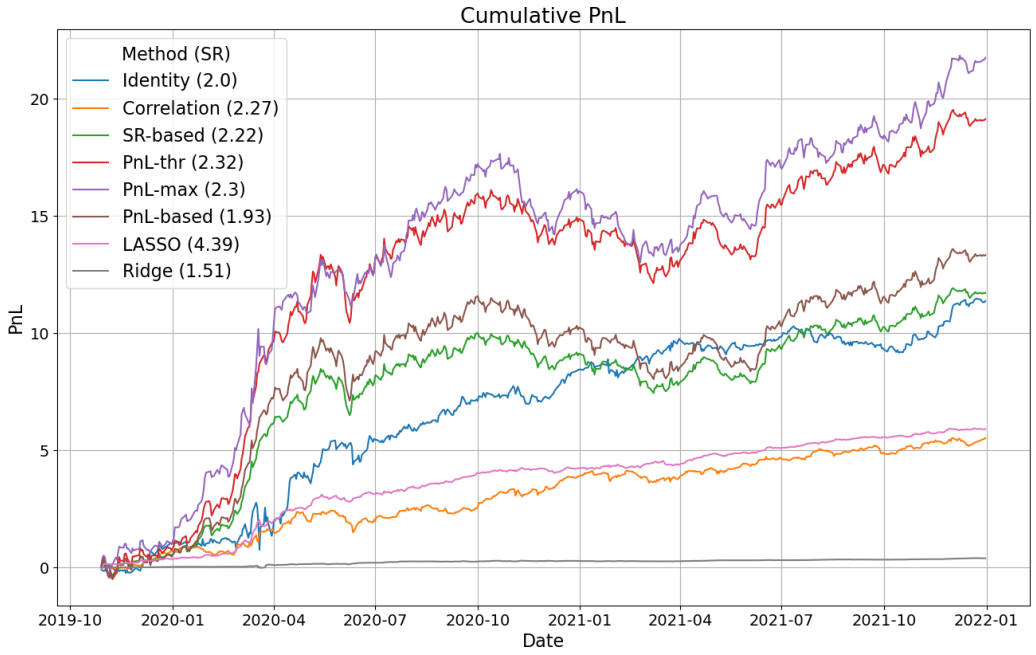


Figure 6.9: Cumulative (portfolio) PnL for each method as a function of time.

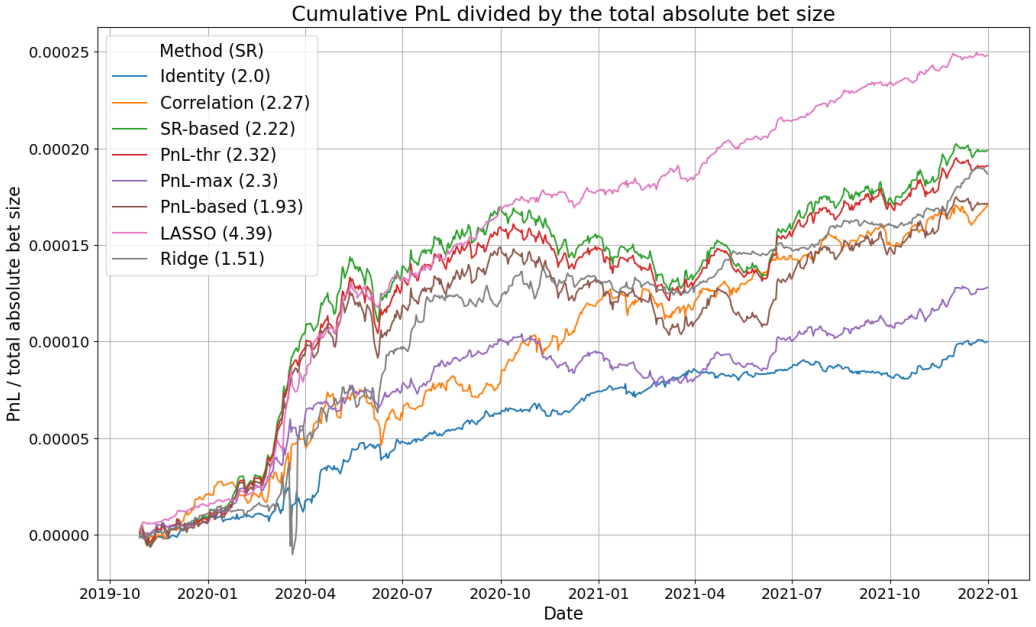
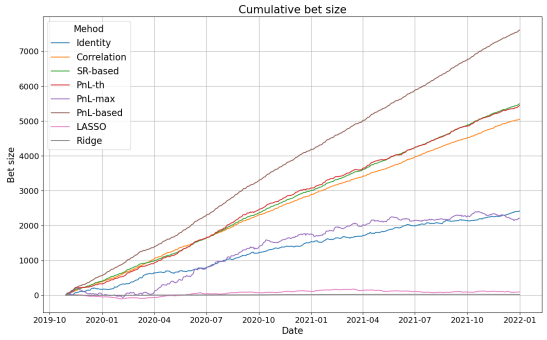
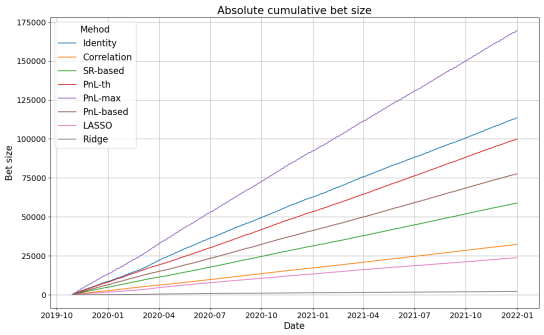


Figure 6.10: Cumulative (portfolio) PnL for each method as a function of time, rescaled by the total absolute bet size over the entire test set.



(a) Cumulative (portfolio) bet size for each method as a function of time.



(b) Cumulative (portfolio) absolute bet size for each method as a function of time.

Figure 6.11: Cumulative and absolute cumulative bet sizes across methods over time.

6.3.2 Robustness with respect to the LASSO regularisation parameter

We investigate the extend to which the LASSO performance changes if the regularisation parameter is changed from $\eta = 0.000001$ to $\eta = 0.000002$ (twice bigger) and $\eta = 0.0000005$ (twice smaller). From Table 6.4 we note that the performance increases slightly for $\eta = 0.000002$, and decreases for $\eta = 0.0000005$. Nevertheless, all statistics are on the same scale, and close to each other, indicating that updating the value from $\eta = 0.000001$ does not have a significant impact on the results.

Regularisation parameter	Portfolio SR	Mean SR	SR Std	Non-zero weights
$\eta = 10^{-6}$	4.39	0.32	0.72	6768
$\eta = 2 \cdot 10^{-6}$	4.57	0.35	0.76	3959
$\eta = 5 \cdot 10^{-7}$	4.08	0.28	0.72	10713

Table 6.4: Comparison of annualised Sharpe Ratios on the test set, as well as the number of non-zero weights $w_{i,j}$. The mean and standard deviation reflect the distribution of individual Sharpe Ratios computed for each of the 193 assets analysed.

We analysed the overall performance comparison in Table 6.4. To understand how much the results change at the individual ticker level, we plot the Sharpe Ratio achieved by $\eta = 10^{-6}$ vs the alternative values in Figure 6.12. We note that in both cases the points scatter around the $y = x$ line. However, 106 out of 193 points for $\eta = 2 \cdot 10^{-6}$ lie above $y = x$, showing better performance for this particular choice compared to the original one. As discussed previously, opting for a smaller value ($\eta = 5 \cdot 10^{-7}$), reduces the performance. In this instance, the majority of the points (107) are below the line $y = x$. In fact, in order to choose η, we compared in-sample

performance between different choices of η, since the fit was obtained on the whole validation set. Separating the validation set into two parts, with the first 80% used for regression fitting, and the other 20% for comparison would have still resulted in selection of $\eta = 0.000001$. However, as we have seen, our methodology does not heavily depend on this hyperparameter.

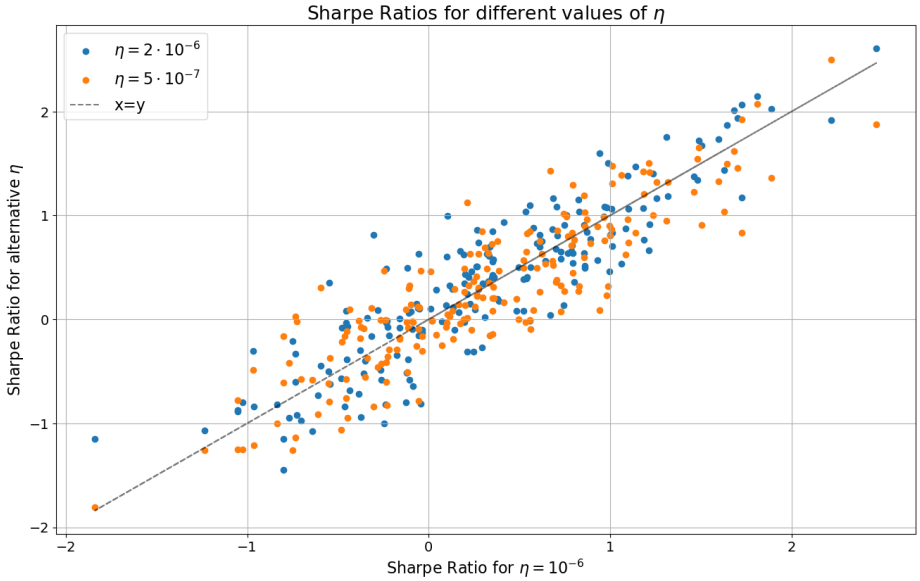


Figure 6.12: Comparing the annualised Sharpe Ratio for each ticker achieved by $\eta = 10^{-6}$ and alternative values. The black dashed line is $y = x$.

6.3.3 The role of uncertainty estimates

To showcase the importance of soft classification in our approach, we compare the financial performance with trade sizes given by three different cases:

1. by considering the direction of the overall vote, with the expected signs still taken from the individual generators;

2. by the linear combination of the direction of the trades implied by individual generators;

3. by the direction of the linear combination of directions of individual trades.

In all settings, we still retain the same weights used so far. The financial performance on the test set is given in columns I in Table 6.5 for the first case (i.e., considering the sign of the overall bet), column II for the second instance (i.e., considering the linear combination of trade directions), and for the third instance the results are shown

in column III (where hard classification is performed both on the individual and on the global level). We compare these results with those in which soft classification is performed (Table 6.5), and observe a significant drop in performance in *Identity*, and *LASSO* implied weights. However, performance remains relatively similar for the graphs constructed via measures of financial performance. The only instance in which *Ridge* regression has visible improvement is when hard classification is performed after the trade size has been aggregated across the generators.

Statistics	Portfolio SR				Mean SR				SR Std			
Type	0	I	II	III	0	I	II	III	0	I	II	III
Identity	2.00	1.37	1.73	1.37	0.16	0.12	0.12	0.12	0.74	0.75	0.77	0.75
Correlation-based	2.27	2.06	2.28	1.63	0.41	0.35	0.40	0.28	0.75	0.72	0.82	0.75
SR-based	2.22	2.29	2.35	**2.50**	0.45	0.35	0.45	0.37	0.79	0.72	0.69	0.72
PnL-thr	2.32	2.40	2.46	2.40	**0.45**	**0.38**	**0.45**	**0.38**	0.85	0.78	0.85	0.78
PnL-max	2.30	2.17	2.17	2.17	0.31	0.28	0.28	0.28	0.86	0.83	0.83	0.83
PnL-based	1.93	1.88	2.05	2.03	**0.49**	0.30	0.41	0.31	0.77	**0.67**	0.77	**0.67**
LASSO	**4.39**	**3.62**	**3.45**	2.32	0.32	0.32	0.22	0.21	0.72	0.72	**0.67**	0.68
Ridge	1.51	2.55	1.18	1.01	0.18	0.21	0.09	0.08	**0.70**	0.71	0.68	**0.67**

Table 6.5: Sharpe ratio comparison (test set) for various methods. Three statistics are shown: Portfolio SR, Mean SR, and SR Std, across three types of classification: I - sign of the linear combination of expected sign, II - linear combination of the sign of the expected direction, and III - sign of the linear combination of signs of expected directions. The value 0 is the baseline for soft classification, results for which are displayed in Table 6.2. The best results for every column are in bold.

We show the cumulative PnL of the portfolio corresponding to the results of column I in Table 6.5 in Figure 6.13. This corresponds to only considering directionality of the overall *vote*, and discarding its intensity. As discussed previously, we observed a decrease in performance, compared to the initial approach in Figure 6.9. Our analysis indicates that the uncertainty estimates actually reduce the variance in daily PnL. The impact of uncertainty estimates is most prominent in the *LASSO* setup.

Figure 6.14 considers the relationship between the bet size and performance. We observe that the highest absolute bet size is obtained for tickers belonging to the XLU sector, indicating higher forecast certainty. Furthermore, in the vast majority of the cases where the total absolute bet size is larger than the bulk (above 200), positive Sharpe Ratios are achieved.

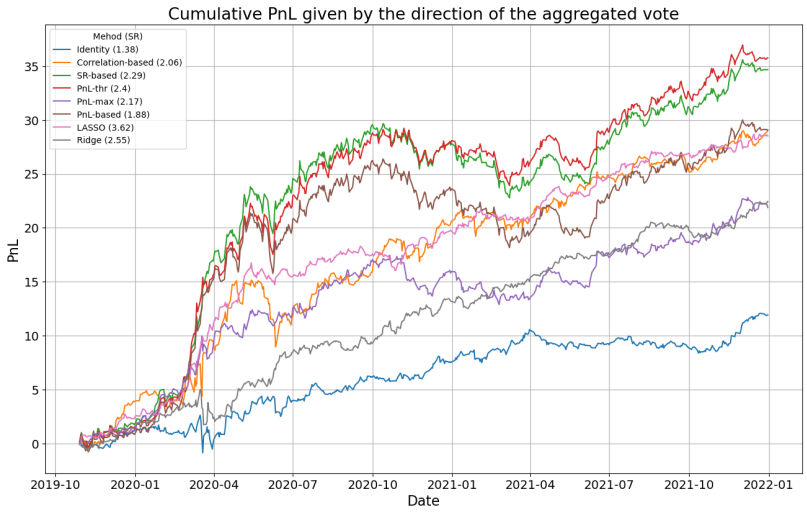


Figure 6.13: Cumulative PnL in the case when only the direction of the overall vote is accounted for.

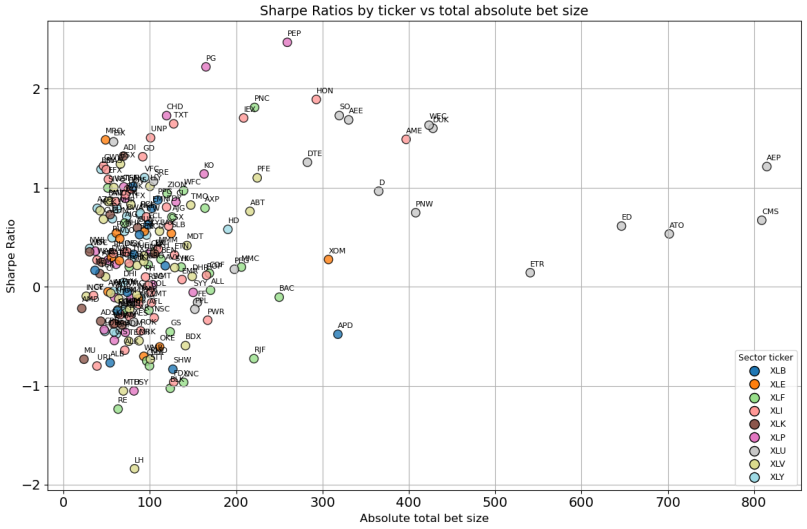


Figure 6.14: Total absolute bet size vs annualised Sharpe Ratio.

6.4 Graph analysis

We examine the graph implied by LASSO regression (6.20). There are only 31 self-loops, out of 193, i.e., for only 31 data sets the generator trained on that particular data is used. The average weight of the self-loop is 0.013, with the smallest being -0.105 (K) and the largest 0.225 (DTE). Out of the 31 self-loops, 18 are positive and 13 are negative. Overall, there are 3438 positive edges and 3330 negative edges, a total of 18.17% of weights are non-zero. The average positive weight is 0.030, and

the average negative weight is -0.027. The smallest weight is -0.88, and it is an edge from ATO to JNJ. The highest weight is 1, since there are two instances in which only one generator is used per ticker, AEP (PPG generator used) and CMS (CPB generator used).

Figure 6.16 shows the number of generators used to forecast each ticker (out nodes). The average number of genertors used is 35.06, and the median is 33. Usually, the Sharpe Ratio decreases as the number of generators selected increases, as implied by the scatter plot displayed in Figure 6.15. This could be both due to the difficulty of forecasting a particular stock, and the regularisation parameter not being high enough for certain stocks.

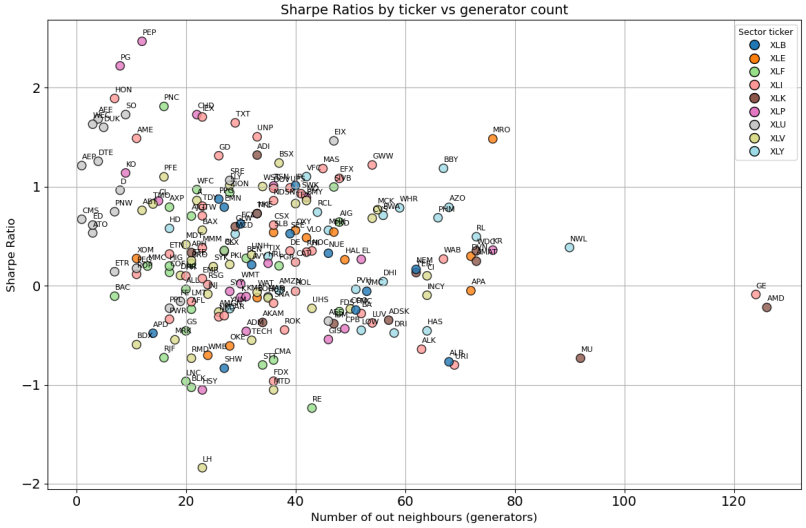


Figure 6.15: Number of generators used to forecast each ticker vs the annualised Sharpe Ratio on the test set.

Similarly, in Figure 6.17 we analyse the number of tickers utilising each generator (in-degree nodes). The median number of tickers utilising a particular generator is 28. There are tickers whose generators are never used: 'CHD' (XLP), 'GIS' (XLP), 'WMT' (XLP), 'XOM' (XLP), 'HAL' (XLE), 'PXD' (XLE), 'A' (XLE), 'LLY' (XLV), 'ETN' (XLI), 'HON' (XLI), 'GLW' (XLK), 'AVY' (XLB), 'ED' (XLU), 'PPL' (XLU). There are 4 generators which are used on more than 50% of all tickers: LH (105, XLV), RHI (101, XLI), PNW (100, XLU), PPG (99, XLB).

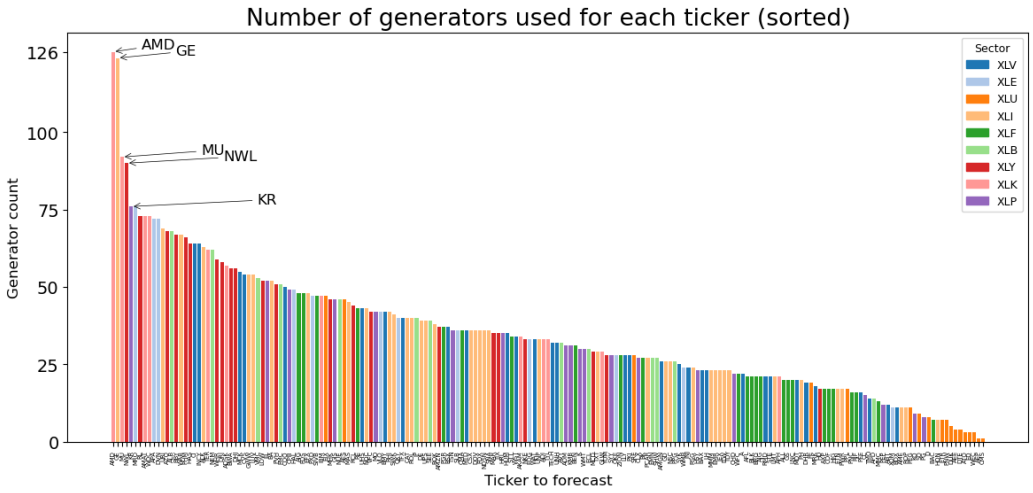


Figure 6.16: Number of generators used to forecast each ticker. For each ticker i, the number of generators used is the number of out-neighbours, i.e., the size of the set $\{j \in \{1, \ldots, N\} : w_{i,j} \neq 0\}$.

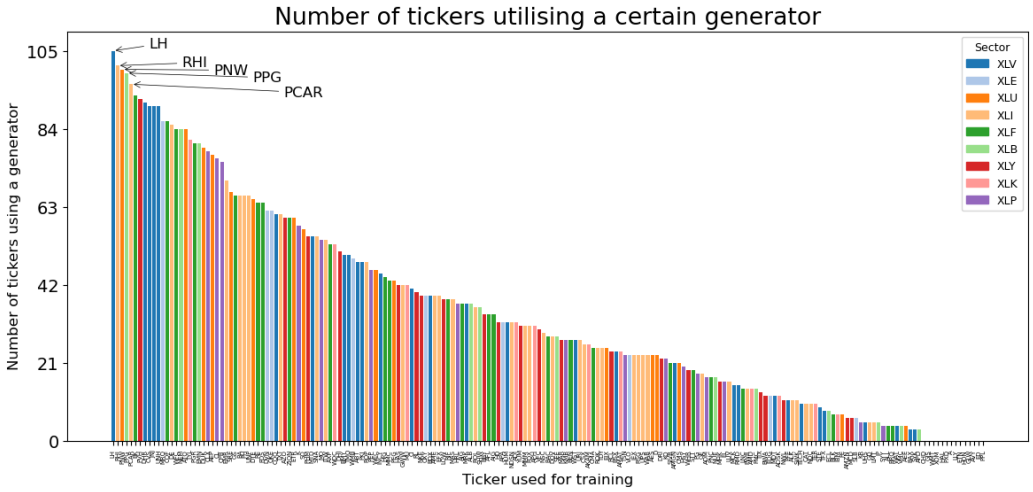


Figure 6.17: Number of tickers which use a given generator for forecasting. For each generator trained on ticker j, the number of generators used is the number of in-neighbours, i.e., the size of the set $\{i \in \{1, \ldots, N\} : w_{i,j} \neq 0\}$.

Table 6.6 shows that on average, generators trained on the XLU sector are most commonly used, and that stocks from XLU use the lowest number of generators. It is also by far the sector with the highest Sharpe Ratio. In turn, the second lowest Sharpe Ratio is achieved on stocks from XLK, whose generators are used the least, and whose stocks opt for the most generators.

Node centrality plays a crucial role in network analysis as it quantifies the relative importance or influence of individual nodes within the overall network structure. To

Sector	Avg OUT	Avg IN	Avg OUT - IN	Average SR
XLB	39.62	31.92	7.69	0.16
XLE	44.17	30.92	13.25	0.25
XLF	25.92	41.84	-15.92	0.07
XLI	36.55	35.20	1.35	0.41
XLK	56.67	27.67	29.00	0.14
XLP	31.68	30.32	1.37	0.46
XLU	12.39	43.39	-31.00	0.82
XLV	31.31	35.59	-4.28	0.18
XLY	50.27	31.82	18.45	0.35

Table 6.6: Average number of out neighbours (OUT), in-neighbours (IN), and their difference (out - in) by sector. The number of out neighbours counts the number of generators used for a particular stock, and number of in neighbours counts how many stocks use a generator trained on a particular stock's data. The highest values are in red, and the lowest are in blue.

this end, in order to better understand node importance within our network, Table 6.7 shows top Page Rank centrality scores [109], computed using the absolute edge weights, both at the ticker (node) level, and also averaged across each sector. In this instance, a higher Page Rank score for a node i denotes that there are many other (also important) nodes pointing towards node i, i.e. tickers with high Page Rank score have more important generators, which are commonly used. We find that the highest ranked node by Page Rank is PPG (XLB), which has 99 in-neighbours (assets which use the PPG generator for their meta-generator), and 26 out-neighbours (number of generators used to construct the meta-generator). It is the sixth highest ranking ticker in terms of the difference between the number of in- and out-neighbours, and, as noted in Figure 6.17, the fourth node by the number of in-neighbours. Unsurprisingly, the sector with the highest average Page Rank score is XLU, very closely followed by XLF. XLU is also the sector with the highest difference between the number of in- and out-neighbours, with the overall lowest number of out-neighbours and the highest number of in-neighbours, as shown in Table 6.6. All of these statistics are followed by XLF.

It is natural to consider the industry sectors as clusters in our settings, especially since the returns under consideration were in excess of the corresponding ETF return. In order to better understand the network structure, we perform Singular Value Decomposition (SVD) on the adjacency matrix implied by the *LASSO* weights. The singular value plot in Figure 6.18 indicates that there are approximatively 8 clusters present, with 4-5 noticeable spectral gaps in line with the differences in the number of

Top 9 Tickers by Page Rank Score				Average Page Rank by Sector		
Rank	**Ticker**	**Page Rank Score**	**Sector**	**Rank**	**Sector**	**Average Page Rank**
1	PPG	0.0210	XLB	1	XLU	0.0060
2	JNJ	0.0195	XLV	2	XLF	0.0059
3	LH	0.0146	XLV	3	XLI	0.0055
4	CL	0.0141	XLP	4	XLV	0.0053
5	SRE	0.0141	XLU	5	XLB	0.0052
6	PH	0.0139	XLI	6	XLP	0.0047
7	BK	0.0136	XLF	7	XLY	0.0045
8	CPB	0.0132	XLP	8	XLE	0.0045
9	BA	0.0130	XLI	9	XLK	0.0037

Table 6.7: Top 9 Tickers and Average Page Rank by Sector

tickers under consideration per sector (40, 32, 25, 22, 19, 18, 13, 12, 12).

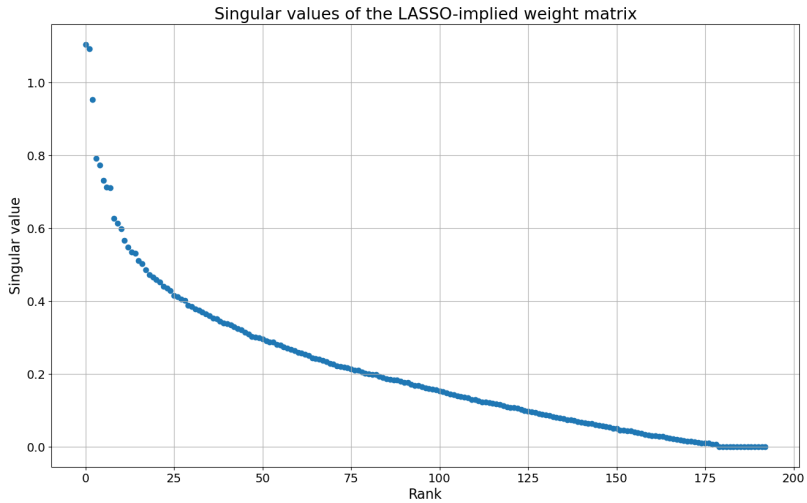
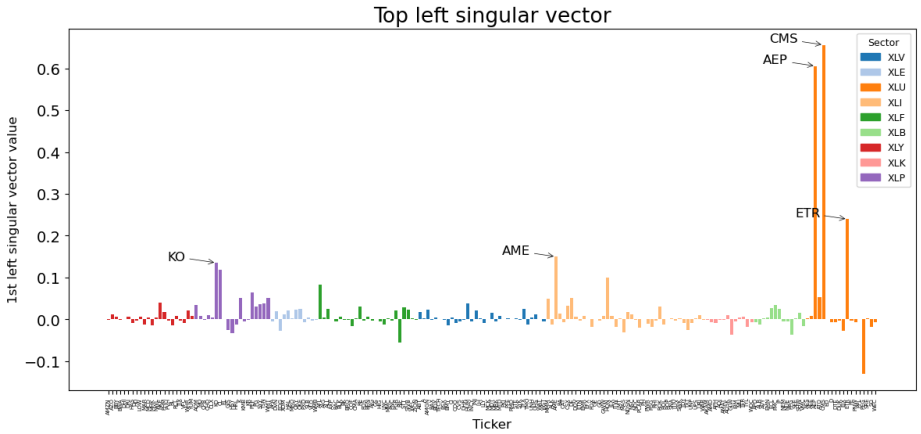


Figure 6.18: Singular values of the weight matrix. There are four large gaps separating the top eight singular values from the bulk. There is a sharp decline in the singular values, indicating a low-rank structure for the weight matrix.

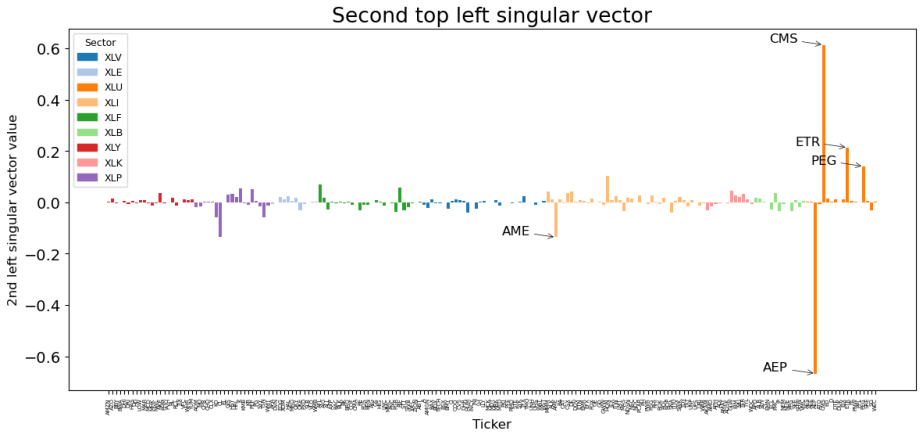
The two top left singular vectors, shown in Figure 6.19 are significantly driven by CMS (XLU), and AEP (XLU), ETR (XLU), AME (XLI), PPG (XLU), and KO (XLP). All of these tickers have high ratio of in- neighbours to out- neighbours, i.e. their generators are used more than they use generators trained on different data sets. Furthermore, we observe very strong support on the XLU sector, and similar levels of importance from tickers grouped by sector membership.

When it comes to the right singular vectors, displayed in Figure 6.20, by far the most prominent tickers are PPG (XLB) and CPB (XLP). These are in top 10 most important nodes when ranked by Page Rank (Table 6.7).

Given the spectral structure and the setup, we perform the remaining analysis of the underlying graph by grouping tickers by sector. In order to understand the connectivity of the underlying graph, we visualise within-sector edges in Figure 6.21,



(a) Top left singular vector. The highest support is on CMS, ARP, ETR, KO, and AME.

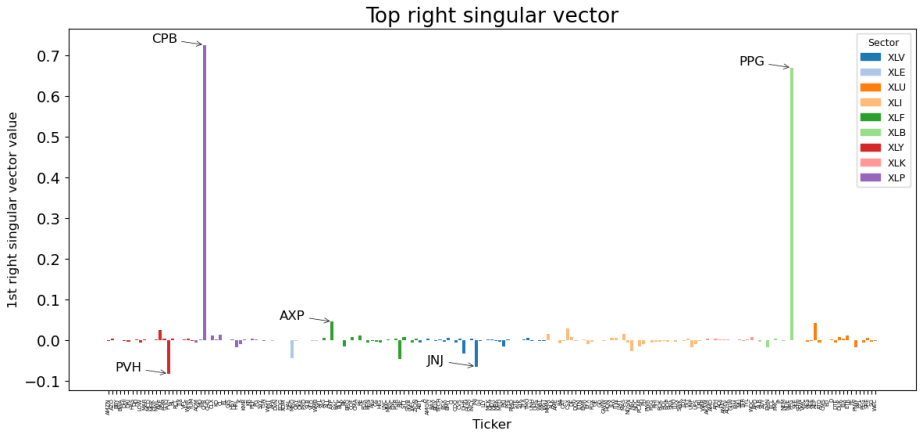


(b) Second top left singular vector. The highest support is on AEP, CMS, ETR, PEG, and AME.

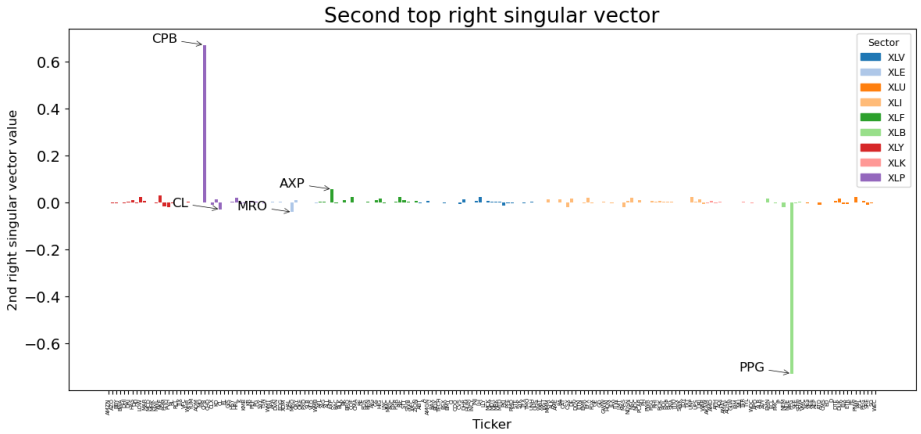
Figure 6.19: Top two left singular vectors of the weight matrix.

and observe that the highest number of edges is within XLV, XLI, and XLY. Some sectors are poorly inter-connected, eg XLK, XLB, XLU, and XLE. However, the edges present inside XLU are of higher importance compared to the edges of some more inter-connected sectors, such as XLY.

In Figure 6.22 we show the proportion of total absolute weight between sectors, rescaled by their size. Each row corresponds the data for evaluation, and each column corresponds the data for training. Since the sectors have different sizes, total weight from sector i to sector j is divided by the square root of the product of the sizes of the two sectors. The corresponding meta-graph whose edges are the total weights between the sectors is displayed in Figure 6.23, with nodes proportional to the size of the group.



(a) Top right singular vector. The highest support is on CPB, PPG, PVH, JNJ, and AXP.



(b) Second top right singular vector. The highest support is on PPG, CPB, and AXP.

Figure 6.20: Top two right singular vectors of the weight matrix.

The heatmap shown in Figure 6.22 indicates that the most self-weight is contained within XLI, XLV, and XLU. Even though there are more edges between the nodes within XLY sector than XLU, the edges inside the XLU sector have a higher weight, as displayed in Figure 6.21. We note that the generators from XLI, XLF, XLV, and XLU are commonly used, whereas those from XLK are not a popular choice. The strongest connectivity is between XLI, XLV, XLF, XLY, and XLU.

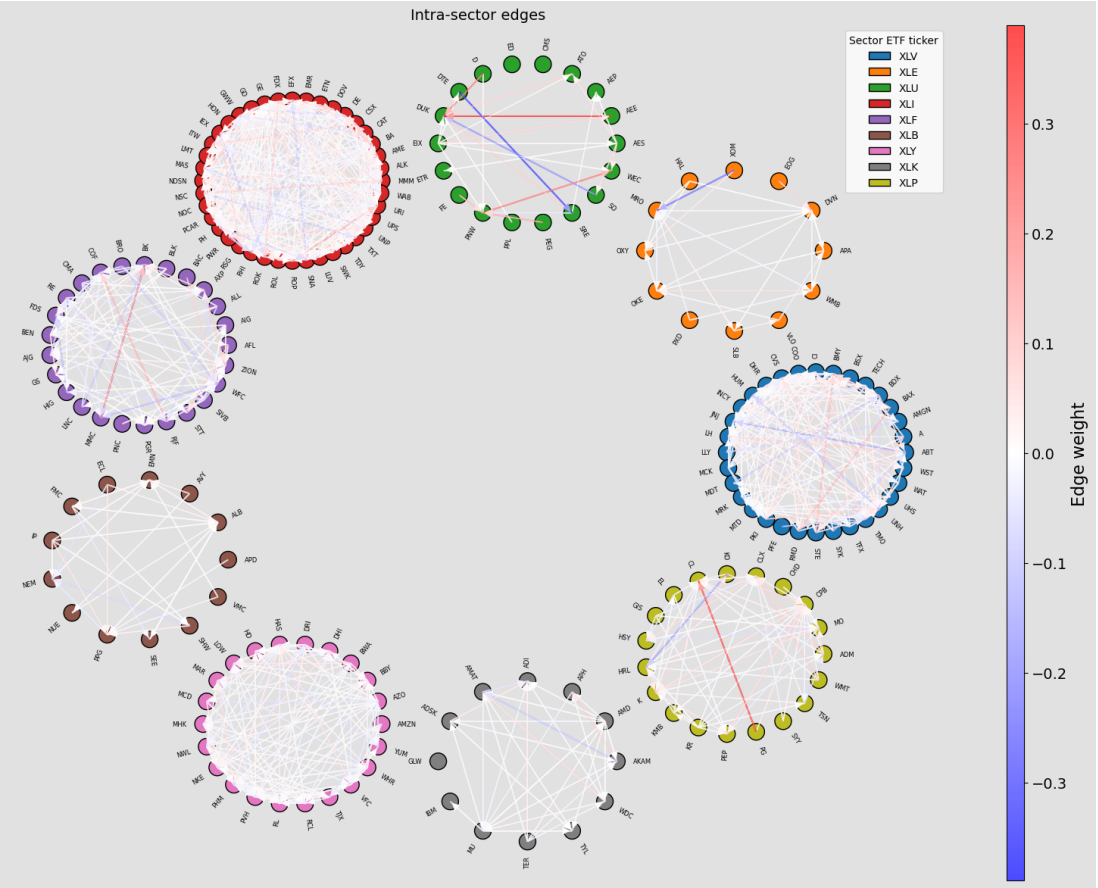


Figure 6.21: Intra-sector edges (without self-loops).

Figure 6.22: Rescaled total absolute weight between sectors. Total weight from i to j is divided by the square root of the product of the sizes of the two sectors.

There are high degrees of symmetry in the adjacency matrix corresponding to the meta-graph shown in 6.23, with the graph being fully-connected and the average absolute distance between each pair of directed weights being only 0.02. The strongest inter-sector connections are between XLF and XLV, and XLI.

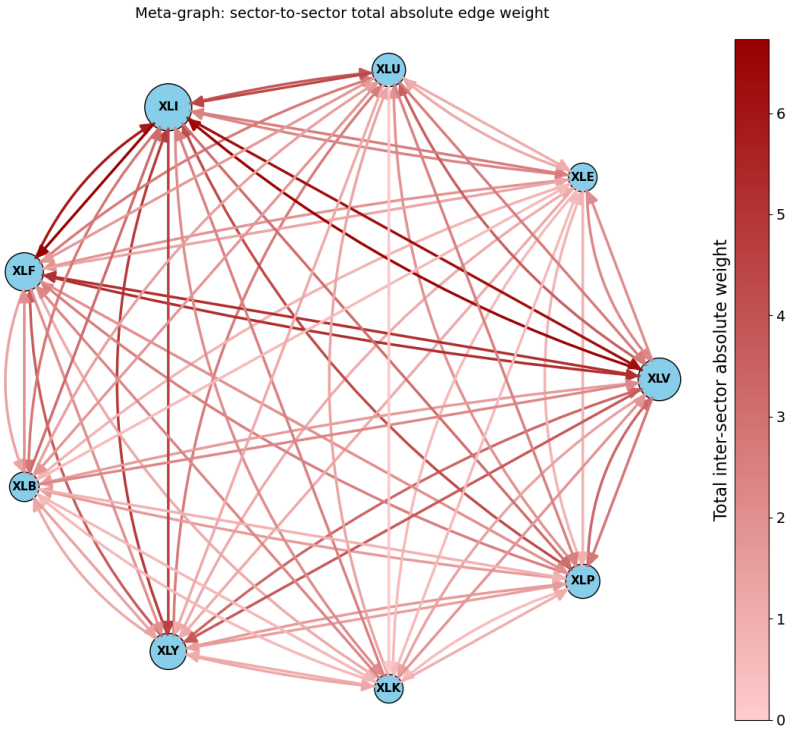


Figure 6.23: Inter-sector total absolute weights.

Lastly, we visualise the edges above 15% (without the DTE self-loop) in Figure 6.24. We find that the strong connections are mainly between nodes corresponding to different sectors, and not intra-sector. The edges of the highest weight are all from tickers from the XLU sector: AEP to PPG, ATO to JNJ, and CMS to CPB. In turn, these are some of the nodes with the highest Page Rank score (Table 6.7). From our analysis, we note that the data from the XLU sector potentially had the highest generalisation potential, since there is evidence to suggest that the generators trained on data from this particular sector are commonly used by other tickers. Furthermore, it is the most successful sector in terms of the Sharpe Ratio achieved on it, both prior to applying the meta-generator (Figure 6.4) and afterwards (Table 6.6). This could indicate higher data quality compared to other sectors, or lower signal-to-noise ratio.

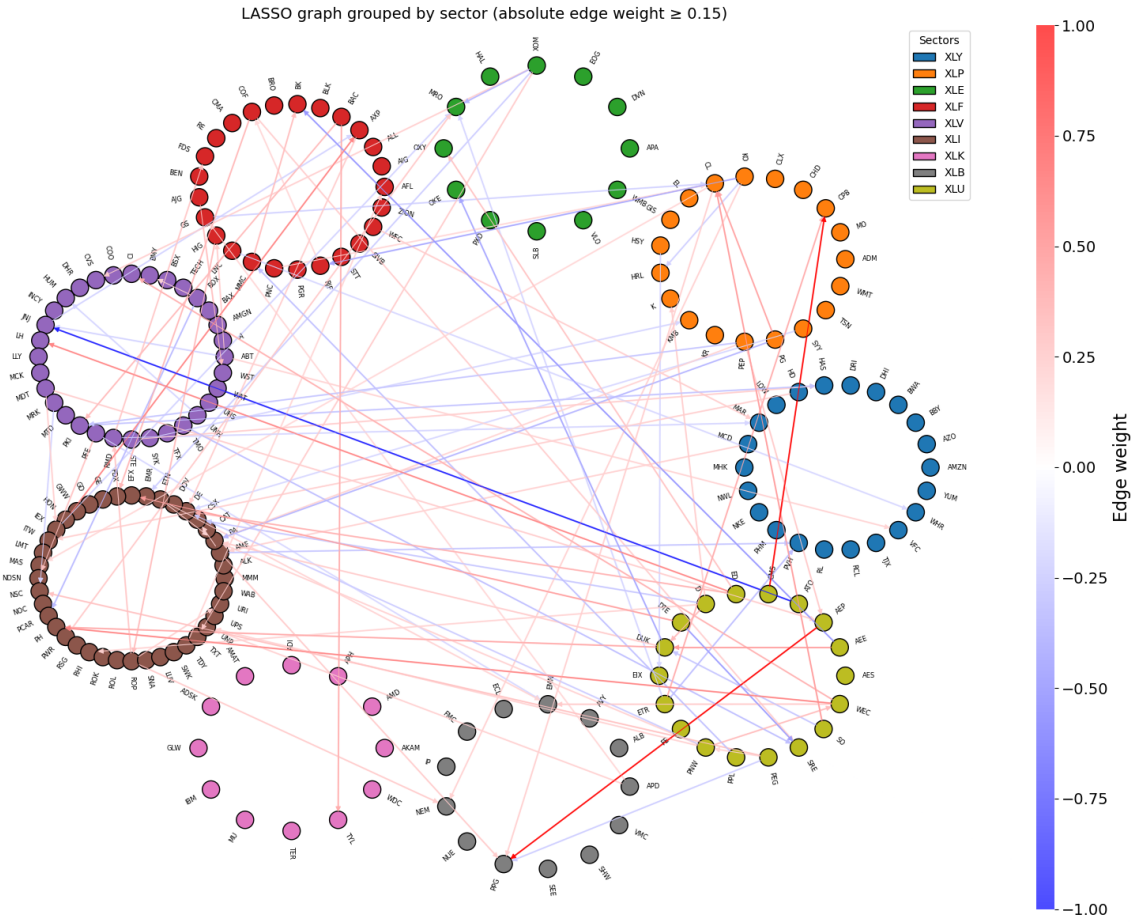


Figure 6.24: Weights holding more than 15% of the out weight for a node. The only high self-loop (DTE) is not included.

Chapter 7

Conclusion

This thesis has demonstrated the usefulness of generative models for tackling various modelling and risk management applications in finance. However, their applicability extends beyond the specific examples considered in our work. The methodology proposed in this thesis opens the door to a number of extensions, some of which are summarised below.

- **Hybrid factor–generative models.** While the focus of this thesis has been on fully data-driven models, an important next step would be to incorporate factor structures, such as those discussed in Chapter 2, into the generative architecture. These could serve as inductive biases or latent variables, enabling a balance between interpretability and expressiveness. Hybrid models have recently been explored using generative diffusion models [30] and GANs [26]. Unlike neural SDEs, which typically assume Brownian drivers, the factor dynamics could be entirely learned by the generative model. In the context of implied volatility surfaces, factor-based priors could improve regularity and potentially eliminate the need for the smoothness penalty used in VolGAN. However, regularisation, e.g. penalising deviations between the arbitrage penalty (2.9) of the observed data and the simulated scenarios, may be required.

- **Theoretical guarantees and convergence.** While GANs in particular are difficult to study from a theoretical perspective, once trained, *any* conditional generative model effectively defines a Markov Chain. This observation opens the door to studying its ergodicity, convergence properties, and asymptotic behaviour using tools from theory of stochastic processes.

- **Path simulation with generative models.** The focus in this thesis has been on one-step-ahead simulations, which suffice for many local tasks in risk

management and trading. An extension would be to explore recursive path generation by chaining the generator in a Markovian fashion.

- **Integration with portfolio optimisation and execution.** The probabilistic forecasts generated by the models in this thesis could be embedded within broader decision-making frameworks, such as portfolio allocation or order execution pipelines. This would enable a data-driven approach to trading under uncertainty.

- **Incorporating transaction costs in forecasting.** While transaction costs were considered in the hedging methodology of Chapter 4, similar ideas could be adapted to the forecasting frameworks of Chapters 5 and 6. One could, for instance, design policies that only trigger trades when the expected directional signal exceeds a threshold, thus filtering out low-conviction forecasts.

- **Dynamic graph weight updates.** In Chapter 6, the graph-based ensemble weights are static and derived via a one-time optimisation. Future work could explore dynamic, online updates of these weights in response to market conditions, or even integrate them into a differentiable learning framework via backpropagation.

- **Application to other markets and asset classes.** Although this thesis has focused primarily on equities, the methodologies are general and could be extended to other asset classes. In less liquid markets, data scarcity may become a limiting factor. In such cases, hybrid approaches incorporating stylised models or transfer learning may be necessary.

Bibliography

[1] Shamima Ahmed, Muneer M. Alshater, Anis El Ammari, and Helmi Hammami. Artificial intelligence and machine learning in finance: A bibliometric review. *Research in International Business and Finance*, 61:101646, 2022.

[2] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[3] Mehdi El Amrani, Antoine Jacquier, and Claude Martini. Dynamics of Symmetric SSVI Smiles and Implied Volatility Bubbles. *SIAM Journal on Financial Mathematics*, 12(2):SC1–SC15, 2021.

[4] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[6] Marco Avellaneda, Robert Buff, Craig Friedman, Nicolas Grandchamp, Lukasz Kruk, and Joshua Newman. Weighted Monte Carlo: a new technique for calibrating asset-pricing models. *International Journal of Theoretical and Applied Finance*, 4(01):91–119, 2001.

[7] Marco Avellaneda, Brian Healy, Andrew Papanicolaou, and George Papanicolaou. PCA for Implied Volatility Surfaces. *The Journal of Financial Data Science*, 2(2):85–109, 2020.

[8] Katia Amrit Babbar. *Aspects of stochastic implied volatility in financial markets*. PhD thesis, Imperial College London, 2001.

[9] Alberto Bemporad, Leonardo Bellucci, and Tommaso Gabbriellini. Dynamic option hedging via stochastic model predictive control based on scenario simulation. *Quantitative Finance*, 14(10):1739–1751, 2014.

[10] Sana Ben Hamida and Rama Cont. Recovering volatility from option prices by evolutionary optimization. *Journal of Computational Finance*, 8(4):43–76, 2005.

[11] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, USA, 1996.

[12] Siddharth Bhatia, Arjit Jain, and Bryan Hooi. ExGAN: Adversarial Generation of Extreme Samples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6750–6758, May 2021.

[13] Zsolt Bitvai and Trevor Cohn. Day trading profit maximization with multi-task learning and technical analysis. *Machine Learning*, 101:187–209, 2015.

[14] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347, 2021.

[15] Douglas G Bonett and Thomas A Wright. Sample size requirements for estimating Pearson, Kendall and Spearman correlations. *Psychometrika*, 65:23–28, 2000.

[16] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[17] H. Buehler, L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[18] Jay Cao, Jacky Chen, Soroush Farghadani, John Hull, Zissis Poulos, Zeyu Wang, and Jun Yuan. Gamma and Vega Hedging Using Deep Distributional Reinforcement Learning. *Frontiers in Artificial Intelligence*, 6:1129370, 2023.

[19] Jay Cao, Jacky Chen, and John Hull. A neural network approach to understanding implied volatility movements. *Quantitative Finance*, 20(9):1405–1413, 2020.

[20] Luca Capriotti and Mike Giles. 15 years of Adjoint Algorithmic Differentiation (AAD) in finance. *Quantitative Finance*, 24(9):1353–1379, 2024.

[21] Rene Carmona, Yi Ma, and Sergey Nadtochiy. Simulation of Implied Volatility Surfaces via Tangent Lévy Models. *SIAM Journal on Financial Mathematics*, 8(1):171–213, 2017.

[22] Peter Carr and Dilip B. Madan. Towards a theory of volatility trading. In Robert J. Elliott and Phelim P. Boyle, editors, *Volatility: New Estimation Techniques for Pricing Derivatives*, pages 417–427. Risk Books, London, 2001.

[23] Andrew P Carverhill and Terry HF Cheuk. Alternative neural network approach for option pricing and hedging. *Available at SSRN 480562*, 2003.

[24] Angelo Casolaro, Vincenzo Capone, Gennaro Iannuzzo, and Francesco Camastra. Deep learning for time series forecasting: Advances and open problems. *Information*, 14(11):598, 2023.

[25] CBOE. Volatility Index Methodology: Cboe Volatility Index. `https://cdn.cboe.com/api/global/us_indices/governance/VIX_Methodology.pdf`, 2022. [Online; accessed 8-May-2023].

[26] Adil Rengim Cetingoz and Charles-Albert Lehalle. Synthetic data for portfolios: A throw of the dice will never abolish chance. *arXiv preprint arXiv:2501.03993*, 2025.

[27] Siu-Ming Cha and Laiwan Chan. Trading signal prediction. In *International Conference on Neural Information Processing—ICONIP*, pages 842–846. Citeseer, 2000.

[28] Boyang Chen, Zongxiao Wu, and Ruoran Zhao. From fiction to fact: the growing role of generative AI in business and finance. *Journal of Chinese Economic and Business Studies*, 21(4):471–496, 2023.

[29] Fei Chen and Charles Sutcliffe. Pricing and Hedging Short Sterling Options Using Neural Networks. *Intelligent Systems in Accounting, Finance and Management*, 19(2):128–149, 2012.

[30] Minshuo Chen, Renyuan Xu, Yumin Xu, and Ruixun Zhang. Diffusion Factor Models: Generating High-Dimensional Returns with Factor Structure, 2025. arXiv preprint arXiv:2504.06566.

[31] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[32] Tianqi Chen. Introduction to boosted trees. *University of Washington Computer Science*, 2014.

[33] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[34] Samuel N. Cohen, Christoph Reisinger, and Sheng Wang. Detecting and Repairing Arbitrage in Traded Option Prices. *Applied Mathematical Finance*, 27(5):345–373, 2020.

[35] Samuel N. Cohen, Christoph Reisinger, and Sheng Wang. Estimating Risks of European Option Books Using Neural Stochastic Differential Equation Market Models. *Journal of Computational Finance*, 26(3):33–72, 2022.

[36] Samuel N Cohen, Christoph Reisinger, and Sheng Wang. Hedging option books using neural-SDE market models. *Applied Mathematical Finance*, 29(5):366–401, 2022.

[37] Samuel N. Cohen, Christoph Reisinger, and Sheng Wang. Arbitrage-Free Neural-SDE Market Models. *Applied Mathematical Finance*, 30(1):1–46, 2023.

[38] Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001.

[39] Rama Cont. Model uncertainty and its impact on the pricing of derivative instruments. *Mathematical Finance*, 16(3):519–547, 2006.

[40] Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. Tail-GAN: Learning to Simulate Tail Risk Scenarios. *Management Science*, 2025.

[41] Rama Cont and José da Fonseca. Dynamics of implied volatility surfaces. *Quantitative Finance*, 2(1):45–60, 2002.

[42] Rama Cont, Jose da Fonseca, and Valdo Durrleman. Stochastic models of implied volatility surfaces. *Economic Notes*, 31(2):361–377, 2002.

[43] Rama Cont and Yu Hang Kan. Dynamic hedging of portfolio credit derivatives. *SIAM Journal on Financial Mathematics*, 2(1):112–140, 2011.

[44] Rama Cont and Milena Vuletić. Data-driven hedging with generative models. *Available at SSRN 5282525*, 2025.

[45] Rama Cont and Milena Vuletić. Simulation of Arbitrage-Free Implied Volatility Surfaces. *Applied Mathematical Finance*, 30(2):94–121, 2023.

[46] Christa Cuchiero, Wahid Khosrawi, and Josef Teichmann. A generative adversarial network approach to calibration of local stochastic volatility models. *Risks*, 8(4):101, 2020.

[47] Mark HA Davis and David G Hobson. The range of traded option prices. *Mathematical Finance*, 17(1):1–14, 2007.

[48] Joseba Iñaki De La Peña, Iván Iturricastillo, Rafael Moreno, Francisco Román, and Eduardo Trigo. Towards an immunization perfect model? *International Journal of Finance & Economics*, 26(1):1181–1196, 2021.

[49] David A. Dickey and Wayne A. Fuller. Likelihood ratio statistics for autoregressive time series with a unit root. *Econometrica*, 49(4):1057–1072, 1981.

[50] Doris Dobi. *Modeling Systemic Risk in The Options Market*. PhD thesis, Department of Mathematics, New York University, 2014.

[51] Bernard Dumas, Jeff Fleming, and Robert E Whaley. Implied volatility functions: Empirical tests. *The Journal of Finance*, 53(6):2059–2106, 1998.

[52] Bruno Dupire. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

[53] Ricard Durall, Avraam Chatzimichailidis, Peter Labus, and Janis Keuper. Combating Mode Collapse in GAN training: An Empirical Analysis using Hessian Eigenvalues. In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), Volume 4*, pages 211–218, 01 2021.

[54] Lei Fan and Justin Sirignano. Machine learning methods for pricing financial derivatives. *arXiv preprint arXiv:2406.00459*, 2024.

[55] Christian P Fries. Stochastic automatic differentiation: automatic differentiation for Monte-Carlo simulations. *Quantitative Finance*, 19(6):1043–1059, 2019.

[56] H. Föllmer and M. Schweizer. Hedging by sequential regression: An introduction to the mathematics of option trading. *ASTIN Bulletin*, 18(2):147–160, 1988.

[57] Jim Gatheral. *The Volatility Surface: A Practitioner's Guide*. John Wiley & Sons, Hoboken, NJ, 2011.

[58] Jim Gatheral and Antoine Jacquier. Arbitrage-free SVI volatility surfaces. *Quantitative Finance*, 14(1):59–71, 2014.

[59] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NeurIPS 2014)*, pages 2672–2680, 2014.

[61] Ian J. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. `https://arxiv.org/abs/1701.00160`, 2017. NeurIPS 2016 Tutorial. arXiv:1701.00160.

[62] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[63] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 5769–5779. Curran Associates, Inc., 2017.

[64] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503, 2023.

[65] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer New York, New York, NY, 2nd edition, 2009.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[67] Ronald Heynen. An empirical investigation of observed smile patterns. *Review of Futures Markets*, 13:317–317, 1994.

[68] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural Networks for Machine Learning, Lecture 6. Coursera, 2012.

[69] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[70] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[71] Arthur E Hoerl and Robert W Kennard. Ridge regression—1980: Advances, algorithms, and applications. *American Journal of Mathematical and Management Sciences*, 1(1):5–83, 1981.

[72] Blanka Horvath, Jonathan Plenk, and Milena Vuletić. Market Generators: A Paradigm Shift in Financial Modelling. In Christian Bayer, Goncalo dos Reis, Blanka Horvath, and Harald Oberhauser, editors, *Signature Methods in Finance*, chapter 4. Springer, 2025.

[73] Ferenc Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?, 2015. arXiv preprint arXiv:1511.05101.

[74] James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49(3):851–889, 1994.

[75] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[76] Nicolas Jackson, Endre Suli, and Sam Howison. Computation of deterministic volatility surfaces. *Journal of Computational Finance*, 2(2):5–32, 1999.

[77] Adel Javanmard, Jingwei Ji, and Renyuan Xu. Multi-task dynamic pricing in credit market with contextual information. *Available at SSRN 4993594*, 2024.

[78] Gyeeun Jeong and Ha Young Kim. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138, 2019.

[79] Michael Kamal and Jim Gatheral. Implied volatility surface. In Rama Cont, editor, *Encyclopedia of Quantitative Finance*. John Wiley & Sons, Ltd, 2010.

[80] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[81] Patrick Kidger. *On Neural Differential Equations*. Phd thesis, University of Oxford, 2021.

[82] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707. Curran Associates, Inc., 2021.

[83] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[84] Diederik P. Kingma and Max Welling. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[85] Alireza Koochali, Peter Schichtel, Andreas Dengel, and Sheraz Ahmed. Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks–ForGAN. *IEEE Access*, 7:63868–63880, 2019.

[86] Adriano Koshiyama, Stefano B. Blumberg, Nick Firoozye, Philip Treleaven, and Sebastian Flennerhag. QuantNet: transferring learning across trading strategies. *Quantitative Finance*, 22(6):1071–1090, 2022.

[87] Sohyeon Kwon and Yongjae Lee. Can GANs Learn the Stylized Facts of Financial Time Series? In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 126–133, 2024.

[88] Damien Lamberton, Huyên Pham, and Martin Schweizer. Local risk-minimization under transaction costs. *Mathematics of Operations Research*, 23(3):585–612, 1998.

[89] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1558–1566, New York, New York, USA, 20–22 Jun 2016. PMLR.

[90] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In Gökhan Bakir, Thomas Hofmann, Bernhard Schölkopf, Alex Smola, Ben Taskar, and S.V.N. Vishwanathan, editors, *Predicting Structured Data*, pages 1–59. MIT Press, 2006.

[91] David Kuo Chuen Lee, Chong Guan, Yinghui Yu, and Qinxu Ding. A comprehensive review of generative AI in finance. *FinTech*, 3(3):460–478, 2024.

[92] Mark T. Leung, Hazem Daouk, and An-Sing Chen. Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of Forecasting*, 16(2):173–190, 2000.

[93] Xiaoyue Li, A. Sinem Uysal, and John M. Mulvey. Multi-period portfolio optimization using model predictive control with mean-variance and risk parity frameworks. *European Journal of Operational Research*, 299(3):1158–1176, 2022.

[94] Yannick Limmer and Blanka Horvath. Robust Hedging GANs: Towards Automated Robustification of Hedging Strategies. *Applied Mathematical Finance*, 31(3):164–201, 2024.

[95] Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348, 2014.

[96] Michael Ludkovski. Statistical machine learning for quantitative finance. *Annual Review of Statistics and Its Application*, 10(1):271–295, 2023.

[97] Eva Lütkebohmert, Thorsten Schmidt, and Julian Sester. Robust deep hedging. *Quantitative Finance*, 22(8):1465–1480, 2022.

[98] Harry Markowitz. The utility of wealth. *Journal of Political Economy*, 60(2):151–158, 1952.

[99] Claude Martini and Arianna Mingone. No Arbitrage SVI. *SIAM Journal on Financial Mathematics*, 13(1):227–261, 2022.

[100] Fabio Mercurio and Ton Vorst. Option pricing with hedging at fixed trading dates. *Applied Mathematical Finance*, 3(2):135–158, 1996.

[101] Fabio Mercurio and Ton Vorst. Options pricing and hedging in discrete time with transaction costs. *Mathematics of Derivative Securities*, 15:190, 1997.

[102] Ebikella Mienye, Nobert Jere, George Obaido, Ibomoiye Domor Mienye, and Kehinde Aruleba. Deep learning in finance: A survey of applications and techniques. *AI*, 5(4):2066, 2024.

[103] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets, 2014. arXiv preprint arXiv:1411.1784.

[104] Rudy Morel, Stéphane Mallat, and Jean-Philippe Bouchaud. Path shadowing Monte Carlo. *Quantitative Finance*, 24(9):1199–1225, 2024.

[105] James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434, 1963.

[106] Brian Ning, Sebastian Jaimungal, Xiaorong Zhang, and Maxime Bergeron. Arbitrage-free implied volatility surface generation with variational autoencoders. *SIAM Journal on Financial Mathematics*, 13(4):1214–1240, 2022.

[107] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *Advances in Neural Information Processing Systems*, 29, 2016.

[108] OptionMetrics. IvyDB US Reference Manual. `https://wrds-www.wharton.upenn.edu/documents/1504/IvyDB_US_Reference_Manual_rn2hAXz.pdf`, 2021. Version 5.0.

[109] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. Technical Report.

[110] Daniel Poh, Stephen Roberts, and Stefan Zohren. Transfer Ranking in Finance: Applications to Cross-Sectional Momentum with Data Scarcity, 2023.

[111] Eghbal Rahimikia and Felix Drinkall. Re(Visiting) Large Language Models in Finance. *Available at SSRN*, 2024.

[112] Douglas A. Reynolds. Gaussian Mixture Models. In Stan Z. Li and Anil K. Jain, editors, *Encyclopedia of Biometrics*, pages 659–663. Springer, Boston, MA, 2009.

[113] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[114] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16:119–147, 1991.

[115] R Tyrrell Rockafellar. Solving stochastic programming problems with risk measures by progressive hedging. *Set-Valued and Variational Analysis*, 26:759–768, 2018.

[116] L. C. G. Rogers and M. R. Tehranchi. Can the implied volatility surface move by parallel shifts? *Finance Stochastics*, 14(2):235–248, 2010.

[117] Johannes Ruf and Weiguan Wang. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance*, 24(1):1–46, 2020.

[118] Johannes Ruf and Weiguan Wang. Hedging with linear regressions and neural networks. *Journal of Business & Economic Statistics*, 40(4):1442–1454, 2024.

[119] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[120] Philipp J Schönbucher. A market model for stochastic implied volatility. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 357(1758):2071–2092, 1999.

[121] Martin Schweizer. Variance-optimal hedging in discrete time. *Mathematics of Operations Research*, 20(1):1–32, 1995.

[122] Martin Schweizer and Johannes Wissel. Arbitrage-free market models for option prices: The multi-strike case. *Finance and Stochastics*, 12(4):469–505, 2008.

[123] Piet Sercu and Xueping Wu. Cross and Delta-Hedges: Regression Versus Price-Based Hedge Ratios. *Journal of Banking & Finance*, 24(5):735–757, 2000.

[124] William F Sharpe. Mutual fund performance. *The Journal of Business*, 39(1):119–138, 1966.

[125] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

[126] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

[127] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

[128] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527:121261, 2019.

[129] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[130] Chih-Fong Tsai, Yuah-Chiao Lin, David C. Yen, and Yan-Min Chen. Predicting stock returns by classifier ensembles. *Applied Soft Computing*, 11(2):2452–2459, 2011.

[131] Ruey S Tsay. *Analysis of Financial Time Series*. John Wiley & Sons, 2005.

[132] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR.

[133] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[134] Milena Vuletić and Rama Cont. VolGAN: A generative model for arbitrage-free implied volatility surfaces. *Applied Mathematical Finance*, 31(4):203–238, 2024.

[135] Milena Vuletić and Mihai Cucuringu. GraFiN-Gen: graph-based ensemble generative modelling for multi-asset forecasting, 2025. Available at SSRN 5317725.

[136] Milena Vuletić, Felix Prenzel, and Mihai Cucuringu. Fin-GAN: forecasting and classifying financial time series via generative adversarial networks. *Quantitative Finance*, 24(2):175–199, 2024.

[137] Kevin T Webster. *Handbook of price impact modeling*. Chapman and Hall/CRC, 2023.

[138] Magnus Wiese, Lianjun Bai, Ben Wood, and Hans Buehler. Deep hedging: learning to simulate equity option markets, 2019. arXiv preprint arXiv:1911.01700.

[139] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant GANs: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

[140] Johannes Stefan Wissel. *Arbitrage-free market models for liquid options*. PhD thesis, ETH Zurich, 2008.

[141] Qiong Wu, Jian Li, Zhenming Liu, Yanhua Li, and Mihai Cucuringu. Symphony in the Latent Space: Provably Integrating High-dimensional Techniques with Non-linear Machine Learning Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10361–10369, 2023.

[142] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. *arXiv preprint arXiv:2112.07804*, 2021.

[143] Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. COT-GAN: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems*, 33:8798–8809, 2020.

[144] Rui Ye and Qun Dai. A novel transfer learning framework for time series forecasting. *Knowledge-Based Systems*, 156:74–99, 2018.

[145] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Generative Adversarial Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[146] Majid Zanjirdar. Overview of portfolio optimization models. *Advances in Mathematical Finance and Applications*, 5(4):419–435, 2020.

[147] Wenyong Zhang, Lingfei Li, and Gongqiu Zhang. A two-step framework for arbitrage-free prediction of the implied volatility surface. *Quantitative Finance*, 23(1):21–34, 2023.