Volatility surface completion using score-based generative models

by

Ying Kit Hui

A research project
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

An implied volatility surface (IVS) is an important tool in finance that allows comparison between options with different strikes and maturities, and more importantly their valuation. However, due to data scarcity or insufficient market transactions, there are missing points in the observable market IVS. In this research project, we propose employing score-based generative models, a novel generative model originated from image generation, to estimate the missing implied volatilities while simultaneously ensuring no-arbitrage conditions. We demonstrate the plausibility of this approach by training score-based models on synthetic Heston-model generated IVS and achieving an average relative error on missing volatilities below 0.5%.

Acknowledgements

I would like to thank my supervisor, Prof. Justin Wan, whose guidance and support throughout this research project were invaluable. Without him, this report would not have been possible. I would also like to thank Prof. David Saunders for his precious time to be the second reader of this paper.

I also owe so much to the friends that I made over the course of the Master's program.

Finally, I must thank my family.

Dedication

This is dedicated to my grandma, my mum, and my high school physics teacher Mr.Yam.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The implied volatility surface is a widely used and valuable tool for pricing, hedging, and comparing financial derivatives. A one-to-one correspondence between an implied volatility surface and a call option price surface exists [6]. Consequently, the values and availability of a (market) implied volatility surface depend on the market transactions of the associated call options. When only a limited number of transactions have occurred in the market, we can obtain only a few points in the implied volatility surface, necessitating estimations of the missing implied volatilities. This problem is referred to as "the volatility completion problem" [5]. Addressing the volatility completion problem enables more accurate pricing of options, efficient evaluations of traders' portfolios, and a better understanding of the market's expectations regarding the fluctuations of the underlying asset.

Traditional methods for solving the volatility completion problem involve modeling the underlying asset price dynamics using stochastic differential equations. For example, the Black-Scholes model [6] assumes that asset prices follow a geometric Brownian motion. More complex models include stochastic volatility models [13], stochastic volatility with jumps [4], local volatility models [9], rough volatility models [12], and others. The dynamics of the asset price under these models are controlled by their model parameters. As a result, these models require calibration, *i.e.*, finding the optimal model parameters so that the output option prices from these models are closest to the observed market option prices under a specific distance metric.

With recent advancements in machine learning, many deep learning techniques have been employed to address the volatility completion problem. For example, the authors of [15, 21, 22] propose using neural networks, specifically feedforward multilayer neural networks, to approximate the pricing maps of traditional models. This approach significantly

reduces calibration time, thereby enhancing the efficiency and applicability of traditional methods. This line of thought is further extended in [3], where a neural network is used to directly approximate the model parameters, bypassing calibration entirely. In contrast, other efforts [5, 24] propose directly learning the implied volatility surface using variational autoencoders, without making assumptions about asset price dynamics.

Although these data-driven methods can improve the efficiency of traditional model-based approaches, as pointed out in [3], the option prices produced by artificial neural networks by default do not guarantee no-arbitrage. In fact, completing the volatility surface is not a simple estimation problem. There are specific mathematical constraints [23] that an implied volatility surface must satisfy to be free of static arbitrage. As demonstrated in [23], even methods not involving machine learning techniques, such as jump models and parametric models like stochastic volatility inspired representation [11], may not ensure no-arbitrage. Despite this common pitfall, the authors of [1] attempt to address it by incorporating soft constraints into the training objective of the artificial neural network, thereby ensuring that the output surface is arbitrage-free. However, their machine learning framework remains model-dependent, requiring a choice of how to model the underlying asset price dynamics.

In this research paper, we propose adopting the perspective of viewing each implied volatility surface as an image and employing score-based generative models, a novel machine learning tool originally developed for image generation and inpainting, to address the volatility completion problem. By leveraging the separated processes of training and sampling in score-based generative models, we develop a modification to the sampling process that ensures no arbitrage. Furthermore, we can provide a numerical estimation of the deviation of an implied volatility surface from the no-arbitrage conditions, thus offering a guarantee of the quality of the output implied volatility surface.

Another advantage of this approach is that it is model-free, *i.e.,* it does not require modeling the underlying asset price. In fact, we can train the score-based generative models using purely historical data. Additionally, by incorporating implied volatility surfaces produced by different option pricing models into the training dataset, our proposed method allows an easy mixture of option pricing models. Moreover, since our method directly solves the volatility completion problem without the need for calibration, once our score network is trained, the volatility completion task can be performed quickly. Lastly, our proposed method is flexible in the sense that it does not assume any patterns in the missing implied volatilities. We have evaluated our method on synthetic datasets, achieving an average relative percent error of order 10^{-4} in the interpolation setting and a maximum relative error less than 0.5% in an extreme setting where 80% of volatilities are assumed to be missing.

The rest of this research paper is organized as follows. Chapter 2 covers the necessary background, focusing on the definition of implied volatility, no-arbitrage conditions, and most importantly, the fundamental concepts of score-based generative models. Chapter 3 formulates the volatility completion problem precisely and motivates our perspective on viewing an implied volatility surface as an image. Moreover, it explains in detail the inpainting algorithm and how we adapt it to ensure no arbitrage. It also provides the model framework of our approach and briefly discusses how to tune our score network. Chapter 4 presents the synthetic datasets' generation process and their details. Furthermore, it presents the performance of our proposed method in solving the volatility completion problem.

Chapter 2

Background

In this chapter, we aim to cover the necessary background of this research project. We will define what implied volatility is, and discuss no-arbitrage conditions on an implied volatility surface. Next, we will describe in detail what score-based generative modeling is, and how and why it could be employed to sample from the true data distribution via Langevin dynamics. We will also discuss the training and inference process for the noise conditional score network.

2.1 Definition of Implied Volatility

An option is a type of financial derivative that has been widely used for leveraging, market speculation, or risk-management purpose. A call (put) option is a contract that gives its owner the right, not the obligation, to buy (sell) 1 unit of the underlying asset, such as stock, index, or commodity, at a specified price (strike) $K \geq 0$ and at a specified time (maturity) $T > 0$[1]. The seller of a call (put) option can be viewed as an insurance provider such that the option owner can benefit from the upside (downside) movement of the underlying without bearing risk from the downside (upside) movement. Thus, a price must be paid to the option seller for the buyer to acquire the purchasing/selling right. This is the option price V. Finding the fair value of V is called option pricing.

[1]The definition given here refers to European options. The American options are similarly defined with the difference that they can be exercised at any time on or before the maturity T. In this paper, we restrict ourselves to European options.

Different models have been proposed to solve the option pricing problem. The famous Black-Scholes model [6] assumes the price of the underlying at time t, denoted by S_t, follows the geometric Brownian motion under the risk-free measure:

$$dS_t = rS_t dt + \sigma S_t dZ_t, \tag{2.1}$$

where r is the interest rate, σ is a scalar parameter called volatility that needs to be calibrated, and Z_t is the standard Brownian motion. Note that volatility σ describes how fluctuating the price path S_t will be[2]. With some other additional assumptions, they then derive that the call option price V should satisfy the following Black-Scholes equation with the terminal condition representing the payoff:

$$\begin{aligned} &\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \\ &V(S,T) = \max(S-K, 0) \end{aligned} \tag{2.2}$$

where $V(S,t)$ refers to the option price at time t with the price of the underlying at time t given by S, K is the strike price and T is the maturity of the call option. The only unobservable parameter in the Black-Scholes model is the volatility σ and there is a one-to-one correspondence between the option price and the volatility parameter σ. Implied volatility refers to the choice of volatility parameter that is consistent with the market option price. It is a critical tool in finance that can allow comparison between options with different strikes and maturities and is more preferred by practitioners than the option price V itself. Here we give its definition rigorously.

For simplicity, we consider a call option with strike $K \geq 0$, and time to maturity $T > 0$, on a fixed underlying with its current price denoted by S_0. We also assume that the risk-free interest rate r is constant throughout the option's life and that the underlying asset does not pay dividends. Let C_{BS} denote the price of this option as calculated by the Black-Scholes formula [6], a solution to (2.2). Let $C_{\text{mkt}}(K,T)$ be the observed market price of the European call option. Then the implied volatility $\sigma_{\text{BS}}(K,T)$ of this option is defined as the solution of the following equation[3]:

$$C_{\text{BS}}(S_0, K, T, r, \sigma_{\text{BS}}(K,T)) = C_{\text{mkt}}(K,T). \tag{2.3}$$

In particular, the Black-Scholes formula of this European call option is given by:

$$C_{\text{BS}}(S_0, K, T, r, \sigma) = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

[2]One naive way to choose the volatility parameter σ is to use the standard deviation of the historical return of the underlying. The so-obtained value is called historical volatility.

[3]Note that the uniqueness and existence of $\sigma_{\text{BS}}(K,T)$ is guaranteed if $C_{\text{mkt}}(K,T) \in [\max(0, S_0 - Ke^{-rT}), S_0)$.

where $d_1 = \frac{1}{\sigma\sqrt{T}}\left[\ln\left(\frac{S_0}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T\right]$, $d_2 = d_1 - \sigma\sqrt{T}$ and Φ is the c.d.f. of standard normal. Given the value of $C_{\text{BS}}(S_0, K, T, r, \sigma)$, there is no analytic formula that allows us to recover σ from $C_{\text{BS}}, S_0, K, T, r$. Hence, root-finding methods like the bisection method, Newton's method, Brent's method, and the recent machine-learning method [22] have been employed to solve the equation (2.3) numerically.

In this paper, we assume the underlying asset is fixed, we also call the function $(K, T) \mapsto \sigma_{\text{BS}}(K, T)$ implied volatility surface. We can recover the market option prices $C_{\text{mkt}}(K, T)$ from the implied volatility surface $\sigma_{\text{BS}}(K, T)$. So implied volatility surface can be viewed as a preferred metric that reflects the market option prices at a given time point.

2.2 No-arbitrage conditions

Historically, implied volatility surface has come in different shapes, leading to so-called volatility skew or volatility smile. However, the implied volatility surface $\sigma_{\text{BS}}(K, T)$ does not come in arbitrary shape. One constraint is that the market option prices $C_{\text{mkt}}(K, T)$ associated with it should not allow any arbitrage opportunities since in a rational market such opportunities are assumed to be exploited right away and disappear. In fact, there are mathematical conditions [1, 11, 23] that an implied volatility surface has to satisfy so that there is no static arbitrage, *i.e.*, a risk-free profit obtained by trading (without any re-balancing) the options that are associated with $\sigma_{\text{BS}}(K, T)$.

There are different formulations of the no-arbitrage conditions. For implementation convenience, we choose the one from [11], as summarized in [1]. We make an additional assumption that the underlying asset of the options has no dividend. We define the log-moneyness k of the option with strike K and maturity T to be

$$k = k(K, T) = \log(K/S_0) - rT.$$

For a given option with maturity T, we can recover the strike K from the log-moneyness k by $K = S_0 e^{k+rT}$. Log-moneyness k provides a measure of how deviant the strike K is from the current price of underlying S_0 that is independent of the value of S_0. With the correspondence between k and K in mind, from now on whenever we use k and K together, we assume they are correlated as such. We also define the total implied variance w for the option with the strike-maturity pair (K, T) by

$$w(k, T) = (\sigma_{\text{BS}}(S_0 e^{k+rT}, T))^2 \cdot T = (\sigma_{\text{BS}}(K, T))^2 \cdot T,$$

noting that w is a function of log-moneyness k and maturity T.

Theorem 2.2.1 ([1, 11, 23]). *For the total implied variance surface $(k,T) \mapsto w(k,T)$, if the following conditions are satisfied, then there is no static arbitrage on the options associated with $w(k,T)$:*

1. $w(k,T) > 0$
2. $w(k,0) = 0$
3. $w(\cdot,T)$ *is twice-differentiable*
4. $\ell_{\text{cal}}(k,T) = \partial_T \omega(k,T) \geq 0$.
5. $\ell_{\text{but}}(k,T) = \left(1 - \frac{k\partial_k \omega(k,T)}{2\omega(k,T)}\right)^2 - \frac{\partial_k \omega(k,T)}{4}\left(\frac{1}{\omega(k,T)} + \frac{1}{4}\right) + \frac{\partial^2_{kk}\omega(k,T)}{2} \geq 0$
6. $w(k,T)/(T|k|) = \sigma^2(k,T)/|k| < 2$ *as* $k \to \pm\infty$.

The first three conditions are basic conditions that any reasonable implied volatility surface must satisfy. The fourth and fifth conditions are needed to ensure the absence of calendar spread arbitrage and butterfly arbitrage, which are defined in Appendix A.1. The sixth condition is a technical condition that is slightly stronger than necessary to make the proof easier. Moreover, the sixth condition is concerned with the large log-moneyness behavior of total implied variance, which is often impossible to measure or approximate due to the finite amount of transactions in the market. In this paper, we will focus on ensuring the fourth and fifth conditions are satisfied, *i.e.*, to ensure no calendar spread arbitrage and butterfly arbitrage for the implied volatility surface we constructed.

2.3 Score-Based Generative Modeling

In this research essay, we are going to compute the implied volatility surface using machine learning techniques. In particular, we will consider a score-based generative model. In this section, we will formulate the goal of generative modeling, define what is score and explain why it could be useful in a generative model, define what is score-based generative model, and describe its training process and inference process. According to [8], the score-based generative model is one of the three types of diffusion models [25], whose formulation is inspired by Langevin Dynamics. In our paper, we will focus solely on score-based generative models. For alternative types of diffusion models, readers can find them in [7, 14, 26, 31]. The following treatment on score-based generative models are largely extracted from [27, 28].

In generation problem, we are given a dataset of N samples $\mathscr{D} := \{x_i \in \mathbb{R}^D\}_{i=1}^N$, where each x_i are i.i.d. D-dimensional samples from the unknown data distribution $p_{\text{data}}(x)$. The goal of generative modeling is to learn a model from the dataset $\mathscr{D}$ which allows us to generate samples from $p_{\text{data}}(x)$. We also define the (Stein) score of a probability density $p(x)$ to be $\nabla_x \log p(x)$, the gradient of the log-density at the data point x. The information of the score function $\nabla_x \log p(x)$ can enable us to sample from $p(x)$ via Langevin Dynamics.

2.3.1 Naive Langevin dynamics

Recall that our goal is to sample from the unknown data distribution $p_{\text{data}}(x)$. The traditional approach is to build a statistical model $p_\theta(x) \approx p_{\text{data}}(x)$. We have to ensure the normalization property is satisfied by p_θ, *i.e.*, $\int p_\theta(x)dx = 1$. This is a big challenge since the integration over $x \in \mathbb{R}^D$ is particularly hard for large D. To sidestep this difficulty, [27] proposed to model the score $\nabla_x \log p(x)$. In fact, the score is agnostic as to whether the given probability density is normalized or not. Suppose $\tilde{p}(x)$ is an unnormalized probability density with normalization constant $Z := \int \tilde{p}(x)dx \neq 1$. The score is given by:

$$\nabla_x \log \frac{\tilde{p}(x)}{Z} = \nabla_x \log \tilde{p}(x) - \nabla_x \log Z = \nabla_x \log \tilde{p}(x).$$

We see that the score does not require the value of Z. In the next paragraph, we will discuss the mathematical framework – Langevin dynamics, that allows us to sample from a probability density $p(x)$ given its score $\nabla_x \log p(x)$.

For a probability density $p(x)$ where $x \in \mathbb{R}^D$, Langevin dynamics is an iterative method to sample from $p(x)$ using the score $\nabla_x \log p(x)$. We fix a small step size $\alpha > 0$ and a step number M. Suppose we have some initial sample point $\tilde{x}_0$. Theoretically, the choice of $\tilde{x}_0$ does not matter. In fact, we can sample $\tilde{x}_0$ from any prior distribution $\pi(x)$. A common choice will be $\pi(x) = \mathcal{N}(0, I_D)$, a multivariate normal distribution with mean vector $0 \in \mathbb{R}^D$ and covariance matrix given by $D \times D$ identity matrix I_D. Langevin dynamics provides the following update rule for $k = 0, 1, \ldots, M-1$:

$$\tilde{x}_{k+1} = \tilde{x}_k + \alpha \nabla_x \log p(\tilde{x}_k) + \sqrt{2\alpha} z_k, \text{where } z_k \sim \mathcal{N}(0, I_D). \tag{2.4}$$

The z_k are i.i.d samples from $\mathcal{N}(0, I_D)$, also independent from $\tilde{x}_0$. Intuitively, we use the score (gradient of the log-density) to drive the sample points toward regions with high probability while adding noise z_k to ensure there is an appropriate amount of exploration. Under some technical conditions, Langevin dynamics tells us that we can recover the

probability density $p(x)$ by $\rho_\infty(x)$: the stationary distribution of $\tilde{x}_M$ when $M \to \infty$, $\alpha \to 0$.

In reality, we can only approximate the case of $T \to \infty$, $\alpha \to 0$. In the case of $T < \infty, \alpha > 0$, a Metropolis-Hastings update rule can be employed to reduce the error of the sampling process (2.4), at the expense of requiring the information of the transition probability density $q(x'|x)$[4]. We define β by

$$\beta := \min\left\{1, \frac{\pi(\tilde{x}_{k+1})q(\tilde{x}_k \mid \tilde{x}_{k+1})}{\pi(\tilde{x}_k)q(\tilde{x}_{k+1} \mid \tilde{x}_k)}\right\}. \tag{2.5}$$

We also generate $u \sim U[0,1]$. Thus, a Metropolis-Hastings update rule reads as: if $\beta \leq u$, then the proposal is accepted and we set $\tilde{x}_{k+1} = \tilde{x}_k + \alpha\nabla_x \log p(\tilde{x}_k) + \sqrt{2\alpha}z_k$; else the proposal is rejected and we set $\tilde{x}_{k+1} = \tilde{x}_k$. The authors of [28] suggested that the error of not employing the Metropolis-Hastings update rule is often negligible in practice. While we follow this guide and assume the error of not employing it is negligible for our choice of α, T, we still describe the Metropolis-Hastings update rule here since it inspires our modified sampling process from Langevin dynamics that can ensure no-arbitrage conditions. More details can be found in Section 3.2.2.

We observe that the only quantity we need to know in the sampling process (2.4) is the score function $\nabla_x \log p(x)$. To solve the generation problem, *i.e.*, to learn a model from the dataset $\mathscr{D} = \{x_i \in \mathbb{R}^D\}_{i=1}^N$ so that we can sample from $p_{\text{data}}(x)$, one approach is that we train a score network $s_\theta : \mathbb{R}^D \to \mathbb{R}^D$, parameterized by θ, such that $s_\theta(x) \approx \nabla_x \log p_{\text{data}}(x)$. This task is called score matching. Assuming such s_θ can be trained successfully, then we can sample from $p_{\text{data}}(x)$ using the Langevin Dynamics (2.4), with $\nabla_x \log p(\tilde{x}_k)$ replaced by $s_\theta(\tilde{x}_k)$. In the next section, we will describe how score matching can be effectively done.

2.3.2 Score matching

Observant readers might notice that s_θ should be a conservative vector field since the score function $\nabla_x \log p_{\text{data}}(x)$ is a gradient of a scalar-valued function. Practically this constraint can be lifted without affecting the performance of model training and sampling [27]. Without this constraint, we can directly approximate the score function $\nabla_x \log p_{\text{data}}(x)$, instead of first approximating $\log p_{\text{data}}(x)$ and then taking the gradient of it. The naive objective

[4]Precisely, $q(x'|x)$ means the probability density of obtaining x' as a new update by (2.4) given that x is the last sample point. In particular, we have $q(x'|x) \propto \exp\left(-\frac{1}{4\alpha}\left\|x' - x - \alpha\nabla \log p(x)\right\|_2^2\right)$.

function to minimize is hence:

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\|s_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|_2^2\right]. \tag{2.6}$$

However, we do not have access to $\nabla_x \log p_{\text{data}}(x)$. Fortunately, it has been shown [16] that minimizing the objective function (2.6) is equivalent to minimizing the following:

$$\mathbb{E}_{p_{\text{data}}}\left[\text{tr}\left(\nabla_x s_\theta(x)\right) + \frac{1}{2}\|s_\theta(x)\|_2^2\right], \tag{2.7}$$

where $\nabla_x s_\theta(x)$ denotes the Jacobian of $s_\theta(x)$. However, the computation of $\text{tr}(\nabla_x s_\theta(x))$ is expensive and not scalable to higher dimension D.

There are two popular methods for large-scale score matching: denoising score matching [33] and sliced score matching [30]. The former method perturbs the underlying data distribution by adding noise and transforms the objective (2.7) so that the costly term $\text{tr}(\nabla_x s_\theta(x))$ is avoided, at the expense that it is actually estimating the score of the noise-perturbed distribution. In contrast, the latter method employs random projections to estimate the score of the unperturbed true data distribution p_{data}, at the expense of 4 times more computations [28]. While both methods are applicable to our training, our experiments on the task of volatility surface completion show that denoising score matching performs better. Moreover, as will be shown more clearly in Section 2.3.3, denoising score matching fits naturally to how we motivate the construction of a noise conditional score network. Hence, we adopt the denoising score matching and will only present it in this paper.

Denoising score matching first adds noise to the true data distribution $p_{\text{data}}(x)$. Then, instead of approximating the score function of $p_{\text{data}}(x)$, we now approximate the score function of the noise-perturbed distribution, where we could circumvent the expensive and unscalable calculation of $\text{tr}(\nabla_x s_\theta(x))$ for reasons below. Specifically, given the datapoint $x \sim p_{\text{data}}(x)$, we define noise-perturbed distribution conditional on x by $q_\varphi(\tilde{x}|x) = \mathcal{N}(x, \varphi^2 I_D)$, where φ is the noise level[5]. Hence the noise-perturbed distribution probability density is given by $q_\varphi(\tilde{x}) = \int q_\varphi(\tilde{x}|x)p_{\text{data}}(x)dx$. Assuming φ is small enough that $q_\varphi(x) \approx p_{\text{data}}(x)$ and so that $\nabla_x \log q_\varphi(x) \approx \nabla_x \log p_{\text{data}}(x)$, then we can employ score matching to estimate the score of $q_\varphi(\tilde{x})$ as a proxy of estimating $\nabla_x \log p_{\text{data}}(x)$. It is shown in [33] that the objective of approximating the score of $q_\varphi(\tilde{x})$ is equivalent to minimizing the following:

$$\frac{1}{2}\mathbb{E}_{q_\varphi(\tilde{x}|x)p_{\text{data}}(x)}\left[\|s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\varphi(\tilde{x} \mid x)\|_2^2\right]. \tag{2.8}$$

[5]The usual notation for noise is σ. However, since σ has been used for volatility, we use φ to denote noise instead.

Moreover, the optimal network, denoted as $s_{\theta^*}(x)$, that minimizes (2.8) satisfies $s_{\theta^*}(x) = \nabla_x \log q_\varphi(x) \approx \nabla_x \log p_{\text{data}}(x)$ almost surely. Note that the objective (2.8) involves taking the expectation first over $x \sim p_{\text{data}}(x)$ and then the conditional noise-perturbed distribution $\tilde{x} \sim q_\varphi(\tilde{x}|x)$. Moreover, the term being considered is $\nabla_{\tilde{x}} \log q_\varphi(\tilde{x} \mid x)$ instead of $\nabla_{\tilde{x}} \log q_\varphi(\tilde{x})$ so that we can exploit the analytical form of $\nabla_{\tilde{x}} \log q_\varphi(\tilde{x} \mid x)$ to scale up score matching. In fact, since we have chosen $q_\varphi(\tilde{x}|x) = \mathcal{N}(x, \varphi^2 I_D)$, $\nabla_{\tilde{x}} \log q_\varphi(\tilde{x} \mid x) = -(\tilde{x} - x)/\varphi^2$, which can be easily calculated for any dimension D. Thus, making use of the analytical form of $\nabla_{\tilde{x}} \log q_\varphi(\tilde{x} \mid x)$ we can scale up score matching by solving (2.8).

2.3.3 Noise Conditional Score Network: Training

It is tempting to apply denoising score matching to train a score network for a sufficiently small φ. However, in practice, using only one small noise level will suffer from the problem of inaccurate score estimation in the regions of low data density and slow mixing of Langevin Dynamics in (2.4) [28]. Precisely, for multi-modal data distribution with low density in between, the naive application of denoising score matching requires a very large T and a very small α to correctly recover the relative weights of the modes. In contrast, if a large noise level φ is employed, the low data density regions will be filled and these concerns will be addressed. But in this case, we no longer have $q_\varphi(x) \approx p_{\text{data}}(x)$. We see that there is a trade-off between the sampling quality of $x \sim q_\varphi(x)$ versus whether the noise-perturbed distribution $q_\varphi(x)$ is close enough to the true data density $p_{\text{data}}(x)$ when we are choosing a single noise level φ.

To address this dilemma, [28] proposed to perturb the data using a sequence of decreasing noise levels, obtaining a sequence of noise-perturbed distribution that converges to data distribution p_{data}, and to train a noise conditional score network (NCSN) that keeps track of the scores of each noise level. Thus, we can use Langevin Dynamics (2.4) to sample from a perturbed distribution with large noise, and gradually anneal down to a lower noise level. Intuitively, we can make use of the intermediate perturbed distribution and confer the benefits of score matching on large noise levels to smaller noise levels, hence improving the efficiency of Langevin Dynamics in sampling an approximate $p_{\text{data}}(x)$.

We will now describe the training objective of a noise conditional score network. We consider a geometric sequence of decreasing noise levels: $\varphi_1 > \varphi_2 > \ldots > \varphi_L$ with $\frac{\varphi_{i+1}}{\varphi_i} = \gamma < 1$. The appropriate choices of hyperparameters: initial large noise φ_1, final noise φ_L, and the number of noise levels L are problem-dependent. More details can be found in Section 3.4 on how to tune them. Note that these hyperparameters are not trained by the gradient descent method, only θ is. We assume here that the choices of φ_1, φ_L, L are good

so that it allows accurate score matching and good sampling properties. For a given noise φ, we denote the noise-perturbed data distribution by $q_\varphi(x) = \int p_{\text{data}}(t)\mathcal{N}(x|t, \varphi^2 I_D)dt$. Then a noise conditional score network $s_\theta(\cdot,\cdot)$[6] is trained so that $\forall\varphi \in \{\varphi_i\}_{i=1}^L : s_\theta(x,\varphi) \approx \nabla_x \log q_\varphi(x)$. We can then employ denoising score matching to each noise level φ_i and train one single network $s_\theta(\cdot,\cdot)$ that minimizes a weighted average of denoising score matching objectives of each noise level.

To be precise, we denote the denoising score matching objective of $s_\theta(\cdot,\cdot)$ for a given φ by

$$\begin{aligned}\ell(\theta;\varphi) :&= \frac{1}{2}\mathbb{E}_{q_\varphi(\tilde{x}|x)p_{\text{data}}(x)}\left[\|s_\theta(\tilde{x},\varphi) - \nabla_{\tilde{x}}\log q_\varphi(\tilde{x}\mid x)\|_2^2\right] \\ &= \frac{1}{2}\mathbb{E}_{p_{\text{data}}(x)}\mathbb{E}_{\tilde{x}\sim\mathcal{N}(x,\varphi^2 I_D)}\left[\left\|s_\theta(\tilde{x},\varphi) + \frac{\tilde{x}-x}{\varphi^2}\right\|_2^2\right].\end{aligned}$$

Thus, a noise conditional score network $s_\theta(\cdot,\cdot)$ is trained to minimize the following unified objective for the whole sequence of noise levels $\{\varphi_i\}_{i=1}^L$:

$$\mathcal{L}(\theta;\{\varphi_i\}_{i=1}^L) := \frac{1}{L}\sum_{i=1}^L \varphi_i^2\ell(\theta;\varphi_i) \tag{2.9}$$

$$= \frac{1}{2L}\sum_{i=1}^L \varphi_i^2\mathbb{E}_{q_{\varphi_i}(\tilde{x}|x)p_{\text{data}}(x)}\left[\|s_\theta(\tilde{x},\varphi_i) - \nabla_{\tilde{x}}\log q_{\varphi_i}(\tilde{x}\mid x)\|_2^2\right]. \tag{2.10}$$

Note that the weights for each $\ell(\theta;\varphi_i)$ are chosen to be φ_i^2. This is motivated by the empirical observation [28] that such a choice can lead to the order of magnitude of $\varphi_i^2\ell(\theta;\varphi_i)$ being independent of φ_i. When training the noise conditional score network, the expectations in (2.9) are estimated using empirical averages.

2.3.4 Noise Conditional Score Network: Inference

After we used the given dataset $\mathscr{D} := \{x_i \in \mathbb{R}^D\}_{i=1}^N$ to train a noise conditional score network $s_\theta(\cdot,\cdot)$ that minimizes the unified objective (2.9), we can then use $s_\theta(\cdot,\cdot)$ to sample from p_{data} via a procedure called annealed Langevin dynamics (ALD) [28], which addresses the problem of inaccurate score estimation and slow mixing of naive Langevin dynamics.

[6]Note that there are two arguments to $s_\theta(\cdot,\cdot)$. The first argument is the usual input x and the second argument is the noise level φ, which adds flexibility to the network to approximate the score on each noise level.

In general, the higher the noise φ, the better the mixing rate and score matching for the noise-perturbed distribution q_φ. The idea of annealed Langevin dynamics (Algorithm 1) is to exploit the sampling advantage of a larger noise level φ_i, obtaining a good sample $\tilde{x}_M$ from q_{φ_i} using simple Langevin dynamics with step size α_i. This good sample $\tilde{x}_M$ will likely be residing in the high-density region of q_{φ_i}. Then since the difference of successive noises φ_i, φ_{i+1} is small, the good sample $\tilde{x}_M$ from q_{φ_i} will also likely be residing in the high-density region of $q_{\varphi_{i+1}}$. Since score estimation and the mixing of Lagenvin dynamics work better in the high-density region, the good sample $\tilde{x}_M$ from q_{φ_i} will provide a good initial sample point for sampling from $q_{\varphi_{i+1}}$ using simple Langevin dynamics with a reduced step size α_{i+1}. Repeating this process for $i = 1, 2, \ldots, L$, we obtain a good final sample $\tilde{x}_M$ from $q_{\varphi_L} \approx p_{\text{data}}$. The choice of step size $\alpha_i = \epsilon \cdot \varphi_i^2/\varphi_L^2$ is motivated so that the ratio between the terms $\alpha_i s_\theta(\tilde{x}_{t-1}, \varphi_i)$ and $\sqrt{2\alpha_i} z_t$ in Algorithm 1 does not depend on φ_i [28], *i.e.*, the signal (score) to noise ratio is roughly a constant across each noise level.

Algorithm 1 Annealed Langevin dynamics (ALD)

Input: $\{\varphi_i\}_{i=1}^L, M, \epsilon, s_\theta(\cdot, \cdot)$
▷ Noise Levels, number of sampling steps per noise scale, step size parameter, trained NCSN

Output: $\tilde{x}_M \sim q_{\varphi_L} \approx p_{\text{data}}$ ▷ Sample from $q_{\varphi_L} \approx p_{\text{data}}$

Initialize $\tilde{x}_0 \sim \mathcal{N}(0, I_D)$ ▷ The prior distribution of $\tilde{x}_0$ could also be $\mathcal{U}[0, 1]$
for $i = 1, 2, \ldots, L$ **do**
  $\alpha_i = \epsilon \cdot \varphi_i^2/\varphi_L^2$ ▷ α_i is the step size for sampling q_{φ_i}
  for $t = 1, 2, \ldots, M$ **do**
    $z_t \sim \mathcal{N}(0, I_D)$ ▷ Sample independently from all previous random samples
    $\tilde{x}_t = \tilde{x}_{t-1} + \alpha_i s_\theta(\tilde{x}_{t-1}, \varphi_i) + \sqrt{2\alpha_i} z_t$
  end for
  $\tilde{x}_0 = \tilde{x}_M$
  ▷ $\tilde{x}_M$ is a good sample from q_{φ_i} and is a good initial point for sampling $q_{\varphi_{i+1}}$
end for

To provide a specific example, let us consider the dataset $\mathscr{D}$ as the MNIST dataset. In this case, each element $x_i \in \mathscr{D}$ is a 28×28 matrix containing pixel values, which represent grayscale images of digits ranging from 0 to 9. Then, the true data distribution, p_{data} over $x \in \mathbb{R}^{28 \times 28}$ represents images of handwritten digits between 0 and 9. The output $\tilde{x}_M$ from Algorithm 1 will be a good and new sample from p_{data}, meaning that $\tilde{x}_M$, when represented as an image, just looks like a handwritten integer from 0 to 9, while not identical to any $x_i \in \mathscr{D}$. It is important to note that we input nothing but noise to the score network $s_\theta(\cdot, \cdot)$

for the generation of $\tilde{x}_M$, which involves repeated Langevin updates, as described in line 6 of Algorithm 1.

Chapter 3

Methodology

In this chapter, we will explore the application of score-based generative models to a problem related to implied volatility surfaces. In particular, we will treat each implied volatility surface as an image, and our aim is to estimate missing volatilities by filling in the gaps in these images, much like an inpainting task, where missing parts of an image are filled to form a complete image. We will then discuss in detail how a score-based generative model can perform inpainting and how we adapt it to ensure no-arbitrage conditions. Furthermore, we will delve into the methods we employ to tune the hyperparameters of a score-based generative model, thereby enhancing its performance. Throughout this chapter, we will follow the notations established in Chapter 2.

3.1 Volatility completion problem

3.1.1 General formulation

Recall in Chapter 2 that the implied volatility surface refers to the function $(K, T) \mapsto \sigma_{\text{BS}}(K, T)$. When it is plotted graphically, it is a surface with its shape usually[1] constrained by Theorem 2.2.1. The market implied volatility surface $(K, T) \mapsto \sigma_{\text{BS}}(K, T)$ is determined by the bid and ask occurring in the market. Due to the limited transactions in the market, we might need to estimate the missing implied volatilities based on known implied volatilities at other points. Hence we will formulate the volatility surface computation as a completion problem:

[1] Unless there is a market inefficiency where a static arbitrage can be obtained.

Problem 1 (Volatility completion problem, general form)**.** *Given the implied volatilities* $\{\sigma_{BS}(K_i, T_i)\}_{i=1}^N$ *of European call options with different strikes and maturities, all on the same underlying. Based on that information, how can we estimate the implied volatility* $\sigma_{BS}(K, T)$ *for a strike-maturity pair* $(K, T) \notin \{(K_i, T_i)\}_{i=1}^N$.

Formulating the volatility surface computation as a completion problem enables us to develop a systematic approach to estimate missing implied volatilities. Unlike traditional option models that place a significant emphasis on modeling the dynamics of the asset price path S_t, this formulation focuses on the estimation of missing data. This data-driven perspective makes the problem more amenable to machine-learning approaches. In the next section, motivated by practical interest, we will further refine the problem, resulting in a form where score-based generative models can be applied.

3.1.2 Image perspective

The general formulation of the volatility completion problem, as presented earlier, does not take into account the distribution of strike prices K_i and maturities T_i. In other words, it lacks constraints that reflect the real-world patterns of trading activity, which predominantly occur for specific strike-maturity pairs. The absence of these constraints could make the estimation of implied volatilities unnecessarily difficult, less accurate, and less relevant for practical applications.

In practice, options with strike prices K that significantly deviate from the current underlying price S_0 are traded less frequently, as well as options with maturities T greater than 1 year. Therefore, it is important to refine the volatility completion problem by focusing on strike-maturity pairs that are more relevant to actual trading activity. To this end, we consider an increasing sequence of strikes $K_1 < K_2 < \ldots < K_n$ with $K_1 = e^{-1}S_0$ and $K_n = e^1 S_0$, limiting the problem to options with strikes relatively close to S_0. We also consider an increasing sequence of maturities $0 < T_1 < T_2 < \ldots < T_m$ with $T_m = 1$, further limiting the problem to options with a maturity of at most one year.

To connect this practical consideration with the original volatility completion Problem 1, we introduce a fixed grid of strikes and maturities, denoted by $\Delta := \{(K_i, T_j)\}_{i=1,j=1}^{n,m}$. Thus, we will only consider options with a strike-maturity pair $(K_i, T_j) \in \Delta$. We also extend Problem 1 slightly by allowing estimating multiple missing volatilities at once. Hence, the problem can be described as: given implied volatilities for some grid points in Δ, how can we estimate the unknown implied volatilities associated with the remaining grid points in Δ?

To be precise, following the notation in [15], for any subset A of Δ, we denote the implied volatilities associated with strike-maturity pairs in A by $\Sigma_{\text{BS}}(A) := \{\sigma_{\text{BS}}(K_i, T_j) \mid (K_i, T_j) \in A\}$. Thus, the complete discretized implied volatility surface is just $\Sigma_{\text{BS}}(\Delta) = \{\sigma_{\text{BS}}(K_i, T_j)\}_{i=1,j=1}^{n,m}$. Using subset A to denote the set of strike-maturity pairs to which implied volatilities are missing, the general formulation of the volatility completion Problem 1 can be refined to the following:

Problem 2 (Volatility completion problem, image form)**.** *Fix a subset A of Δ. Assume that we are given $\Sigma_{BS}(\Delta \setminus A)$. Based on this information, how can we estimate $\Sigma_{BS}(A)$, thus obtaining the complete implied volatility surface $\Sigma_{BS}(\Delta)$?*

In this paper, we propose that we can view the complete implied volatility surface $\Sigma_{\text{BS}}(\Delta)$ as an image with $n \cdot m$ pixels. This is a significant insight that allows machine learning methods, in particular, score-based generative models, which have been well-developed for image inpainting, to be extended to performing implied volatility completion.

Score-based generative models are capable of reconstructing an incomplete image of human faces to the point there is no visible difference to humans [27]. The image of the implied volatility surface $\Sigma_{\text{BS}}(\Delta)$, described visually, is just a picture consisting of gradually changing grey colors, which is way less complicated than human faces. Inspired by the success of image inpainting capability of score-based generative models on human faces, we will explore the use of score-based generative models for implied volatility surfaces. Another advantage of using score-based generative models is that its sampling algorithm is more amenable than other generative models like generative adversarial networks. As a result, we are able to modify it to ensure no-arbitrage conditions.

Transforming Problem 1 to Problem 2, we can treat it as an image inpainting problem, to which score-based generative models can solve. We have mentioned in Chapter 2 how score-based generative models can sample from an unknown data distribution $p_{\text{data}}(x)$. In the coming sections, we will describe how to modify the Algorithm 1 so that it can perform inpainting. Furthermore, we will discuss how we can ensure no-arbitrage conditions by further tweaking the sampling process.

Before moving to the next section, it should also be mentioned that the image point of view is first explored in [15] and has inspired the above formulation. However, the problem at hand here is different from the problem in [15]. They are concerned with speeding up existing option pricing models by training a feedforward multilayer neural network F_θ to approximate the option evaluation map F. In contrast[2], in our setting, the emphasis is given to the supposedly given knowledge $\Sigma_{\text{BS}}(\Delta \setminus A)$ and how we can estimate $\Sigma_{\text{BS}}(A)$

[2]Nonetheless, [15] points out an advantage of the grid-based approach that is also applicable in our

while preserving no-arbitrage on $\Sigma_{\text{BS}}(\Delta)$, leveraging the image inpainting capability of score-based generative models.

3.2 Inpainting algorithm with no-arbitrage conditions

Image inpainting is the task of reconstructing the missing parts of an incomplete image. For simplicity, we assume the image is grey scale. Hence, we can denote the complete image by $x \in \mathbb{R}^{n\times m}$, a $n \times m$ matrix storing the pixel values, where n, m are the height and the width of the image respectively. Let $\Xi \in \{0,1\}^{n\times m}$ be the mask that indicates regions not missing in the incomplete image so that $\Xi_{ij} = 1$ means the pixel at (i, j) position is not missing and $\Xi_{ij} = 0$ means the pixel at (i, j) position is missing. Then the incomplete image is just $x \odot \Xi$, where $\odot$ means element-wise multiplication. The goal of inpainting is to recover x based on the incomplete image $x \odot \Xi$. In the following, we will first discuss the original inpainting algorithm [28] and then we will discuss how to modify it to solve the Problem 2 with an emphasis on ensuring no-arbitrage conditions on $\Sigma_{\text{BS}}(\Delta)$.

3.2.1 Original

Score-based generative models have been developed for image inpainting [27, 28]. Intuitively, it is achieved by modifying the sampling Algorithm 1 to incorporate the signal of the given pixels into the noise conditional score network $s_\theta(\cdot, \cdot)$. During the sampling process 1, we instead hard-code the given pixels to each $\tilde{x}_t$, with the unknown pixels guided by the usual Langevin update. That is, we set $\tilde{x}_t \odot \Xi = x \odot \Xi$ and allow the missing parts $\tilde{x}_t \odot (1 - \Xi)$ to be guided by the usual Langevin update. The conditional score $s_\theta(\tilde{x}_t, \varphi_i)$ so calculated will be guiding us towards the higher probability region of the noise-perturbed distribution with respect to noise φ_i conditioned on the knowledge of the given pixels $x \odot \Xi$, *i.e.*, $q_{\varphi_i}(\tilde{x}|\tilde{x} \odot \Xi = x \odot \Xi)$. By the same logic as in Algorithm 1, through annealing the noise level, we can, in the end, obtain a good sample from $q_{\varphi_L}(\tilde{x}|\tilde{x} \odot \Xi = x \odot \Xi) \approx p_{\text{data}}(\tilde{x}|\tilde{x} \odot \Xi = x \odot \Xi)$.

The annealed Langevin dynamics (ALD) inpainting algorithm is presented in Algorithm 2. For notation convenience, whenever we sample a random number in Algorithm 2, it is

setting. Suppose we prepare our training dataset by the usual Monte Carlo approach, with certain (usually around 100000) stock price sample paths at hand. Then we can easily evaluate options with additional strikes and maturities using the same set of sample paths. This allows an easy refinement on Δ. While we still need to retrain our score model on the implied volatility surfaces corresponding with new Δ, the data preparation step needs not to be redone completely and a significant amount of time is saved.

assumed to be drawn independently from all previous random samples. Moreover, we reuse the symbols with the same definitions from Algorithm 1. Recall that the goal is to recover the complete image x based on the given incomplete image $x \odot \Xi$. Compared to Algorithm 1, Algorithm 2 takes additional inputs: mask Ξ and the incomplete image $x \odot \Xi$. As seen in lines 4 and 5, for each noise level φ_i, we add noise $\tilde{z}$ to $x \odot \Xi$, obtaining $y = x \odot \Xi + \tilde{z}$. This is to make sure that when we hard-code the given pixels value to the post-Langevin sample $\tilde{x}_t$ in line 9, we are still sampling from q_{φ_i}. The ability of Algorithm 2 to complete the image relies on the signal $s_\theta(\tilde{x}_{t-1}, \varphi_i)$ in line 8. Note that $\tilde{x}_{t-1} \odot \Xi = y \odot \Xi = x \odot \Xi + \tilde{z} \odot \Xi \approx x \odot \Xi$. Thus, $s_\theta(\tilde{x}_{t-1}, \varphi_i)$ implicitly contains the information of the given incomplete image. Intuitively, it encodes the direction in which we can make the image more realistic, starting from the incomplete image $x \odot \Xi$ and noise elsewhere.

Algorithm 2 Annealed Langevin dynamics (ALD) inpainting

Input: $\{\varphi_i\}_{i=1}^{L}, M, \epsilon, s_\theta(\cdot, \cdot)$
Input: $\Xi, x \odot \Xi$
▷ Ξ is mask indicating regions not occluded, x is the complete image and so $x \odot \Xi$ is the given incomplete image
Output: $\tilde{x}_M$
▷ Approximate sample from $p_{\text{data}}(\tilde{x} | \tilde{x} \odot \Xi = x \odot \Xi)$. Estimate of x
1: Initialize $\tilde{x}_0 \sim \mathcal{N}(0, I_D)$
2: **for** $i = 1, 2, \ldots, L$ **do**
3: $\quad \alpha_i = \epsilon \cdot \varphi_i^2 / \varphi_L^2$
4: $\quad \tilde{z} \sim \mathcal{N}(0, \varphi_i^2 I_D)$ ▷ Random sample
5: $\quad y = x \odot \Xi + \tilde{z}$ ▷ Consistent with q_{φ_i}
6: $\quad$ **for** $t = 1, 2, \ldots, M$ **do**
7: $\quad\quad z_t \sim \mathcal{N}(0, I_D)$ ▷ Random sample
8: $\quad\quad \tilde{x}_t = \tilde{x}_{t-1} + \alpha_i s_\theta(\tilde{x}_{t-1}, \varphi_i) + \sqrt{2\alpha_i} z_t$
9: $\quad\quad \tilde{x}_t = \tilde{x}_t \odot (1 - \Xi) + y \odot \Xi$
▷ The given pixels are hard-coded to the sample $\tilde{x}_t$ while the missing pixels are guided by the Langevin update
10: $\quad$ **end for**
11: $\quad \tilde{x}_0 = \tilde{x}_M$
12: **end for**

3.2.2 With no-arbitrage conditions

Following our image perspective on the discrete implied volatility surface $\Sigma_{\text{BS}}(\Delta)$ and previous notations in Section 3.2.1, we will use x to denote $\Sigma_{\text{BS}}(\Delta)$, treating $\Sigma_{\text{BS}}(\Delta)$ as an image. Also, the subset A of Δ, recording the positions of missing implied volatilities, can be represented by $1 - \Xi$. Thus, $\Sigma_{\text{BS}}(\Delta \setminus A)$ is just the given incomplete image $x \odot \Xi$ and based on that we want to recover x, *i.e.*, $\Sigma_{\text{BS}}(\Delta)$. Thus we see that Problem 2 is the same inpainting problem that Algorithm 2 is trying to solve.

However, recall that the implied volatility surface $(K, T) \mapsto \sigma_{\text{BS}}(K, T)$ is constrained by Theorem 2.2.1. Thus, to solve Problem 2, we cannot simply adopt the original inpainting Algorithm 2. We have to impose additional constraints to make the resulting estimate of the discrete implied volatility surface $\Sigma_{\text{BS}}(\Delta)$ satisfies Theorem 2.2.1. The authors of [1] proposed to use soft constraints in the training process of their feedforward multilayer neural network. Their approach relies on the fact that their neural network actually takes input (K, T) and outputs an estimate of $\sigma_{\text{BS}}(K, T)$. In other words, they are training a neural network $\tilde{w}_\theta$ directly approximating the implied total variance function w. Thus, by employing twice-differentiable activation functions, they can exactly calculate $\partial_T \tilde{w}_\theta(k, T), \partial_k \tilde{w}_\theta(k, T), \partial_{kk} \tilde{w}_\theta(k, T)$. This allows them to calculate $\ell_{\text{cal}}, \ell_{\text{but}}$ for $\tilde{w}_\theta$ and ensuring no arbitrage in the training process of $\tilde{w}_\theta$. However, the same approach cannot be adopted here since our noise conditional score network $s_\theta(\cdot, \cdot)$ does not directly approximate the implied total variance function $(K, T) \mapsto w(K, T)$. In fact, it is not even a function with variables (K, T).

Instead of modifying the training objective, as was done in [1], we propose here to modify the original inpainting Algorithm 2 to ensure no-arbitrage conditions when estimating $x = \Sigma_{\text{BS}}(\Delta)$. The basic idea is first to define butterfly and calendar spread loss functions B, C to measure how an intermediate estimate $\tilde{x}_t$ of the true implied volatility surface x, as generated by Algorithm 2, deviates from no-arbitrage conditions 4, 5 in Theorem 2.2.1. Inspired by the Metropolis-Hasting update rule (2.5), we propose a no-arbitrage update rule. By using the losses $B(\tilde{x}_t), C(\tilde{x}_t)$, we define an acceptance ratio β. The no-arbitrage update rule will accept $\tilde{x}_t$ as our next sample if β is small enough. Intuitively, the no-arbitrage rule forces the sampling process to explore more the region of probability space that exhibits no-arbitrage and ensures on average our intermediate samples will gradually satisfy no-arbitrage. Since the true implied volatility surface x is in fact free of arbitrage, this is beneficial to solving the volatility surface completion. Now we will describe the method in detail.

For a discrete implied volatility surface $\Sigma_{\text{BS}}(\Delta)$, by making use of the grid $\Delta = \{(K_i, T_j)\}_{i=1,j=1}^{n,m}$, we use finite difference method to approximate $\partial_T w(k, T)$, $\partial_k w(k, T)$,

$\partial_{kk}w(k,T)$. To be precise, for a $\Sigma_{\text{BS}}(\Delta)$, we let k_i be the log-moneyness corresponding to K_i and denote the finite difference approximations by $\tilde{\partial}_T w(k_i,T_j)$, $\tilde{\partial}_k w(k_i,T_j)$, $\tilde{\partial}_{kk}w(k_i,T_j)$. These quantities are given by the following formulae:

$$\begin{aligned}
\tilde{\partial}_T w(k_i,T_j) &:= \frac{w(k_i,T_{j+1})-w(k_i,T_j)}{T_{j+1}-T_j}, \quad 1\le i\le n, 1\le j\le m-1,\\
\tilde{\partial}_k w(k_i,T_j) &:= \frac{w(k_{i+1},T_j)-w(k_i,T_j)}{k_{i+1}-k_i}, \quad 1\le i\le n-1, 1\le j\le m,\\
\tilde{\partial}_{kk} w(k_i,T_j) &:= \frac{\tilde{\partial}_k w(k_{i+1},T_j)-\tilde{\partial}_k w(k_i,T_j)}{k_{i+1}-k_i}, \quad 1\le i\le n-2, 1\le j\le m.
\end{aligned}$$

Note that we can only obtain all three approximations simultaneously on the restricted grid $\Delta_{-2,-1} := \{(K_i,T_j)\}_{i=1,j=1}^{n-2,m-1}$. Based on Theorem 2.2.1, we can then obtain the finite difference approximation of $\ell_{\text{cal}}(k,T), \ell_{\text{but}}(k,T)$ at $(K_i,T_j)\in\Delta_{-2,-1}$ which we denote by $\tilde{\ell}_{\text{cal}}(k_i,T_j), \tilde{\ell}_{\text{but}}(k_i,T_j)$ respectively. Now, we define the butterfly loss function B and the calendar spread loss function C evaluated at $x=\Sigma_{\text{BS}}(\Delta)$ by

$$\begin{aligned}
B(\Sigma_{\text{BS}}(\Delta)) &= \frac{1}{(n-2)(m-1)}\sum_{i=1}^{n-2}\sum_{j=1}^{m-1}\max(\tilde{\ell}_{\text{but}}(k_i,T_j),0),\\
C(\Sigma_{\text{BS}}(\Delta)) &= \frac{1}{(n-2)(m-1)}\sum_{i=1}^{n-2}\sum_{j=1}^{m-1}\max(\tilde{\ell}_{\text{cal}}(k_i,T_j),0).
\end{aligned}$$

Let $\tilde{x}'_t$ denote the proposed update for the intermediate estimate. We first calculate $B_t := B(\tilde{x}'_t), C_t := C(\tilde{x}'_t)$[3]. Then we compare the current losses with the losses of the previous step by computing the following no-arbitrage acceptance ratio:

$$\beta = \max\Big(\frac{B_t}{B_{t-1}+\epsilon'}, \frac{C_t}{C_{t-1}+\epsilon'}\Big), \tag{3.1}$$

where ε' can be any small positive number to avoid division by 0 with default value 0.001. Moreover, we define $B_0=0, C_0=0$ ensuring (3.1) is well-defined when $t=1$.

Then the no-arbitrage update rule is as follows. We first generate a random number $u\sim\mathcal{U}[0,1.2]$. If the acceptance ratio $\beta\le u$, then we accept $\tilde{x}'_t$ as our next sample and set

[3]Note that these losses are saved so that we can plot the quantiles of the losses for each noise level to understand how the proposed samples gradually satisfy the no-arbitrage conditions through annealing the noise level.

$\tilde{x}_t = \tilde{x}'_t$. If $\beta > u$, then we reject the proposed update $\tilde{x}'_t$ and set $\tilde{x}_t = \tilde{x}_{t-1}$. The upper bound 1.2 is chosen so that in some cases we accept $\tilde{x}_t$ even though it might be slightly worse in terms of satisfying no-arbitrage conditions. Intuitively, it allows our no-arbitrage inpainting algorithm to explore a larger region of the probability space.

Additionally, as opposed to Algorithm 2, we do not add noise to the given incomplete image $x \odot \Xi$ when hard-coding these pixels to the intermediate sample. We observe that such a change can slightly improve the performance of estimating missing implied volatilities.

Moreover, after all the sampling steps are done, we add a denoising step setting our final estimation to be $\tilde{x}_M + \varphi_L^2 s_\theta(\tilde{x}_M, \varphi_L)$, as suggested by [18]. This is a technique that can improve the sample quality when applied at the end of Langevin dynamics [18, 27]. As mentioned in [18], if we denote the optimal score network that minimizes (2.9) by $s_{\theta^*}(\cdot, \cdot)$, then the expected denoised sample given a noisy sample $\tilde{x}_M$ can be expressed as

$$\mathbb{E}_{x \sim q_{\varphi_L}(x|\tilde{x}_M)}[x] = \tilde{x}_M + \varphi_L^2 s_{\theta^*}(\tilde{x}_M, \varphi_L).$$

Practically, we also observe improved performance in solving Problem 2 when we employ the denoising step. After the denoising step, for the sake of consistency, we hard-code the given incomplete image $x \odot \Xi$ again.

The no-arbitrage inpainting algorithm is presented in Algorithm 3.

3.3 Model framework

In this section, we will discuss how to apply Algorithm 3 to solve Problem 2, the image version of the original volatility surface problem. In particular, we will describe the framework for generating synthetic datasets, training the model, and finally evaluating the model performance. This will allow the reader to have a better grasp of the results presented in Chapter 4.

Before diving into the steps of the framework, for the architecture of the noise conditional score network $s_\theta(\cdot, \cdot)$, we have followed the U-Net architecture as employed in [28], inspired by its successful performance in the task of semantic segmentation. We only made one minor adjustment to the Refine-Net [20] (a variant of U-Net) architecture of $s_\theta(\cdot, \cdot)$ by reducing the number of filters, *i.e.*, number of feature maps, for layers corresponding to each cascade, accustoming the fact that the volatility surface is not as complicated as human faces.

Algorithm 3 ALD inpainting, with no-arbitrage

Input: $\{\varphi_i\}_{i=1}^{L}, M, \epsilon, s_\theta(\cdot,\cdot)$

Input: $\Xi, x \odot \Xi, \epsilon' = 0.001$

▷ Ξ is mask indicating regions not occluded, x is the complete image and so $x \odot \Xi$ is the given incomplete image, ϵ' can be any small positive number suitable to avoid division by 0.

Output: $\tilde{x}_M$

▷ Approximate sample from $p_{\text{data}}(\tilde{x}|\tilde{x} \odot \Xi = x \odot \Xi)$. Estimate of x

Initialize $\tilde{x}_0 \sim \mathcal{N}(0, I_D)$
for $i = 1, 2, \ldots, L$ **do**
  $\alpha_i = \epsilon \cdot \varphi_i^2 / \varphi_L^2$
  for $t = 1, 2, \ldots, M$ **do**
    $z_t \sim \mathcal{N}(0, I_D)$ ▷ Random sample
    $\tilde{x}'_t = \tilde{x}_{t-1} + \alpha_i s_\theta(\tilde{x}_{t-1}, \varphi_i) + \sqrt{2\alpha_i} z_t$
    $B_t = B(\tilde{x}'_t)$, $C_t = C(\tilde{x}'_t)$
  ▷ Butterfly and calendar losses of the proposal $\tilde{x}'_t$
  ▷ Save B_t, C_t to consider the losses for each noise level
    $\beta = \max(B_t/(B_{t-1} + \epsilon'), C_t/(C_{t-1} + \epsilon'))$ ▷ No-arbitrage acceptance ratio
    $u \sim \mathcal{U}[0, 1.2]$ ▷ Random sample
    if $\beta \leq u$ **then**
      $\tilde{x}_t = \tilde{x}'_t$
    else if $\beta > u$ **then**
      $\tilde{x}_t = \tilde{x}_{t-1}$
    end if
    $\tilde{x}_t = \tilde{x}_t \odot (1 - \Xi) + x \odot \Xi$
  ▷ The given pixels are hard-coded to the sample $\tilde{x}_t$ while the missing pixels are guided by the Langevin update when the acceptance ratio β is small
  end for
  $\tilde{x}_0 = \tilde{x}_M$
end for
$\tilde{x}_M = \tilde{x}_M + \varphi_L^2 s_\theta(\tilde{x}_M, \varphi_L)$ ▷ Denoising Step
$\tilde{x}_M = \tilde{x}_M \odot (1 - \Xi) + x \odot \Xi$ ▷ Hard-code the given pixels again

Given a grid $\Delta := \{(K_i, T_j)\}_{i=1,j=1}^{n,m}$, let $A \subseteq \Delta$ be a set recording the positions of missing implied volatilities. We also define mask $\Xi \in \mathbb{R}^{n\times m}$ associated to A by

$$\Xi_{ij} = \begin{cases} 0, & \text{if } (K_i, T_j) \in A \\ 1, & \text{otherwise.} \end{cases}$$

The data-driven framework for solving the volatility completion problem then involves the following steps:

- Step 1: Generate synthetic discrete implied volatility surfaces $\Sigma_{\text{BS}}(\Delta)$ by existing option pricing models. In particular, we choose a traditional stochastic volatility model – Heston model. Let us denote the Heston model by $\mathcal{M}(\eta)$, where $\eta \in \mathbb{R}^5$ is its model parameters. The model parameters η fully specify the corresponding prices of financial contracts under Heston model. Thus, by solving a Heston model with given model parameters η, we can obtain a synthetic discrete implied volatility surface on Δ which we denote by $\mathcal{M}_\eta(\Delta)$. We randomly sample a sequence of the model parameters of the Heston model, obtaining $\eta_1, \eta_2, \ldots, \eta_N$. Hence, we obtain a dataset $\mathscr{D}_{\text{IVS}} = \{x_i := \mathcal{M}_{\eta_i}(\Delta)\}_{i=1}^N$, where we use the notation x_i to emphasize the image perspective. More details about the Heston model, the dataset $\mathscr{D}_{\text{IVS}}$ and how we sample $\eta_1, \ldots, \eta_N$ will be presented in Section 4.1.

- Step 2: We split the dataset $\mathscr{D}_{\text{IVS}}$ into a training part and a test part, denoted respectively by $\mathscr{D}_{\text{IVS}}^{\text{train}}, \mathscr{D}_{\text{IVS}}^{\text{test}}$. The training part $\mathscr{D}_{\text{IVS}}^{\text{train}}$ and the testing part $\mathscr{D}_{\text{IVS}}^{\text{test}}$ contains respectively $80\%, 20\%$ of the whole dataset $\mathscr{D}_{\text{IVS}}$.

- Step 3: We feed the training dataset $\mathscr{D}_{\text{IVS}}^{\text{train}}$ and train the noise conditional score network $s_\theta(\cdot,\cdot)$ to minimize the unified training objective (2.9). We denote the trained noise conditional score network by $s_{\tilde{\theta}}(\cdot,\cdot)$, where $\tilde{\theta}$ is the updated parameters.

- Step 4: Consider $x \in \mathscr{D}_{\text{IVS}}^{\text{test}}$. We treat it as a complete image. We then employ no-arbitrage inpainting Algorithm 3 with inputs $\{\varphi_i\}_{i=1}^L, M, \epsilon, s_{\tilde{\theta}}(\cdot,\cdot), \Xi, x \odot \Xi$, obtaining $\tilde{x}_M$ – our estimate of the complete image x. We then calculate the relative error $\varepsilon_{\text{rel}}(x) \in \mathbb{R}^{n\times m}$ of our estimate by

$$[\varepsilon_{\text{rel}}(x)]_{i,j} = \begin{cases} \frac{|[\tilde{x}_M]_{ij} - x_{ij}|}{|x_{ij}|}, & \text{if } (K_i, T_j) \in A \\ 0, & \text{otherwise.} \end{cases}$$

- Step 5: Due to limited computational power, we will limit our error calculation on a subset $\{x_1^{\text{test}}, x_2^{\text{test}}, \ldots, x_{100}^{\text{test}}\} \subseteq \mathscr{D}_{\text{IVS}}^{\text{test}}$. For each position (i,j), we will consider the

average, the standard deviation, and the maximum of the set $\{[\varepsilon_{\text{rel}}(x_k^{\text{test}})]_{i,j}\}_{k=1}^{100}$. We will then present these errors by respectively three heat maps. This allows us to evaluate our model performance in solving Problem 2.

3.4 Hyperparamters tuning

There are several hyperparameters of the noise conditional score network and the no-arbitrage Algorithm 3. They include the noise schedule $\{\varphi_i\}_{i=1}^L$, number of sampling steps per noise scale M, step size parameter ϵ, choice of optimizer for minimizing (2.9), numbers of epochs (or equivalently numbers of batches to be scanned) when minimizing (2.9), etc. In this section, we will discuss how we tune and choose them.

The authors of [29] proposed five techniques for choosing the aforementioned hyperparameters. We have adopted three techniques with slight modifications based on model performance in solving Problem 2. They are presented as follows:

Technique 3.4.1 (Initial noise scale)**.** *Choose φ_1 to be as large as the maximum Euclidean Distance between all pairs of training data points.*

Technique 3.4.2 (Other noise scales)**.** *Let $D = n{\cdot}m$, the dimension of the implied volatility surface in the dataset $\mathscr{D}_{IVS}$. Choose $\{\varphi_i\}_{i=1}^L$ as a geometric progression with common ratio γ such that*

$$\Phi(\sqrt{2D}(\gamma-1)+3\gamma) - \Phi(\sqrt{2D}(\gamma-1)-3\gamma) = E := 0.5,$$

where Φ is the c.d.f. of the standard normal.

Note that the choice of E is arbitrary, motivated by practical experiments in image tasks [29]. Moreover, once we have chosen the smallest noise scale φ_L, by making use of the optimal γ as provided by Technique 3.4.2, we can calculate the corresponding L. In fact, recalling that $\gamma = \frac{\varphi_{i+1}}{\varphi_i}$, we have $\varphi_L = \varphi_1(1/\gamma)^{L-1}$ and so $L = -\frac{-\log(\varphi_L/\varphi_1)}{\log(\gamma)} + 1$.

Technique 3.4.3 (Selecting M and ε)**.** *For a given number of sampling steps per noise scale M, we should choose a step size ϵ that makes the following maximally close to* 1*.*

$$\left(1-\frac{\epsilon}{\varphi_L^2}\right)^{2M}\left(\gamma^2 - \frac{2\epsilon}{\varphi_L^2-\varphi_L^2\left(1-\frac{\epsilon}{\varphi_L^2}\right)^2}\right) + \frac{2\epsilon}{\varphi_L^2-\varphi_L^2\left(1-\frac{\epsilon}{\varphi_L^2}\right)^2}$$

It was originally proposed in [29] that one should choose M as large as the computational budget allows. However, our experiments show that it does not necessarily improve our model performance in solving Problem 2. It is plausible since our no-arbitrage inpainting Algorithm 3 deviates from the simple Langevin dynamics. We choose $M = 1000$ because it provides the best model performance among the grid $\{100, 1000, 2000, 4000, 8000, 10000\}$. Note that our limited computational budget does not allow experiments with M of order $\geq 10^5$.

Observant readers might notice that these three techniques do not address how to pick the smallest noise level φ_L. In fact, it is actually the most determining hyperparameter based on the results of our experiments. Intuitively, the smaller φ_L the better since it will make the noise-perturbed distribution q_{φ_L} closer to the true data distribution p_{data}. However, the smaller the φ_L, the larger the L is and hence more terms in the training objective (2.9). This will hinder the training process of the noise conditional score network $s_\theta(\cdot, \cdot)$ resulting inaccurate estimation of scores and hence worse performance in solving Problem 2. By fixing the number of iteration steps (*i.e.*, the number of batches to be scanned) of the minimization process of (2.9) to be 300000, we performed a grid search on different φ_L and adopt the choice $\varphi_L = 0.00001 = 10^{-5}$.

The number of iteration steps 300000 is motivated by that as used in image generation task [29]. We also set the batch size to be 128 and employ the Adam optimizer with a learning rate of 0.001.

Chapter 4

Results

In this chapter, we will discuss in detail how the datasets are generated from the existing option pricing model. In particular, we will generate two datasets with different grid sizes. We will then present our model performance (see Section 3.3 for the model framework) in interpolation and randomized mask settings.

4.1 Dataset details

As mentioned in Section 3.3, we will use the Heston model to generate synthetic discrete implied volatility surfaces $\Sigma_{\text{BS}}(\Delta)$. Recall that the Black-Scholes model assumes a constant volatility σ in (2.1) when modeling the price path S_t. This is a weakness of the Black-Scholes model since its outputted implied volatility surface will be flat and not consistent with the observed market implied volatility surface. By contrast, the Heston model, a type of stochastic volatility models, employs another stochastic differential equation to model the behavior of the volatility, reflecting the volatility skew observed in the financial data. The Heston model [13] employs a system of stochastic differential equations to model the price path S_t and instantaneous variance v_t as follows:

$$dS_t = rS_t dt + \sqrt{v_t} S_t dZ_t^S,\ S_{t_0} = S_0, \tag{4.1}$$

$$dv_t = \kappa(\bar{v} - v_t)dt + \xi\sqrt{v_t} dZ_t^v,\ v_{t_0} = v_0, \tag{4.2}$$

$$dZ_t^S dZ_t^v = \rho dt, \tag{4.3}$$

where Z_t^S, Z_t^v are two Brownian motions for S_t and v_t respectively with correlation coefficient ρ, $\bar{v}$ is the long term variance, κ is the reversion speed of v_t to $\bar{v}$, ξ is the volatility of

the volatility, r is the risk-free interest rate, and S_0, v_0 are the initial values for S_t and v_t respectively.

By standard no-arbitrage arguments [13], it can be shown that the option price V satisfies the following PDE:

$$\begin{aligned} \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\xi Sv\frac{\partial^2 V}{\partial S\partial v} + +\frac{1}{2}\xi^2 v\frac{\partial^2 V}{\partial v^2} + rS\frac{\partial V}{\partial S} \\ + \kappa(\bar{v}-v)\frac{\partial V}{\partial v} - rV + \frac{\partial V}{\partial t} = 0. \end{aligned} \tag{4.4}$$

Note that the option price V does not have an analytical formula and needs to be solved numerically. Common methods to solve (4.4) include Monte Carlo simulation, numerical integration [19], and COS method [10]. These three methods, presented in these papers, involve the use of the Feynman-Kac Theorem stating that the price V can be obtained as the discounted expected value of the payoff of V at its maturity T, under a risk-neutral measure. Moreover, numerical integration and the COS method make further use of the Fourier transform of the risk-neutral density. The applicability of the Feynman-Kac Theorem also ensures the resulting implied volatility surface from the Heston model is free of static arbitrage (*c.f.* Definition 1.2 in [23]).

To generate our synthetic datasets, we will employ numerical integration [19] to ensure that the option price so calculated has a relative error below 10^{-13}. For the datasets we shall generate, for convenience, we will assume $S_0 = 100$ and $r = 0$. Thus, the Heston model (4.1 - 4.3) is completely determined by five model parameters: $\{\rho, \bar{v}, \kappa, \xi, v_0\}$. We use $\eta \in \mathbb{R}^5$ to denote these five model parameters $\eta = [\rho, \bar{v}, \kappa, \xi, v_0]^T$. Using the notation mentioned in Section 3.3, we denote such a fully specified Heston model as $\mathcal{M}(\eta)$. By fixing a grid Δ, we can numerically solve the fully specified Heston model $\mathcal{M}(\eta)$ to obtain a synthetic call price surface. Then by solving the equation (2.3) using a root finding method [17], we obtain a synthetic discrete implied volatility surface $\mathcal{M}_\eta(\Delta)$. We can then generate a synthetic dataset $\mathscr{D}_{\text{IVS}} = \{x_i := \mathcal{M}_{\eta_i}(\Delta)\}_{i=1}^N$ by sampling η_i uniformly from an appropriate range. In particular, we will generate two datasets denoted by $\mathscr{D}_{\text{IVS},1}, \mathscr{D}_{\text{IVS},2}$.

For the first dataset $\mathscr{D}_{\text{IVS},1}$, we consider a 8×8 grid $\Delta_1 = \{(K_i, T_j)\}_{i=1,j=1}^{8,8}$. More precisely, we choose $K_1, \ldots, K_8$ to be 8 equidistant points between (including) $e^{-1}S_0$ to $e^1 S_0$, where $S_0 = 100$ is used for all of our experiments and the second dataset. For maturities, we choose $T_1, \ldots, T_8$ to be 8 equidistant points between (including) 0.5 and 1. We then obtain $100,000$ samples of η using Latin hypercube sampling from the following ranges in Table 4.1.

Furthermore, we impose the Feller condition [2] $2\kappa\bar{v} > \xi^2$ by filtering out some sampled η to ensure the instantaneous variance v_t is strictly positive. We also filter out some choices

Heston parameter	Lower bound	Upper bound
ρ	-0.95	0
$\bar{v}$	0.05	0.5
κ	0.1	2
ξ	0.05	0.5
v_0	0.05	0.5

Table 4.1: Range for Heston parameters η

of η where the associated call price surface contains a call price that is less than 10^{-6} to guarantee that the root-finding method in [17] can be properly applied to obtain $\mathcal{M}_\eta(\Delta_1)$. After applying the above constraints, the resulting dataset $\mathscr{D}_{\text{IVS},1}$ has $91,529$ elements, *i.e.*, $\mathscr{D}_{\text{IVS},1} = \{x_i := \mathcal{M}_{\eta_i}(\Delta_1)\}_{i=1}^{91529}$.

For the second dataset $\mathscr{D}_{\text{IVS},2}$, we consider a finer 16×16 grid $\Delta_2 = \{(K_i, T_j)\}_{i=1,j=1}^{16,16}$. Similarly, we choose $K_1, \ldots, K_{16}$ to be 16 equidistant points between (including) $e^{-1}S_0$ to $e^1 S_0$. For maturities, we choose $T_1, \ldots, T_{16}$ to be 16 equidistant points between (including) 0.2 and 1. So we consider even smaller maturity in this setting. We apply the same sampling method as before. In the end, we obtain a dataset $\mathscr{D}_{\text{IVS},2}$ of size $87,527$.

Note that the dataset $\mathscr{D}_{\text{IVS},2}$ has a finer grid Δ_2. This is a deliberate choice to explore whether a finer grid will affect the model performance or not.

4.2 Experiments

In this section, we will present the performance of our model framework (see Section 3.3) in solving the Problem 2 for the datasets $\mathscr{D}_{\text{IVS},1}, \mathscr{D}_{\text{IVS},2}$ respectively. We will first consider the case where the missing volatilities are assumed to be located in the middle of the image of the implied volatility surface. We will refer to this scenario as Interpolation. After that, we will consider a randomized mask setting, where the locations of the missing volatilities are randomly chosen, thus some of the missing volatilities may be located along the boundary of the grid. This randomized mask setting is more faithful to the market practice since there are not fixed patterns of missing volatilities.

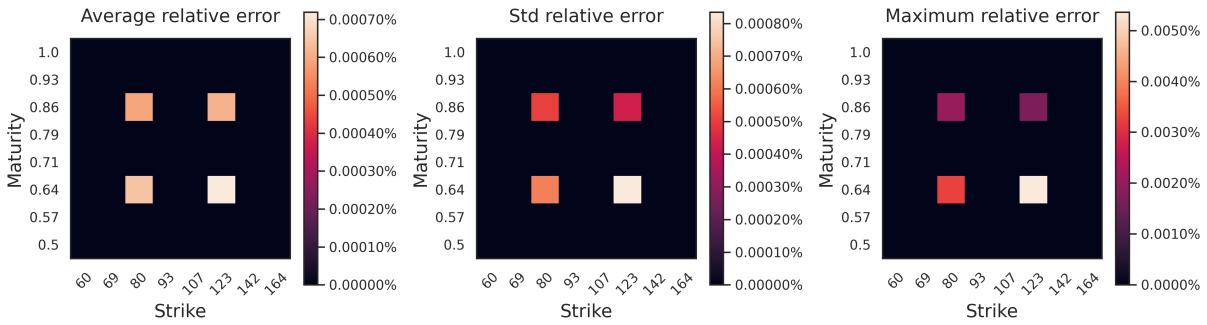


Figure 4.1: 8×8 grid, training dataset size 73,223, errors over 100 test IVS

	Top left	Top right	Bottom left	Bottom right
Average relative errors	0.0005915%	0.0006146%	0.0006451%	0.0007194%
Standard deviations	0.0005045%	0.0004346%	0.0006154%	0.0008349%
Maximum relative errors	0.002046%	0.001681%	0.003260%	0.005362%

Table 4.2: 8×8 grid, training dataset size 73,223, errors over 100 test IVS

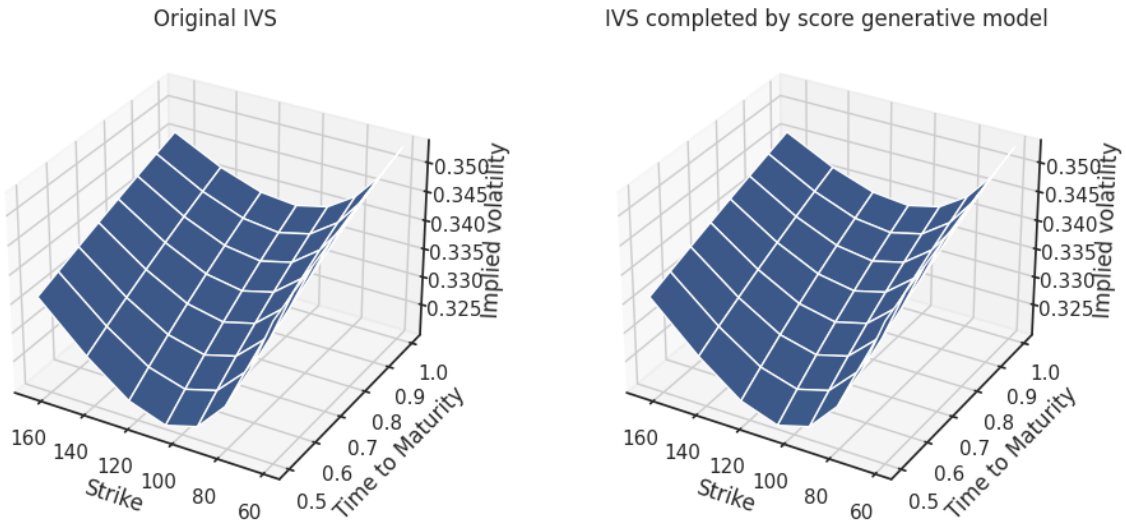
4.2.1 Interpolation, coarse grid

In this experiment, we assume there are 4 missing volatilities located in the middle of the implied volatility surface and we consider the model performance on the dataset $\mathscr{D}_{\text{IVS},1}$ with a coarse 8×8 grid Δ_1.

As mentioned in Section 3.3, we will train our score network on 80% of $\mathscr{D}_{\text{IVS},1}$, resulting a training dataset of size $73,223$ and calculate the testing error on 100 testing implied volatility surfaces. In particular, we present the average, the standard deviation, and the maximum of the relative error set $\{[\varepsilon_{\text{rel}}(x_k^{\text{test}})]_{i,j}\}_{k=1}^{100}$ in Figure 4.1. Note that the colored points in the heat maps represent the assumed missing volatilities in the implied volatility surface. Furthermore, the exact error values are shown in Table 4.2. We see that the average and the standard deviation of the relative percent errors are of the order 10^{-4} while the maximum of the relative percent error is of the order 10^{-3}. This indicates the effectiveness of score-based generative models in interpolating missing volatilities.

We also plot the original and the completed implied volatility surface in Figure 4.2. We observe that the plotted surface is smooth and visibly not differentiable from the true implied volatility surface.

Moreover, we plot the average update rate of the no-arbitrage update rule, as stated



(a) Original IVS

(b) Completed IVS

Figure 4.2: 8×8 grid, training dataset size 73,223. (a) Original IVS. (b) Completed IVS.

in Algorithm 3; see Figure 4.3. It refers to the proportion of samples, out of 100 test IVSs, that have an acceptance ratio $\beta \leq u$ (see lines 8 - 14 in Algorithm 3), hence passing the no-arbitrage update rule, for each noise level φ_i. We observe that the average update rate is around 10% for the initial noise levels and increases to 1 through annealing the noise levels. The low average update rate for the initial large noise levels is expected since with large noise, samples from the perturbed distribution of implied volatility surfaces are heavily affected by randomness. The no-arbitrage rule guides these samples to a region of probability space that exhibits a smaller deviation from no-arbitrage. The convergence of the average update rate to 1 reflects the fact that for small noise levels, most of the samples generated satisfy the no-arbitrage conditions and have zero butterfly and calendar losses.

Furthermore, as mentioned in Section 3.2.2, we plot the 90% quantile of the butterfly and calendar losses B_t, C_t for each noise level of the perturbed distributions in Figure 4.4a - 4.4b. The 90% quantiles are obtained by sorting the losses B_t, C_t for 100 samples we have at each noise level φ_i. We observe that the no-arbitrage conditions are gradually satisfied by the intermediate samples when annealing the noise level, as shown in the decrease of the 90% quantiles to 0. The gradual decrease is accredited to the employment of the no-arbitrage update rule, guiding the sampling process to explore regions of probability space of implied volatility surface that has no-arbitrage property.

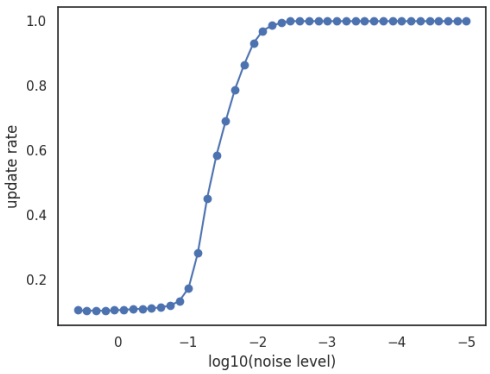
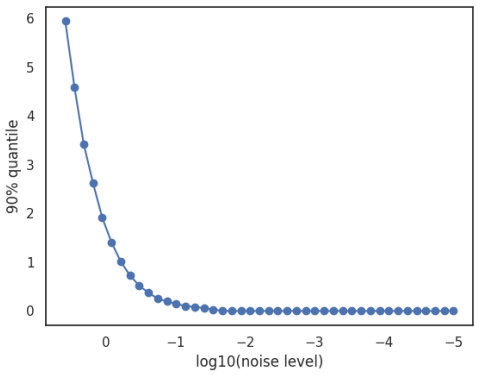
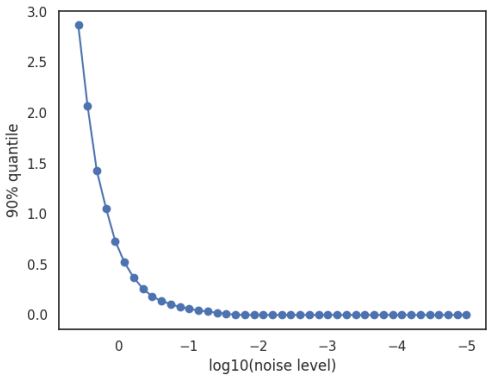


Figure 4.3: 8×8 grid, training dataset size 73,223, average update rate per noise level



(a) Butterfly loss



(b) Calendar loss

Figure 4.4: 8×8 grid, training dataset size 73,223, 90% quantile of losses per noise level. (a) Butterfly loss. (b) Calendar loss.

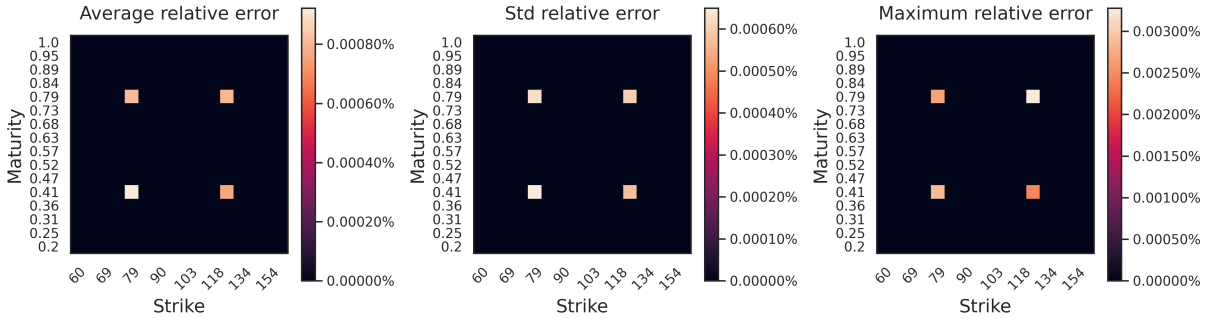


Figure 4.5: 16×16 grid, training dataset size 70,021, errors over 100 test IVS

	Top left	Top right	Bottom left	Bottom right
Average relative errors	0.0008107%	0.0007974%	0.0009226%	0.0007649%
Standard deviations	0.0006258%	0.0006016%	0.0006500%	0.0005785%
Maximum relative errors	0.002680%	0.003279%	0.002865%	0.002448%

Table 4.3: 16×16 grid, training dataset size 70,021, errors over 100 test IVS

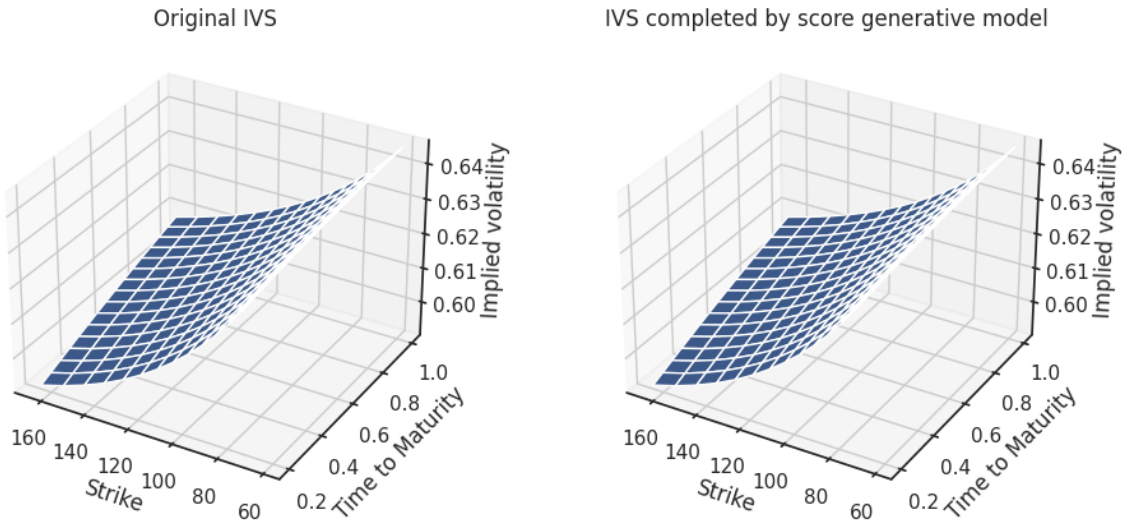
4.2.2 Interpolation, finer grid

This experiment is similar to the one in Section 4.2.1, but we will test it on the dataset $\mathscr{D}_{\text{IVS},2}$ with a finer 16×16 grid Δ_2. Similarly, 80% of $\mathscr{D}_{\text{IVS},2}$ will be used to train the score network, hence the training dataset size is $70,021$.

Figure 4.5 and Table 4.3 demonstrate the average, the standard deviation, and the maximum relative errors of the predicted volatilities, in heat maps and a table respectively. We note that the average and standard deviation of the relative percent errors are of order 10^{-4} and the maximum of the relative errors is of order 10^{-3}. Hence, we see that the errors for 16×16 grid Δ_2 are of the same order as in the case of 8×8 grid Δ_1, the errors for which are shown in Figure 4.1 and Table 4.2. Whether a finer grid would enhance the model performance remains to be verified.

We also plot the original and the completed implied volatility in Figure 4.6. Again, we observe that the plotted surface is smooth and visibly not differentiable from a true implied volatility surface.

Moreover, we plot the average update rate of the no-arbitrage update rule and the 90% quantiles of the butterfly and calendar losses B_t, C_t for each noise level in Figure 4.7 - 4.8. Same as in the case of coarse grid Δ_1, we observe that the average update is around 10% for



(a) Original IVS

(b) Completed IVS

Figure 4.6: 16×16 grid, training dataset size 70,021. (a) Original IVS. (b) Completed IVS.

the initial noise levels and increase to 1 through annealing the noise level. We also observe that the no-arbitrage conditions are gradually satisfied by the intermediate samples when annealing the noise level, as shown in the decrease of the 90% quantiles to 0.

4.2.3 Randomized mask, coarse grid

In this experiment, we consider the model performance on the dataset $\mathscr{D}_{\text{IVS},1}$ with a coarse 8×8 grid Δ_1 under the randomized mask setting. In particular, we assume that the mask Ξ recording the locations of the missing volatilities of the grid Δ_1 is randomly selected with around 50% of Δ_1 being masked.

The average, the standard deviation, and the maximum of the relative errors are presented in Figure 4.9. Even with 50% of Δ_1 assumed to be missing, we can still achieve an average of 0.008% of relative errors with a maximum relative error of around 0.1%. We also note that the model performance on the boundary and especially the corner of the image is generally worse than other parts of the image.

Again, the original and the completed implied volatility surface under this randomized mask setting have no visual difference; see Figure 4.10.

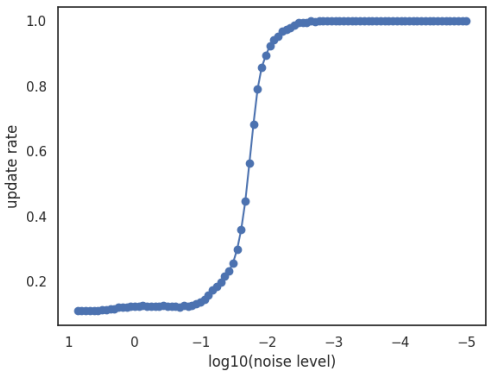
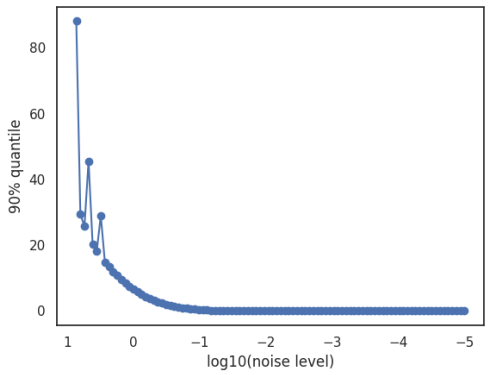
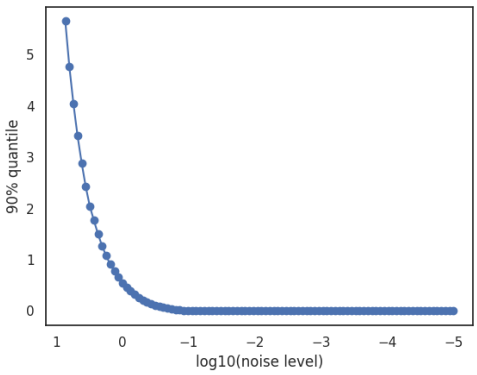


Figure 4.7: 16×16 grid, training dataset size 70,021, average update rate per noise level



(a) Butterfly loss



(b) Calendar loss

Figure 4.8: 16×16 grid, training dataset size 70,021, 90% quantile of losses per noise level. (a) Butterfly loss. (b) Calendar loss.

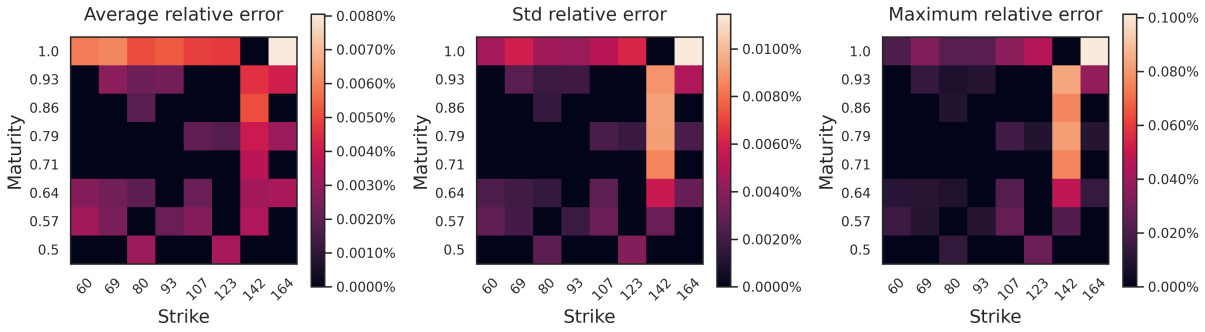
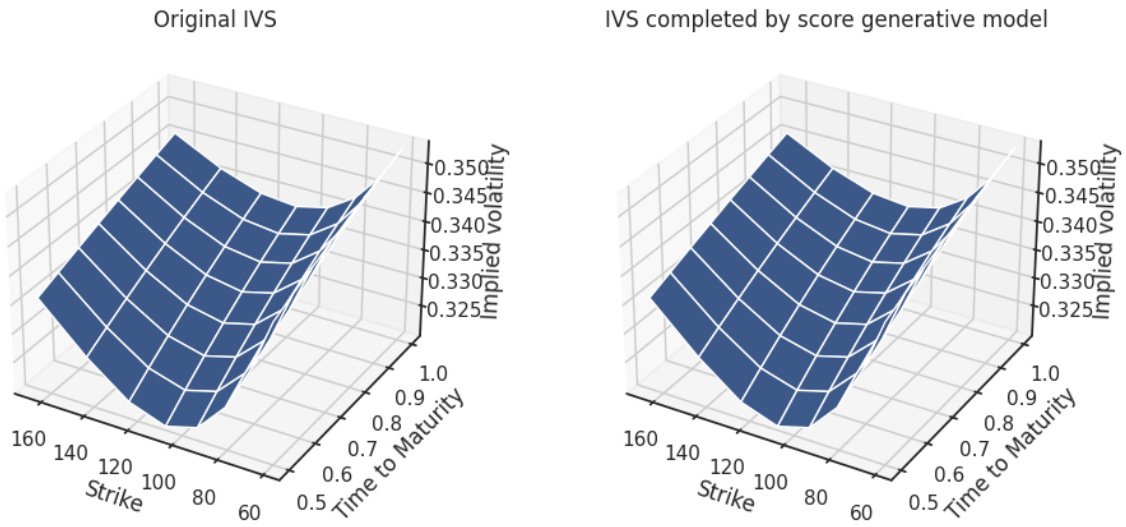


Figure 4.9: 8×8 grid, training dataset size 73,223, errors over 100 test IVS, 50% of the implied volatilities are missing



(a) Original IVS

(b) Completed IVS

Figure 4.10: 8×8 grid, training dataset size 73,223, 50% of the implied volatilities are missing. (a) Original IVS. (b) Completed IVS.

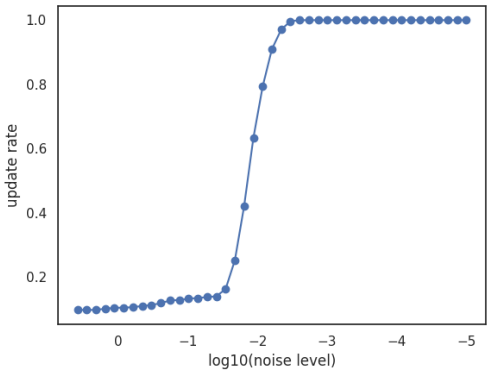


Figure 4.11: 8×8 grid, training dataset size 73,223, 50% of the implied volatilities are missing, average update rate per noise level

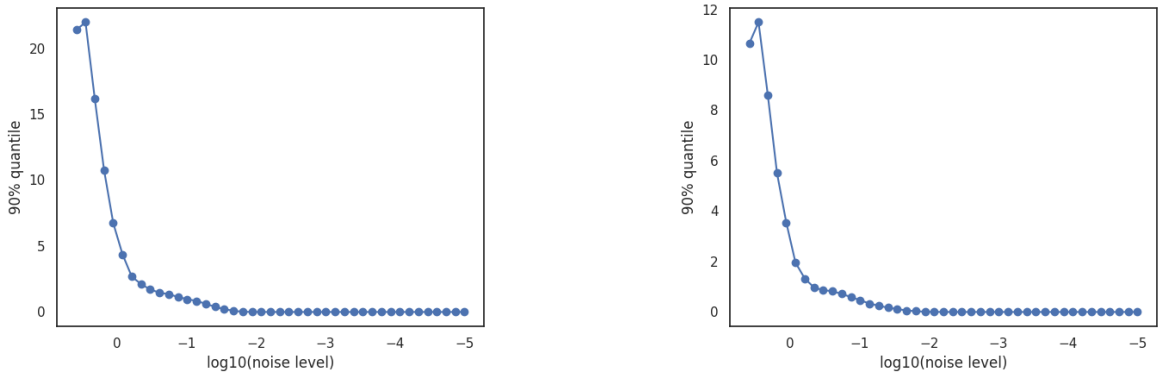
The average update rate of the no-arbitrage rule and the 90% quantiles of the butterfly and calendar losses are shown in Figure 4.11 - 4.12. Compared to the case as displayed in Figure 4.3, we observe that the average update rate in the case of estimating multiple missing volatilities at once, saturates to 1 slower than that in the case of estimating only 4 missing volatilities. As expected, due to the presence of more missing volatilities, the butterfly and calendar losses start with a higher initial value. Nonetheless, the gradual decrease of butterfly and calendar losses to 0 when annealing the noise level is still observed in Figure 4.12a - 4.12b, ensuring no-arbitrage on the final surface.

We would also like to emphasize that when applying a higher percentage of masking, such as 60%, to Δ_1, the maximum relative error increases beyond 1%, leading to an implied volatility surface that no longer appears visually accurate.

4.2.4 Randomized mask, finer grid

In this experiment, we consider the model performance on the dataset $\mathscr{D}_{\text{IVS},2}$ with a finer 16×16 grid Δ_2 under the randomized mask setting. In this case, we assume that the mask Ξ recording the locations of the missing volatilities of the grid Δ_2 is randomly selected with around 80% of Δ_2 being masked, more than the 50% mask in Section 4.2.3.

The average, the standard deviation, and the maximum of the relative errors are presented in Figure 4.9. We achieve an average of 0.012% of relative errors with a maximum relative error of around 0.1%. Moreover, compared to the results of 50% random mask on Δ_1 in Section 4.2.3, we see that by employing a finer grid Δ_2 we can allow a higher



(a) Butterfly loss (b) Calendar loss

Figure 4.12: 8×8 grid, training dataset size 73,223, 50% of the implied volatilities are missing, 90% quantile of losses per noise level. (a) Butterfly loss. (b) Calendar loss.

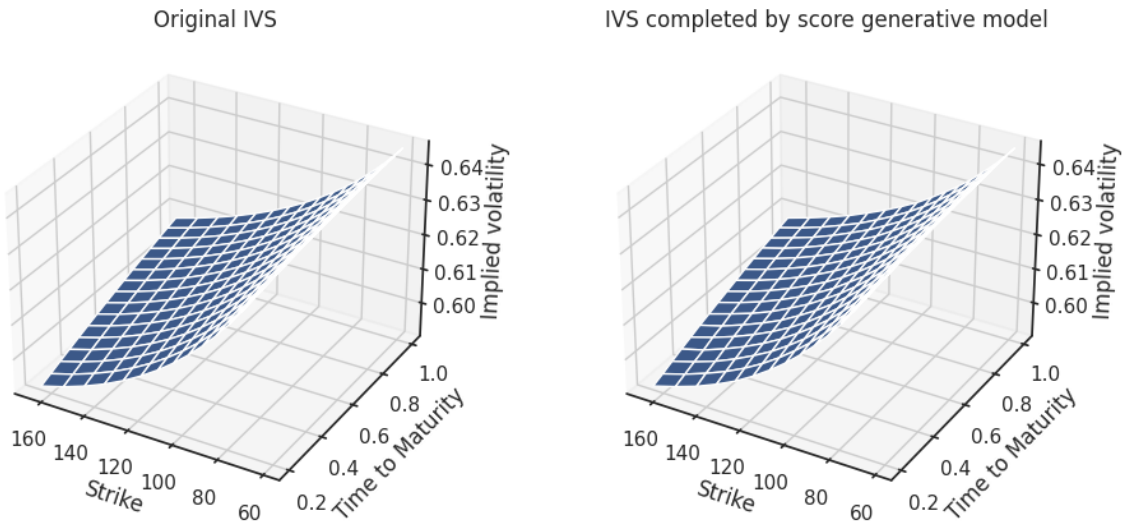
percentage of volatilities to be missing while preserving the same order of errors. Based on these findings, it is recommended to use the finest grid achievable, given the constraints of computational resources and the availability of observable market data.

Once more, there is no discernible visual difference between the original and completed implied volatility surfaces under this randomized mask setting, as illustrated in Figure 4.14.

The average update rate of the no-arbitrage rule, along with the 90% quantiles of the butterfly and calendar losses are shown in Figure 4.15 - 4.16. Similar to the scenario presented in Section 4.2.3, we notice that the average update rate saturates to 1 at a slower pace compared to that in Figure 4.7. Additionally, the butterfly and calendar losses exhibit higher initial values, approximately two orders of magnitude greater than those in Figure 4.8a and 4.8b. Despite this, as shown in Figure 4.16a and 4.16b, both butterfly and calendar losses consistently decrease to zero when the noise level is annealed, ensuring no-arbitrage in the resulting surface.

We remark that when applying a higher percentage of masking, such as 90%, to Δ_2, the maximum relative errors surpass the 1% mark, resulting in an implied volatility surface that does not appear to be visually accurate.

Figure 4.13: 16×16 grid, training dataset size 70,021, errors over 100 test IVS, 80% of the implied volatilities are missing



(a) Original IVS

(b) Completed IVS

Figure 4.14: 16×16 grid, training dataset size 70,021, 80% of the implied volatilities are missing. (a) Original IVS. (b) Completed IVS.

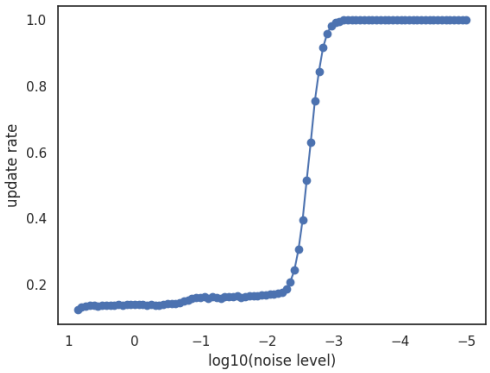
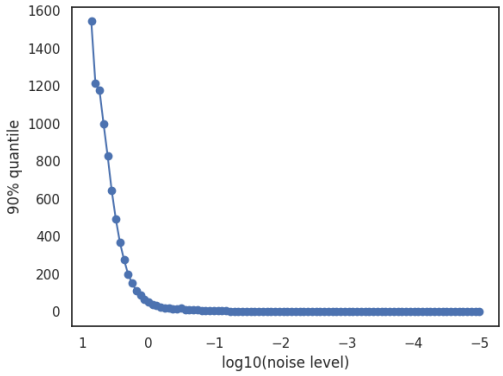
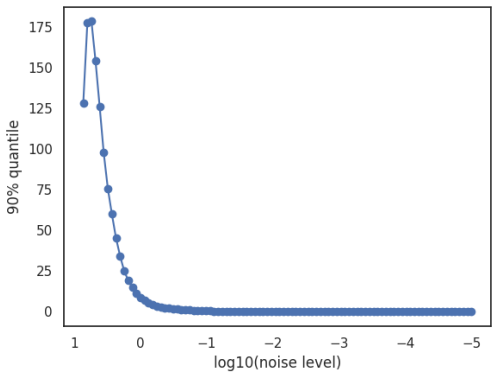


Figure 4.15: 16×16 grid, training dataset size 70,021, 80% of the implied volatilities are missing, average update rate per noise level



(a) Butterfly loss



(b) Calendar loss

Figure 4.16: 16×16 grid, training dataset size 70,021, 80% of the implied volatilities are missing, 90% quantile of losses per noise level. (a) Butterfly loss. (b) Calendar loss.

Chapter 5

Conclusion

In this paper, we have explored the perspective of viewing each implied volatility surface as an image and proposed a score-based generative model approach to solving the volatility surface completion problem. Treating each implied volatility surface as an image, we have developed a no-arbitrage inpainting algorithm for a score-based generative model to output an implied volatility surface that is free of arbitrage. Our numerical results in Chapter 4 demonstrate the effectiveness of our approach, achieving an average relative percent error of order 10^{-4} in the interpolation setting and a maximum relative error less than 0.5% in an extreme setting where 80% of volatilities are assumed to be missing while ensuring no-arbitrage on the produced implied volatility surfaces.

Furthermore, the proposed score-based generative model approach has several advantages over existing methods. First, it is model-free, meaning that it does not require modeling the underlying asset price by stochastic differential equations, bypassing the need for calibration. In fact, we can train our score network purely based on historical data. That is, we can create the dataset $\mathscr{D}_{\text{IVS}} = \{x_i\}_{i=1}^N$ with each x_i being a historical implied volatility surface. Second, by choosing x_i to be surfaces produced by different option pricing models, our score network framework allows an easy mixture of option pricing models. Moreover, the framework is flexible in the sense that we do not assume any patterns in the missing volatilities, *i.e.*, the set A that records the positions of missing implied volatilities, can be an arbitrary subset of Δ, unlike existing methods like [11] where the implied volatility curve for at-the-money options is required. Lastly, once the time is invested in training the score network, the volatility surface completion task can be done quickly via the no-arbitrage inpainting Algorithm 3.

Nonetheless, our score-based generative model is not perfect. An obvious limitation

is that it requires an implied volatility surface to be represented as an image. In terms of training data, this might mean that more time needs to be invested in running the existing option pricing models to generate implied volatility surface for a particular grid Δ. Another difficulty is that one score network can only deal with one grid Δ. While this difficulty can be addressed by making the grid Δ sufficiently fine, this would at the same time require more training data. Moreover, unlike the ANN approach where we can calculate the options' Greeks exactly, we can only obtain the Greeks based on finite difference methods on the grid Δ. This again requires a sufficiently fine grid Δ to ensure numerical accuracy. Future work is encouraged to address these limitations.

There are also other potential directions for future work. One is to understand quantitatively how the size and granularity of the grid Δ will affect our model performance in solving the volatility completion problem. Another direction to further enhance our model performance might be to consider as well the temporal structure of implied volatility surfaces. A good starting point might be to consider the use of score-based generative models to tackle time-series data [32].

References

[1] Damien Ackerer, Natasa Tagasovska, and Thibault Vatter. Deep Smoothing of the Implied Volatility Surface. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December6-12, 2020, virtual*, 2020. 2, 6, 7, 20, 48, 49

[2] Hansjoerg Albrecher, Philipp Mayer, Wim Schoutens, and Jurgen Tistaert. The Little Heston Trap. *Wilmott*, pages 83–92, 3 2007. 28

[3] Andrey Itkin. Deep learning calibration of option pricing models: some pitfalls and solutions. 3 2019. 2

[4] David S Bates. Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options. *The Review of Financial Studies*, 9(1):69–107, 1996. 1

[5] Maxime Bergeron, Nicholas Fung, John Hull, Zissis Poulos, and Andreas Veneris. Variational Autoencoders: A Hands-Off Approach to Volatility. *The Journal of Financial Data Science*, 4(2):125–138, 2022. 1, 2

[6] Fischer Black and Myron S Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3):637–654, 5 1973. 1, 5

[7] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. 7

[8] Hanqun Cao, Cheng Tan, Zhangyang Gao, Guangyong Chen, Pheng-Ann Heng, and Stan Z. Li. A Survey on Generative Diffusion Model. 9 2022. 7

[9] Bruno Dupire. Pricing with a Smile. *Risk*, pages 18–20, 1994. 1

[10] F. Fang and C. W. Oosterlee. A Novel Pricing Method for European Options Based on Fourier-Cosine Series Expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2009. 28

[11] Jim Gatheral and Antoine Jacquier. Arbitrage-free SVI volatility surfaces. *Quantitative Finance*, 14(1):59–71, 2014. 2, 6, 7, 41

[12] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative Finance*, 18(6):933–949, 6 2018. 1

[13] Steven L Heston. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies*, 6(2):327–343, 4 1993. 1, 27, 28

[14] 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. 7

[15] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance*, 21(1):11–27, 2021. 1, 17

[16] Aapo Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. *J. Mach. Learn. Res.*, 6:695–709, 2005. 10

[17] Peter Jäckel. Let's Be Rational. *Wilmott*, 2015(75):40–53, 1 2015. 28, 29

[18] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Ioannis Mitliagkas, and Remi Tachet des Combes. Adversarial score matching and improved sampling for image generation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 22

[19] Alan Lewis. A Simple Option Formula for General Jump-Diffusion and Other Exponential Levy Processes. *SSRN Electronic Journal*, 4 2002. 28

[20] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D Reid. RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5168–5177. IEEE Computer Society, 2017. 22

[21] S Liu, A Borovykh, L A Grzelak, and C W Oosterlee. A neural network-based framework for financial model calibration. *Journal of Mathematics in Industry*, 9, 2019. 1

[22] Shuaiqiang Liu, Cornelis W Oosterlee, and Sander M Bohte. Pricing Options and Computing Implied Volatilities using Neural Networks. *Risks*, 7(1), 2019. 1, 6

[23] Michael Roper. Arbitrage Free Implied Volatility Surfaces. 2010. 2, 6, 7, 28, 48

[24] Brian Ning, Sebastian Jaimungal, Xiaorong Zhang, and Maxime Bergeron. Arbitrage-Free Implied Volatility Surface Generation with Variational Autoencoders. *CoRR*, abs/2108.04941, 2021. 2

[25] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In Francis R Bach and David M Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015. 7

[26] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 7

[27] Yang Song. *Learning to generate data by estimating gradients of the data distribution*. PhD thesis, Stanford University, 2022. 7, 8, 9, 17, 18, 22

[28] Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32, 7 2019. 7, 9, 10, 11, 12, 13, 18, 22

[29] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, volume 2020-December, 2020. 25, 26

[30] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. In Amir Globerson and Ricardo Silva, editors, *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, volume 115 of *Proceedings of Machine Learning Research*, pages 574–584. AUAI Press, 2019. 10

[31] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 11 2021. 7

[32] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December6-14, 2021, virtual*, pages 24804–24816, 2021. 42

[33] Pascal Vincent. A Connection Between Score Matching and Denoising Autoencoders. *Neural Comput.*, 23(7):1661–1674, 2011. 10

APPENDICES

Appendix A

Miscellaneous

A.1 Calendar spread and butterfly arbitrage

It can be shown [23] that an implied volatility surface is free of arbitrage if and only if it is free of calendar spread arbitrage and each time slice of the surface is free of butterfly arbitrage.

For simplicity, let us denote the price of a call option with strike K and maturity T by $C(K,T)$. Let us also denote the price of the underlying at time t by S_t. A calendar spread is a trading strategy Π that involves the simultaneous purchase of a call option with strike K and maturity T_2 and sell of another call option on the same underlying with the same strike K and a shorter maturity $T_1 < T_2$. At the time T_1, the calendar spread has the value:

$$\Pi_{T_1} = -\max(S_{T_1} - K, 0) + C(K, T_2 - T_1),$$

where the first term is the value (payoff) of the short call at T_1 and the second term is the value of the long call at T_1. By a simple no-arbitrage argument, we know that at the time T_1, $C(K, T_2 - T_1) \geq \max(S_{T_1} - K, 0)$ and hence $\Pi_{T_1} \geq 0$ with the probability of $\Pi_{T_1} > 0$ being nonzero. Thus, by no-arbitrage argument, we must have $\Pi_0 > 0$. Calendar spread arbitrage [1] refers to the case that $\Pi_0 \leq 0$, where one can initialize the portfolio Π without cost and is guaranteed to have a non-negative payoff at time T_1. Generally, the calendar spread strategy is used when one expects an overall increase in implied volatility of the underlying.

On the other hand, butterfly arbitrage is a trading strategy Π that involves the purchase of two calls with the same maturity T and strikes $K_1 < K_2$ respectively, and the sell of

two calls with the same maturity T and a same strike $K_3 = (K_1 + K_2)/2$. So we have $K_1 < K_3 < K_2$. Note that at time T, the value of the portfolio Π is given by:

$$\begin{aligned}\Pi_T &= \max(S_T - K_1, 0) + \max(S_T - K_2, 0) - 2\max(S_T - K_3, 0) \\ &= \begin{cases} 0, & \text{if } S_T \geq K_2 \\ K_2 - S_T > 0, & \text{if } K_3 \leq S_T < K_2 \\ S_T - K_1 > 0, & \text{if } K_1 < S_T < K_3 \\ 0, & \text{if } S_T \leq K_1. \end{cases}\end{aligned}$$

Thus, $\Pi_T \geq 0$ with a nonzero probability that $\Pi_T > 0$. By no-arbitrage argument, we must have $\Pi_0 > 0$. Butterfly arbitrage [1] refers to the case that $\Pi_0 \leq 0$. Generally, the butterfly strategy is used when one expects the future volatility of the underlying S to be smaller than its implied volatility.