mathematics

MDPI

Article

Modeling Implied Volatility Surface Using B-Splines with Time-Dependent Coefficients Predicted by Tree-Based Machine Learning Methods

Zihao Chen, Yuyang Li and Cindy Long Yu *

Department of Statistics, Iowa State University , Ames Iowa, IA 50011, USA; zc78@iastate.edu (Z.C.); yuyangli@iastate.edu (Y.L.)
* Correspondence: cindyyu@iastate.edu; Tel.: +1-(515)294-6885

Abstract: Implied volatility is known to have a string structure (smile curve) for a given time to maturity and can be captured by the B-spline. The parameters characterizing the curves can change over time, which complicates the modeling of the implied volatility surface. Although machine learning models could improve the in-sample fitting, they ignore the structure in common over time and might have poor prediction power. In response to these challenges, we propose a two-step procedure to model the dynamic implied volatility surface (IVS). In the first step, we construct the bivariate tensor-product B-spline (BTPB) basis to approximate cross-sectional structures, under which the surface can be represented by a vector of coefficients. In the second step, we allow for the time-dependent coefficients and model the dynamic coefficients with the tree-based method to provide more flexibility. We show that our approach has better performance than the traditional linear models (parametric models) and the tree-based machine learning methods (nonparametric models). The simulation study confirms that the tensor-product B-spline is able to capture the classical parametric model for IVS given different sample sizes and signal-to-noise ratios. The empirical study shows that our two-step approach outperforms the traditional parametric benchmark, nonparametric benchmark, and parametric benchmark with time-varying coefficients in predicting IVS for the S&P 500 index options in the US market.

Keywords: implied volatility surface; forecasting; tensor-product B-spline; tree-based machine learning methods

MSC: 62P05

Citation: Chen, Z.; Li, Y.; Yu, C.L. Modeling Implied Volatility Surface Using B-Splines with Time-Dependent Coefficients Predicted by Tree-Based Machine Learning Methods. *Mathematics* **2024**, *12*, 1100. https://doi.org/10.3390/math12071100

Academic Editors: Yuhlong Lio and Tzong-Ru Tsai

Received: 27 February 2024
Revised: 29 March 2024
Accepted: 1 April 2024
Published: 6 April 2024

1. Introduction

Implied volatility is extracted from the Black–Scholes formula with the observed option price in the market. It represents the people's belief about the market in the future. Therefore, implied volatility can be used in pricing options and looking for the arbitrage opportunity. It is assumed that volatility is a constant in the Black–Scholes formula. However, implied volatility actually can vary across different strikes and time to maturity, known as the implied volatility surface (IVS). The curvature, observed when plotting the strike price against the implied volatility of a group of options at a fixed time to maturity, is referred to as volatility smile, and the dependence on the time to maturity is called volatility term structure. IVS is a collection of extracted volatilities by inverting the Black–Scholes formula given strike prices and time to maturities.

Researchers have been developing approaches to predict IVS. The most traditional way is to consider parametric linear regression models in terms of moneyness and time to maturity. Dumas et al. [1] compare several different linear functions for implied volatility with respect to the strike price and expiration date. The authors argue that the quadratic forms lead to robust empirical results because the Black–Scholes–implied volatilities for

S&P 500 options tend to have a parabolic shape. They also notice the considerable variation in the coefficient estimates from week to week. Similarly, Goncalves and Guidolin [2] consider a regression model that involves a quadratic term of moneyness and interaction term between moneyness and time to maturity to model the time series of the S&P 500 index options' implied volatility surface. A two-stage approach is proposed to model the surface by a parametric linear model and model the dynamics of the coefficients by autoregression models. Heston and Nandi [3] developed a GARCH model to capture the path dependence in volatility, but the approach is less capable of fitting the volatility smile.

On the other hand, endeavors have been made to model the implied volatility surface directly using machine learning methods. Aït-Sahalia and Lo [4] consider the kernel estimation method for state price density that can be related to the implied volatility function. Fengler et al. [5] analyze IVS using common principal component analysis (CPC) and show that the dynamics of the surface can be traced back to a common eigenstructure. Audrino and Colangelo [6] consider the regression trees to improve the prediction of implied volatility from S&P 500. The authors implement a cross-validation strategy for hyperparameter tuning. Das and Padhy [7] investigate the use of support vector regression combined with the Black–Scholes option pricing model in option pricing. Hahn [8] adopts the artificial neural network to model implied volatility in the Australian equity options market before and after the global financial crisis.

Although the aforementioned nonparametric machine learning models provide more flexibility and, therefore, more accurate fitting compared with parametric models, they are not able to take the volatility smile and volatility term structure into consideration. To maximize the prediction ability while keeping the shape constraints and dynamic structures of IVS, we propose the penalized bivariate tensor product B-splines (BTPBs) with time-varying coefficients. Specifically, we approximate IVS (i.e., implied volatility as a function of strike and time to maturity) with a penalized bivariate tensor product B-spline basis at each time point. Upon that, a tree-based machine learning method is implemented to model the time dependence of the coefficients. The first step is also known as the bivariate spline smoothing. The use of basis functions guarantees the consistency of the smile shape and simplifies the prediction of future surfaces. Another attempt to pursue both fitting accuracy and closed functional form is the application of symbolic regression (Luo and Yu [9]). Symbolic regression is a machine learning approach that aims to identify the relationship between input variables and output variables in a given dataset without a predefined functional form. Although symbolic regression has the advantage of exploring actual relationships between implied volatility and other variables, the fitted function could be complicated and change over time. The proposed method differs from such techniques by providing a simple and fixed basis in the functional space, which enables dynamic forecasting.

In finance studies, the smoothing technique has been broadly applied, though limited to univariate cases. Shimko [10] first proposes to interpolate the implied volatility smile curves after converting the option prices to implied volatilities through a cubic spline. Bliss and Panigirtzoglou [11] propose using a smoothing spline that achieves better fit by imposing a penalty function on the wiggle and does not require the pass of all original points exactly. Figlewski [12] modifies the smoothing techniques to take account of the market's bid–ask spread to extract the risk-neutral density from the S&P 500 index options. Existing applications of smoothing techniques primarily focus on univariate splines, fitting dynamic implied volatility surfaces (IVSs) by iterating over term structures and time points. Orosi [13] proposes a nonparametric spline-based method for modeling IVS while ignoring the dynamics of IVS. Our approach, however, utilizes bivariate B-splines to model IVs, allowing coefficients to evolve over time. This method ensures the smooth structure of the implied volatility surface, including the smile feature, while allowing flexibility with dynamic coefficients. Additionally, we employ machine learning techniques to forecast future dynamic coefficients, enabling the prediction of IVs.

In this paper, we extend this exercise in the financial area to a complicated bivariate scenario. For bivariate spline smoothing, Eilers and Marx [14] propose the P-spline and its extension to two-dimensional smoothing. Eilers and Marx [15] and Marx and Eilers [16] consider the bivariate P-splines in signal regression. Wood [17] proposes the thin plate spline smoothing for large data sets, and Wood [18] produces a framework for constructing a "scale-invariant" tensor product using low-rank approximation. Xiao et al. [19] propose a sandwich smoother that accelerates the computation of a tensor product structure. Price et al. [20] implement the penalized bivariate tensor product B-splines (BTPBs) to calculate a conditional yield density and provide the large sample theories of the estimator.

We conduct a simulation study to examine the ability of the penalized BTPBs to fit the implied volatility surface. We consider the underlying cross-sectional model for the log implied volatility following Goncalves and Guidolin [2]. We calculate the root mean square error under the combination of different sample sizes and coefficient of variance. The simulation result suggests that BTPB is able to capture the curvature in the IVS surface under all scenarios with a relative squared error ranging from 0.83% to 5.93%.

In our empirical study, we adopt the two-step procedure to predict the dynamic IVS of S&P 500 options between January 2015 and April 2022. In the first step, we use the bivariate B-spline to estimate coefficients of the B-spline bases for each day. In the second step, we employ the gradient boosted trees method using the times series of estimated coefficients and forecast the coefficients for the future days. Then the predicted IVs can be obtained by applying those predicted coefficients on the B-spline bases. We consider a rolling estimation scheme with a 60-day training period, a 20-day validation period, and a 1-day out-of-sample prediction period. We begin to make daily forecasts in March 2015 and shift the estimation sample by 1 day at a time until the testing period reaches April 2022. We utilize the Bayesian optimization with a Gaussian process in hyperparameter tuning. We also consider three benchmark models: the parametric benchmark (i.e., the classical log implied volatility model proposed in Goncalves and Guidolin [2]), a dynamic modification of the parametric benchmark, and the nonparametric machine learning benchmark. Test statistics (root mean squared error and relative root mean squared error) show that the dynamic BTPB method provides a regular and accurate forecasting over the benchmarks.

We contribute to the literature by applying the penalized bivariate tensor product B-spline to the IVS forecast and combining the cross-sectional fitting with the time series forecasting in a semiparametric manner. The proposed method allows for not only the flexibility and the constraints of the shape of IVS but also the dynamics of IVS over time by modeling the time-dependent coefficients. The simulation study shows that BTPB is able to recover the classical parametric benchmark model under different cases and the increase in the sample size improves the prediction of IVS. In the empirical study, we show that our dynamic semiparametric model whose coefficients are modeled by gradient boosted trees has the best performance in predicting IVS for the S&P 500 index options, followed by the nonparametric benchmark, parametric benchmark, and dynamic parametric benchmark model.

The rest of the paper is organized as follows: Section 2 details the proposed method. Section 2.1 introduces the Black–Scholes formula and implied volatility surface and describes the B-splines and penalized BTPB of moneyness and time to maturity. Section 2.2 describes our general framework for modeling the implied volatility surface with dynamic BTPB. The time-varying spline coefficients are modeled through tree-based machine learning methods with a Gaussian process for tuning hyperparameters. Section 3 provides a simulation study on the performance of our method in modeling the classical implied volatility surface. Section 4 shows an empirical study that demonstrates the better performance of our approach compared with three benchmark models.

2. Methodology

In this section, we propose the two-step modeling of the dynamic implied volatility surface. In Section 2.1, we first introduce the implied volatility as a function of two

key factors: moneyness and time to maturity. Based on this idea, we approximate this surface with the bivariate tensor product B-spline basis (BTPB). We also briefly review the formulation and mathematical properties of the BTPB basis. When the basis is fixed, the dynamic change of the surface can be depicted and predicted by the time-varying coefficients. In Section 2.2, we model these coefficients with gradient boosted regression trees. Section 2.3 summarizes the two-step procedure proposed in this paper. Section 2.4 provides the details of hyperparameter tuning with Bayesian optimization. Section 2.5 lists several benchmark models to demonstrate the advantage of combining semiparametric modeling and machine learning methods.

2.1. Approximate Implied Volatility Surface with BTPB

The famous Black–Scholes formula (Black and Scholes [21]) prices the European call and puts options and connects the price of an option contract to several characters: the current underlying asset price S_t, the strike price K, the time to maturity $\tau = T - t$ (i.e., the expiration time T minus the current time t), the risk-free rate r, and the volatility σ. Specifically, the Black–Scholes formula to price a call option $C_{i,t}$ or a put option $P_{i,t}$ at time t is as follows :

$$\begin{aligned} C_{i,t}^{BS} &= N(d_m)S_t - N(d_\tau)K_{i,t}e^{-r(T_{i,t}-t)} \\ P_{i,t}^{BS} &= N(-d_\tau)K_{i,t}e^{-r(T_{i,t}-t)} - N(-d_m)S_t, \end{aligned} \tag{1}$$

where $d_m = \frac{\ln(S_t/K_{i,t})+(r+\sigma^2/2)\tau_{i,t}}{\sigma\sqrt{\tau_{i,t}}}$, $d_\tau = d_m - \sigma_t\sqrt{\tau_{i,t}}$, and $N(\cdot)$ is the cumulative distribution function (cdf) of a standard normal distribution.

When other variants are fixed, Equation (1) suggests a one-to-one relationship between the option price $C_{i,t}^{BS}$ (or $P_{i,t}^{BS}$) and the volatility parameter σ^2. In practice, the volatility parameter is unknown, while the option prices are observable from the market. Then for every observed price $C_{i,t}$, the contract- and time-specific implied volatility $\sigma_{i,t}^{IV}$ could be numerically solved from the inverse of Equation (1):

$$C_{i,t}^{BS}\left(S_t, K_{i,t}, \tau_{i,t}, r, \sigma_{i,t}^{IV}\right) - C_{i,t} = 0. \tag{2}$$

In the world of Black–Scholes, σ should be a constant number, which is violated by the empirical data. Therefore, researchers allow σ to change while keeping the form of Equation (2). In this sense, we are modeling the data pair $(C_{i,t}, \sigma_{i,t}^{IV})$ as a function of other variants in the equation.

Similarly, given the observed put option market price $P_{i,t}$, we can obtain $\sigma_{i,t}^{IV}$ from $P_{i,t}^{BS}\left(S_t, K_{i,t}, \tau_{i,t}, r, \sigma_{i,t}^{IV}\right) - P_{i,t} = 0$. Thus, the implied volatility is a representation of the option market and reflects people's beliefs about the future. The value of implied volatility depends on the strike price $K_{i,t}$, time to maturity $\tau_{i,t}$, and certain time-specific constants (i.e., risk-free return and current asset price).

Researchers have been developing different models to extract curvatures and term structures of the implied volatility surface. In this paper, we implement the penalized bivariate tensor product B-splines (BTPBs) to approximate the implied volatility surface. We use $m_{i,t} = \frac{K_{i,t}}{S_t}$, known as the moneyness, to simplify the modeling and fit $y_{i,t} = \log(\sigma_{i,t}^{IV})$ as a function of $m_{i,t}$ and $\tau_{i,t}$. We follow the common practice in the literature to take log transformation on implied volatility to reduce the impact of skewness and outliers (Fengler et al. [22]). The rest of this subsection reviews B-spline basis functions and details the construction of penalized BTPBs.

2.1.1. B-Spline Basis Functions

Splines are referred to as a class of functions that are able to produce flexible data smoothing. Instead of fitting a single high-degree polynomial at once, the spline method divides the support of data into intervals with knots and fits each subset of values with

a low-degree polynomial called spline. Each successive spline must have equal values, derivatives, and second derivatives at their joining knots to produce smooth interpolation.

A univariate function $f(X)$ in the B-spline basis space could be written as follows:

$$f(X) = \sum_{k=1}^{K+d+1} \gamma_k B_k(X). \tag{3}$$

$B_k^d(x)$ denotes the kth basis function of degree d, and the B-spline basis is defined recursively as follows:

$$\begin{aligned} B_k^0(x) &= \begin{cases} 1, & \xi_k \leq x < \xi_{k+1}, \\ 0, & \text{otherwise} \end{cases} \\ B_k^d(x) &= \frac{x - \xi_k}{\xi_{k+d} - \xi_k} B_k^{d-1}(x) - \frac{\xi_{k+d+1} - x}{\xi_{k+d+1} - \xi_{k+1}} B_{k+1}^{d-1}(x), \end{aligned} \tag{4}$$

$k = 1, \ldots, K + d + 1$. Then $f(X)$ is essentially a piecewise polynomial of degree d and has continuous derivatives up to degree $d - 2$. With K knots, there are $K + 1$ polynomials of degree d along with dK constraints, leading to $(d + 1)(K + 1) - dK = K + d + 1$ free parameters. It is claimed that cubic splines are the lowest-order spline for which the discontinuity at knots is invisible to the human eye. Therefore, we fix the degree at 3 and ignore the superscript d in the rest of the paper.

2.1.2. Penalized Bivariate Tensor-Product B-Spline (BTPB)

The bivariate tensor product B-splines model provides a straightforward generalization of the one-dimensional B-splines and becomes a flexible semiparametric approach in two-dimensional space. At a fixed time point t, our goal is to approximate $y_{i,t}$, the logarithm of the implied volatility, with a bivariate function $G(m, \tau)$ that falls into the space of B-spline space. That is, the representation of $G(\cdot, \cdot)$ in terms of tensor product B-spline basis is provided by the following:

$$G(m, \tau) = \sum_{\substack{1 \leqslant k_m \leqslant K_m + d_{m+1} \\ 1 \leqslant k_\tau \leqslant K_\tau + d_\tau + 1}} \alpha_{k_m, k_\tau} B_{k_m, d_m}^1(m) B_{k_\tau, d_\tau}^2(\tau), \tag{5}$$

where $B_{k_m, d_m}^1(m)$ and $B_{k_\tau, d_\tau}^2(\tau)$ denote B-spline basis functions for moneyness m and time to maturity τ, respectively, and α_{k_m, k_τ} are the coefficients to be estimated. K_m and K_τ are the number of inner knots and d_m and d_τ are the degrees of the B-spline for m and τ. We model the implied volatility $y_{i,t}$ of option i at time t as follows:

$$y_{i,t} = G(m_{i,t}, \tau_{i,t}) + e_{i,t}, \tag{6}$$

where $G(m_{i,t}, \tau_{i,t})$ is defined as in Equation (5) and $e_{i,t}$ is an error term with mean 0. We use $\boldsymbol{\alpha}$ to denote the vector of length $(K_m + d_m + 1)(K_\tau + d_\tau + 1)$ that contains all the coefficients α_{k_m, k_τ}. The function $G(m, \tau)$ is fully determined by the choice of knots and the coefficient vector $\boldsymbol{\alpha}$. Additionally, $\boldsymbol{\alpha}_t$ is a representation of the implied volatility surface at time point t when knots are fixed.

Using the data observed at day t, we estimate $\boldsymbol{\alpha}_t$ by minimizing the loss function below:

$$\hat{\boldsymbol{\alpha}}_t = \arg\min_{\boldsymbol{\alpha}} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \left(y_{i,t} - \sum_{\substack{1 \leqslant k_m \leqslant K_m + d_m + 1 \\ 1 \leqslant k_\tau \leqslant K_\tau + d_\tau + 1}} \alpha_{k_m, k_\tau} B_{k_\tau, d_m}^1(m_{i,t}) B_{k_\tau, d_\tau}^2(\tau_{i,t}) \right)^2 + \boldsymbol{\alpha}^T \mathbf{P} \boldsymbol{\alpha}. \tag{7}$$

The penalty matrix $\mathbf{P}$ in Equation (7) is defined following Xiao et al. [19]:

$$\mathbf{P} = \lambda_1 \mathbf{B}_2^{\mathrm{T}} \mathbf{B}_2 \otimes \mathbf{D}_1^{\mathrm{T}} \mathbf{D}_1 + \lambda_2 \mathbf{D}_2^{\mathrm{T}} \mathbf{D}_2 \otimes \mathbf{B}_1^{\mathrm{T}} \mathbf{B}_1 + \lambda_1 \lambda_2 \mathbf{D}_2^{\mathrm{T}} \mathbf{D}_2 \otimes \mathbf{D}_1^{\mathrm{T}} \mathbf{D}_1, \tag{8}$$

where λ_1 and λ_2 are the smoothing parameters for m and τ, respectively; $\mathbf{B}_1$ and $\mathbf{B}_2$ are defined as $\mathbf{B}_1 = B^1_{k_m,d_m}(m_{i,t})_{1\leqslant i\leqslant N_t, 1\leqslant k_m\leqslant K_m+d_m+1}$, $\mathbf{B}_2 = B^2_{k_\tau,d_\tau}(\tau_{i,t})_{1\leqslant i\leqslant N_t, 1\leqslant k_\tau\leqslant K_\tau+d_\tau+1}$; and the matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ are the difference matrices of difference orders m_1 and m_2 (Price et al. [20]). The difference penalty is used to remove computational difficulty when the penalty term is defined through an integral, and it controls the smoothness to reduce overfitting (Yoshida [23]). Xiao et al. [19] show that the tensor product structure of the sandwich smoother accelerates the computation and smoothing parameter selection.

2.2. Time-Varying Coefficients with Tree-Based Machine Learning Method

It is implied in the previous section that the implied volatility surface is changing over time. In this section, we model the time series of the time-varying coefficient $\boldsymbol{\alpha}_t$ with a tree-based machine learning model and obtain the forecast of a future implied volatility surface by plugging the predicted coefficients into a BTPB basis. We assume that $\boldsymbol{\alpha}_t$ depends on $\mathbf{x}_t$, a vector of known predicting features at time t. In our practice, we consider $\mathbf{x}_t$ to include the historical coefficients up to a lag of 5, i.e., $\boldsymbol{\alpha}_{t-1}$, $\boldsymbol{\alpha}_{t-2}$, $\boldsymbol{\alpha}_{t-3}$, $\boldsymbol{\alpha}_{t-4}$, and $\boldsymbol{\alpha}_{t-5}$.

For the tree-based method, we consider the gradient boosted regression trees (GBRTs). The motivation for boosting is to combine the outputs of many weak learners to produce a powerful committee, where a weak learner is one whose error rate is only slightly better than random guessing. Boosting trees is a boosting procedure using the simple decision trees as the weak learners. In this paper, we use $T(\mathbf{x};\Theta) = \sum_{j=1}^{J} \gamma_j I(\mathbf{x} \in R_J)$ to represent a decision tree. That is, a decision tree splits the space into J regions and assigns the value γ_j to all data points in area j. The number of J usually depends on the max depth of the decision tree, which effectively changes the model complexity and is one of the hyperparameters to tune. Θ is the tree parameters consisting of the split variables and split points. In a boosting tree, the combination expands the target sequentially as $\hat{\boldsymbol{\alpha}}_t(\mathbf{x}_t) = \hat{\boldsymbol{\alpha}}_t^{(M)}(\mathbf{x}_t) = \sum_{m=1}^{M} T(\mathbf{x}_t;\Theta_m)$, where M is the number of week learners, and Θ_m determines the shape of each tree. These models are fitted by minimizing a loss function $L(\cdot,\cdot)$ averaged over training data:

$$\boldsymbol{L} = \sum_{t=1}^{T} L(\boldsymbol{\alpha}_t, \sum_{m=1}^{M} T(\mathbf{x}_t;\Theta_m)).$$

As a member of the boosted tree family, gradient boosting regression trees apply loss function $L(\cdot,\cdot)$ that is suitable for a regression problem and adding new learners in a gradient decent way (Hastie et al. [24]). Algorithm 1 provides the details for gradient boosted regression trees under squared error loss function $L(\boldsymbol{\alpha}_t, \hat{\boldsymbol{\alpha}}_t) = (\boldsymbol{\alpha}_t - \hat{\boldsymbol{\alpha}}_t)^2$. In each step m, the residual $\boldsymbol{\varepsilon}_t^{(m)} = -\frac{\partial}{\partial \hat{\boldsymbol{\alpha}}} L(\boldsymbol{\alpha}_t, \hat{\boldsymbol{\alpha}})$ is calculated. The algorithm fits a new decision tree $T(\mathbf{x}_t;\hat{\Theta}_m)$ to this residual and adds $T(\mathbf{x}_t;\hat{\Theta}_m)$ to the existing trees. In this way, the fitted tree moves towards the negative gradient direction by amount ν, where ν usually represents the learning rate.

The performance of the machine learning method largely depends on the choice of hyperparameters. When the number of hyperparameters exceeds one, the computation burden of hyperparameter tuning increases exponentially. To solve this problem, we identify two hyperparameters (i.e., the max depth of a simple decision tree and the learning rate) and use the Gaussian process Bayesian optimization to tune the hyperparameters.

Algorithm 1 Gradient Boosted Regression Trees (GBRTs)

Start with $\hat{\boldsymbol{\alpha}}_t^{(0)} = \bar{\boldsymbol{\alpha}}_t = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\alpha}_t$
for m from 1 to M **do**
$\boldsymbol{\varepsilon}_t^{(m)} = -\frac{\partial}{\partial\hat{\boldsymbol{\alpha}}}L(\boldsymbol{\alpha}_t, \hat{\boldsymbol{\alpha}})_{|\hat{\boldsymbol{\alpha}}=\hat{\boldsymbol{\alpha}}_t^{(m-1)}} = \boldsymbol{\alpha}_t - \hat{\boldsymbol{\alpha}}_t^{(m-1)}$
$T(\mathbf{x}_t; \hat{\Theta}_m) = \arg\min_{\Theta_m}\sum_{t=1}^{T}\left(\boldsymbol{\varepsilon}_t^{(m)} - T(\mathbf{x}_t; \Theta_m)\right)^2$
Set $\hat{\boldsymbol{\alpha}}_t^{(m)}(\mathbf{x}_t) = \hat{\boldsymbol{\alpha}}_t^{(m-1)}(\mathbf{x}_t) + \nu T(\mathbf{x}_t; \hat{\Theta}_m)$
end for
$\hat{\boldsymbol{\alpha}}_t(\mathbf{x}_t) = \hat{\boldsymbol{\alpha}}_t^{(M)}(\mathbf{x}_t)$

2.3. Two-Step Procedure: BTPB + GBRT

In this section, we propose a two-step procedure to predict IVS and allow for the dynamics of IVS by modeling the time-dependent coefficients on a rolling basis. In the first step, we construct a bivariate B-spline basis and calculate the coefficients $\boldsymbol{\alpha}_t$ using call (or put) option prices observed at time t. In the second step, we use the historical information $\mathbf{x}_t$ to predict the future coefficients and fit the gradient boosted regression trees (GBRTs) on a rolling basis. The two-step procedure could be summarized as follows:

Step 0. For each estimation window, divide the time series into three consecutive subsets: the training period ($\mathcal{I}_{train}$), the validation period ($\mathcal{I}_{val}$), and the test period ($\mathcal{I}_{test}$). Figure 1 provides an illustration of the first three windows.

Step 1. Fit penalized tensor-product B-splines to the call (or put) option data in the training and validation period time (i.e., $t \in \mathcal{I}_{train} \cup \mathcal{I}_{val}$) and obtain $\boldsymbol{\alpha}_t$'s.

Step 2.

(i) Fit a GBRT model on the data pairs $\{\mathbf{x_t}, \boldsymbol{\alpha}_t\}_{t\in\mathcal{I}_{train}}$ in the training period by minimizing the squared error loss. Choose optimal hyperparameters (i.e., the max depth of the decision tree and the learning rate in boosting) based on the model performance in the validation period $\{\mathbf{x_t}, \boldsymbol{\alpha}_t\}_{t\in\mathcal{I}_{val}}$. The predictors in $\mathbf{x}_t$ include $(\boldsymbol{\alpha}_{t-1}, \boldsymbol{\alpha}_{t-2}, \boldsymbol{\alpha}_{t-3}, \boldsymbol{\alpha}_{t-4}, \boldsymbol{\alpha}_{t-5})$.

(ii) Predict $\boldsymbol{\alpha}_t$ for the test period with $\mathbf{x}_t$. The predicted log implied volatility of the options on the test period $t \in \mathcal{I}_{test}$ is

$$\hat{y}_{i,t} = \sum_{\substack{1\leqslant k_m \leqslant K_m + d_m + 1 \\ 1\leqslant k_\tau \leqslant K_\tau + d_\tau + 1}} \hat{\alpha}_{k_m,k_\tau,t} B^1_{k_m,d_m}(m_{i,t}) B^2_{k_\tau,d_\tau}(\tau_{i,t}),$$

where $\hat{\alpha}_{k_m,k_\tau,t}$ is the predicted future coefficient using GBRT, and $i = 1, 2, \ldots, N_t$.

Figure 1. Split data into training period, validation period, and test period on a rolling basis. For each rolling (colored bars), one BTPB is fit on the training period, and the validation period is used for hyperparameter tuning. Black bars indicate the historical period that is not included in the current windows.

2.4. Hyperparameter Tuning via Bayesian Optimization

Bayesian optimization is used for solving black-box optimization problems where the objective function is unknown or difficult to evaluate. Bayesian optimization replaces the objective function by a Gaussian process (GP) so that we can use a sequence of optimizations to approximate the original problem.

The hyperparameter tuning in machine learning can be treated as a Bayesian optimization problem where we want to maximize the metric on the validation set with respect to the hyperparameters. In the context of hyperparameter tuning in machine learning, the response variable is the metric value $f(\boldsymbol{\omega})$ on the validation set, and covariates are $\boldsymbol{\omega}$, the hyperparameters of machine learning methods.

Bayesian optimization with a Gaussian process takes two steps to find $\boldsymbol{\omega}$ that maximizes $f(\boldsymbol{\omega})$. In the first step, construct a surrogate model through a Gaussian process prior; i.e., we impose a Gaussian prior on $f(\boldsymbol{\omega})$ and have a posterior predictive distribution for $f(\boldsymbol{\omega})$ after incorporating the observed samples from $f(\boldsymbol{\omega})$. Because the posterior of a Gaussian process is still a Gaussian distribution, a Gaussian process is one of the popular surrogate priors. In the second step, we pick a new point $\boldsymbol{\omega}^*$ based on the so-called acquisition function (or utility function) under the posterior distribution from the first step. The objective function $f(\boldsymbol{\omega})$ is then evaluated at the new point $\boldsymbol{\omega}^*$ with the potential of achieving a higher objective function value. Incorporate the new sample to the sample data and update the posterior predictive distribution. The second step is repeated until the desired maximum of the objective function is approximated. Appendix A provides more details.

The grid search of hyperparameters is completely uninformed by past evaluations, and hence spends a significant amount of time evaluating "unpromising" hyperparameters. Bayesian optimization with GP, however, attempts to solve the black-box problem by imposing a Gaussian prior on the unknown objective function and proposing more "promising" points that can yield a higher value of the objective function through the acquisition function. Therefore, a Bayesian optimization approach updates the hyperparameter in a more appropriate direction with the knowledge of past observations based on the correlation imposed by a covariance structure, thus locating the optimal hyperparameter more efficiently. More importantly, because the observations (metric values on a validation set in our context) are noisy, i.e., $\mathbf{v} = f(\boldsymbol{\omega}) + \epsilon$, it is difficult to find $\boldsymbol{\omega}$ that maximizes the underlying function $f(\cdot)$ directly using $\mathbf{v}$, which is contaminated with error. Bayesian optimization with GP creates a way of approximating the true underlying $f(\cdot)$ and produces a posterior predictive distribution that is also Gaussian such that the problem becomes tractable.

2.5. Benchmark Models

We consider three types of benchmark models to evaluate the performance of the proposed method. Section 2.5.1 introduces the typical parametric benchmark model in the literature. Section 2.5.2 considers the time-varying coefficients for the parametric benchmark model by following the same two-step procedure used for BTPB. Section 2.5.3 describes the machine learning model (gradient boosted trees) in modeling IVS.

2.5.1. Parametric Benchmark Model

Goncalves and Guidolin [2] argue that a cross-sectional model specified as in Equation (9) below provides the best fit among the alternative specifications, such as models in which IVS is only a function of moneyness and models without interactions.

Consider the underlying cross-sectional model at time t for the log implied volatility as follows:

$$y_{i,t} = \ln \sigma_{i,t} = \beta_{0,t} + \beta_{1,t} m_{i,t} + \beta_{2,t} m_{i,t}^2 + \beta_{3,t} \tau_{i,t} + \beta_{4,t} (m_{i,t} \times \tau_{i,t}) + \varepsilon_{i,t}, \tag{9}$$

where $\varepsilon_{i,t}$ is the standard normal error with zero mean, $t \in I_{test}$, $i = 1, \ldots, N_t$. $\boldsymbol{\beta}_t = (\beta_{0,t}, \beta_{1,t}, \beta_{2,t}, \beta_{3,t})$ are estimated on a rolling basis as in Figure 1. Note that because we do

not have hyperparameters in the model, the estimation of the coefficients is based on the corresponding training and validation set, i.e., $I_{train} \cup I_{vali}$ for each rolling.

2.5.2. Dynamic Parametric Benchmark Model

In Equation (9), we assume that the coefficients are updated on a rolling basis. Another way of forecasting is to follow the same two-step procedure as in BTPB with time-dependent coefficients modeled by tree-based machine learning methods. Instead of α_t, we now focus on the estimated β_t from the training and validation set. The two-step procedure now becomes as follows:

1. Fit a benchmark model (9) to the option data at time $t \in \mathcal{I}_{train} \cup \mathcal{I}_{vali}$, i.e., the training and validation period, and obtain the following:

$$\boldsymbol{\beta}_t = (\beta_{0,t}, \beta_{1,t}, \beta_{2,t}, \beta_{3,t}, \beta_{4,t})$$

2. For $\beta_{j,t} \in \boldsymbol{\beta}_t, j = 0, 1, 2, 3, 4,$
   (a) Fit a tree-based machine learning model on the data pairs in the training period $\left(\beta_{j,t}, \boldsymbol{x}_{\boldsymbol{\beta}_t}\right), t \in \mathcal{I}_{train}$ by minimizing the squared error loss $\sum_{t \in \mathcal{I}_{train}} (\beta_{j,t} - \hat{\beta}_{j,t})^2$, and the hyperparameter tuning is based on data pairs in the validation period $\left(\beta_{j,t}, \boldsymbol{x}_{\boldsymbol{\beta}_t}\right), t \in \mathcal{I}_{val}$. $\boldsymbol{x}_{\boldsymbol{\beta}_t} = (\boldsymbol{\beta}_{t-1}, \boldsymbol{\beta}_{t-2}, \boldsymbol{\beta}_{t-3}, \boldsymbol{\beta}_{t-4}, \boldsymbol{\beta}_{t-5})$. The Gaussian process Bayesian optimization is also implemented to tune the hyperparameters.
   (b) Obtain prediction $\hat{\beta}_{j,t}$ on the test set with $\boldsymbol{x}_{\boldsymbol{\beta}_t}, t \in \mathcal{I}_{test}$.

   The predicted implied volatility of the options on the test period $t \in I_{test}$ are s follows:

$$y_{i,t} = \hat{\beta}_{0,t} + \hat{\beta}_{1,t} m_{i,t} + \hat{\beta}_{2,t} m_{i,t}^2 + \hat{\beta}_{3,t} \tau_{i,t} + \hat{\beta}_{4,t} (m_{i,t} \times \tau_{i,t}) + \varepsilon_{i,t},$$

   where $t = 1, 2, \ldots, N_t$.

2.5.3. Nonparametric Machine Learning Benchmark Models

Following the general framework of a machine learning problem, we consider the model for the log implied volatility of option i at time t, $\ln \sigma_{i,t}$, as follows:

$$y_{i,t} = \ln \sigma_{i,t} = G(\mathbf{x}_{i,t}) + e_{i,t}, \tag{10}$$

where $G(\cdot)$ is an unknown complex function, $e_{i,t}$ is the noise term of option i at time t, and $\mathbf{x}_{i,t} = (m_{i,t}, \tau_{i,t})$ is a vector of predictors of option i at time t with $i = 1, \ldots, N_t$. N_t is the number of options at time t. The Bayesian optimization with a Gaussian process is used in hyperparameter tuning.

3. Simulation

In this section, we evaluate the approximation performance of BTPB based on a theoretical implied volatility surface. We consider the underlying cross-sectional model for the log implied volatility proposed by Goncalves and Guidolin [2] and demonstrate the result under different combinations of sample size and signal-to-noise ratio.

For some specific time t, the implied volatility for the ith option is generated by the following:

$$\ln \sigma_i = \beta_0 + \beta_1 m_i + \beta_2 m_i^2 + \beta_3 \tau_i + \beta_4 (m_i \times \tau_i) + \varepsilon_i, i = 1, \ldots, N, \tag{11}$$

where $m_i \sim N(0.6, 1)$ is truncated at 0 and 1.2, $\tau_i \sim \frac{1}{365} BetaBin(365, 1, 2)$ and $\epsilon_i \sim N(0, \omega^2)$. $(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4)$ are specified in a similar manner as in Goncalves and Guidolin [2] to have appropriate curvature. ω is set to adjust for different signal-to-noise ratios. Define the coefficient of variation (CV) as the standard deviation of ε_i over the empirical mean of $|\beta_0 + \beta_1 m_i + \beta_2 m_i^2 + \beta_3 \tau_i + \beta_4 (m_i \times \tau_i)|$, i.e., $\frac{\omega}{\frac{1}{N} \sum_{i=1}^{N} |\beta_0 + \beta_1 m_i + \beta_2 m_i^2 + \beta_3 \tau_i + \beta_4 (m_i \times \tau_i)|}$. Once N and CV are fixed, ω is determined accordingly.

We consider four different cases with $N = 200$ and 1000, $CV = 0.02$ and 0.1. Then, fit penalized tensor-product B-splines for each case with $K_m = K_\tau = 3$, $d_m = 3$, and $d_\tau = 2$. Lower order is considered for τ because the data are more sparse in the direction of τ. Table 1 shows the relative root mean squared error (RMSE) ($\frac{\text{RMSE}}{\frac{1}{n}\sum|y_i|}$) for each case. There is no significant difference in the relative RMSE between the different sample sizes under the same CV level.

Table 1. Relative RMSE for four cases of two sample sizes and two CV levels. There is no significant difference between the different sample sizes under the same CV level. Therefore, the model performance on the sample is not too sensitive to the sample size.

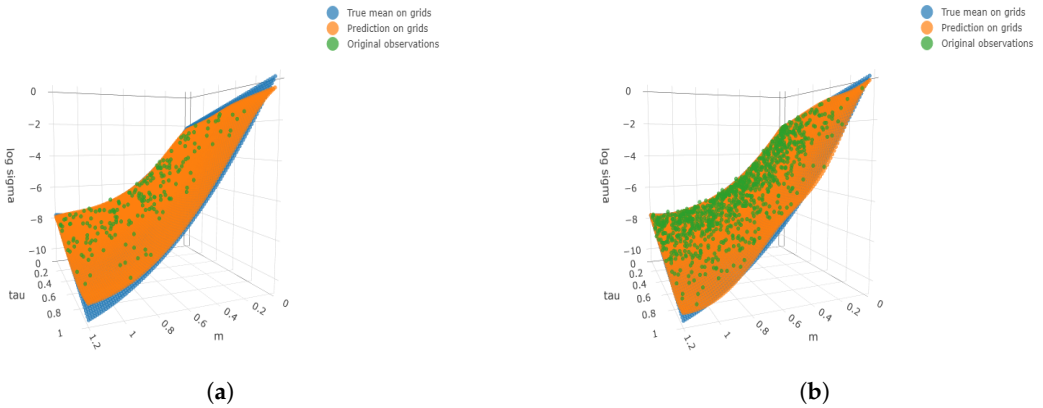
	$CV = 0.02$	$CV = 0.1$
$N = 200$	0.0169	0.0860
$N = 1000$	0.0196	0.0991

To further examine the fit of the surface, we construct a 100 by 50 grid with 100 equal-spaced breaks for $m \in (0, 1.2)$ and 50 equal-spaced breaks for $\tau \in (0, 1)$. For each case, we evaluate the model on the 100 by 50 grid, i.e., 5000 points in total, and compare the prediction to the underlying true means on the grid. Define $\text{RMSE}_g = \sqrt{\frac{\sum_{j=1}^{N_g}(\hat{y}_j - \mu_j)^2}{N_g}}$ and relative $\text{RMSE}_g = \frac{\text{RMSE}_g}{\frac{1}{N_g}\sum_{j=1}^{N_g}|\mu_i|}$ to measure the difference between the predictions and the underlying means of the 5000 points on the grid where $N_g = 5000$. Table 2 shows the relative RMSE_g based on the predictions and underlying means of the grid. Notice that the relative RMSE_g is significantly lower when we have $N = 1000$ compared with $N = 200$ under the same CV level because the more data, the better fit to the underlying true surface.

Table 2. Relative RMSE_g on grid for four cases of two sample sizes and two CV levels. Notice that the relative RMSE_g is significantly lower when we have $N = 1000$ compared with $N = 200$ under the same CV level because the more data, the better fit to the underlying true surface.

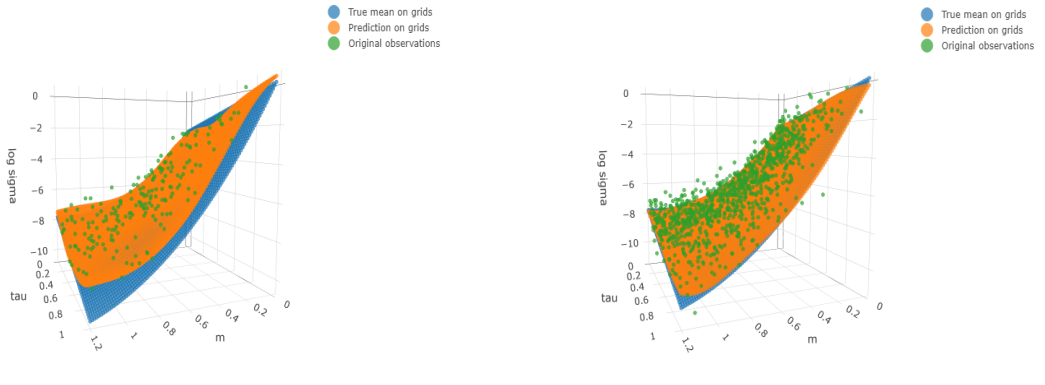
	$CV = 0.02$	$CV = 0.1$
$N = 200$	0.0178	0.0593
$N = 1000$	0.0083	0.0134

Figure 2 shows the 3D surface plots where we plot the predictions on the grid, the true means on the grid, and the simulated observations. Figures 3–6 are the 2D cross-section plots conditional on a specific τ or m. For the 2D cross-section plots conditional on τ, we split the range of τ into eight equal-sized groups. The predictions and true means are plotted across m conditional on the centers of the groups. Simulated observations falling into the corresponding group are scattered green dots. Plots for the first, third, fifth, and eighth group are shown. A similar procedure is applied to have the 2D cross-section plots conditional on m.



(**a**) (**b**)

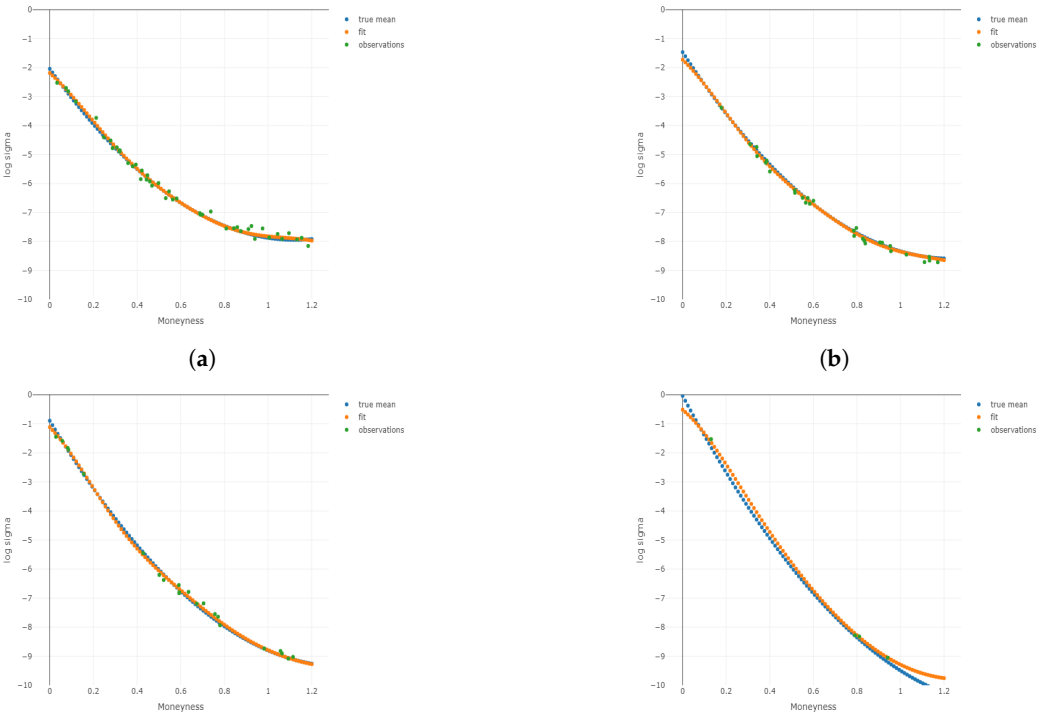
Figure 2. *Cont.*



(**c**) (**d**)

Figure 2. Three-dimensional surface plots under four cases. We consider the finer grids 100 and 50 for $m \in (0, 1.2)$ and $\tau \in (0, 1)$ respectively. The true means, simulated observations, and predicted values are plotted on the finer grids as well as the simulated observations. (**a**) $N = 200, CV = 0.02$; (**b**) $N = 1000, CV = 0.02$; (**c**) $N = 200, CV = 0.1$; (**d**) $N = 1000, CV = 0.1$.

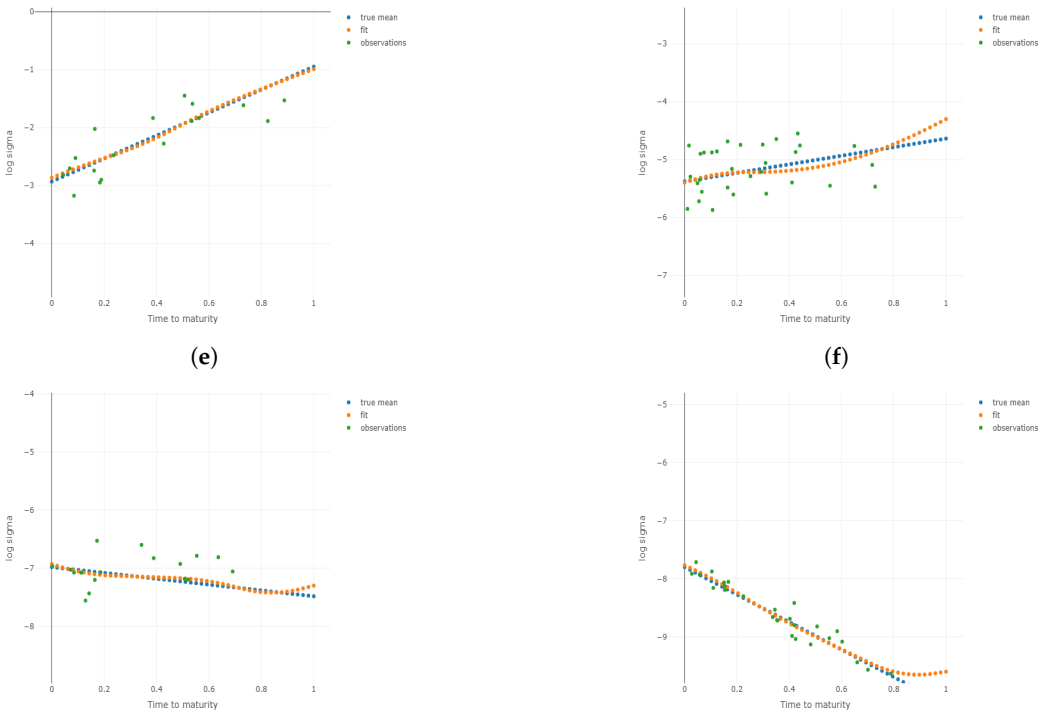
In general, BTPB is able to capture the curvature in the IVS surface in all cases, although in the 2D cross-section plots, we notice that the model is not performing well close to the boundary. When $N = 200, CV = 0.1$, we have the largest (relative) RMSE on the grid due to the small sample size and high CV ratio. Figure 5 shows poor fit in the 2D cross-section plots conditional on τ. However, the increase in the sample size mitigates the issue, and the model recovers the IVS surface better when $N = 1000$ for both CV ratios.



(**a**) (**b**)

(**c**) (**d**)

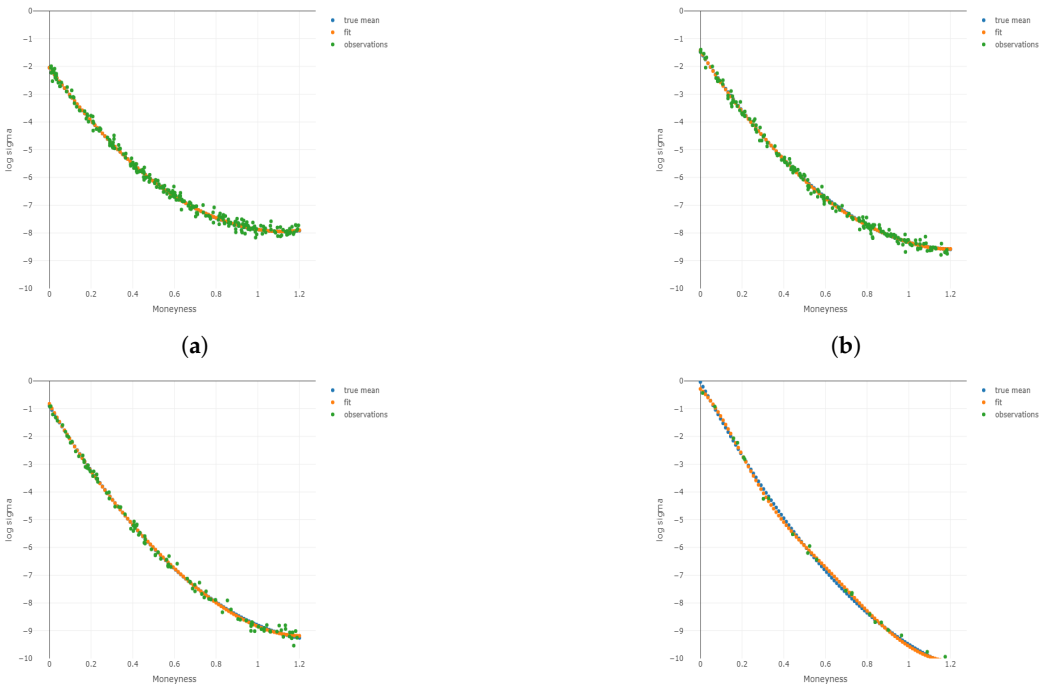
Figure 3. *Cont.*



(**e**) (**f**)

(**g**) (**h**)

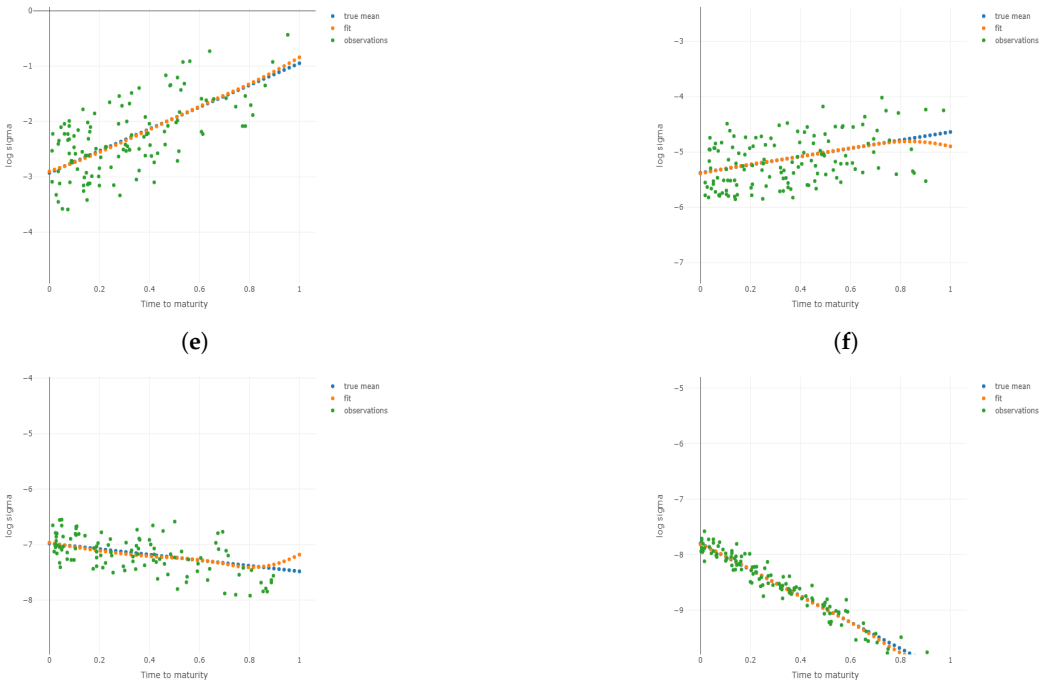
Figure 3. Two-dimensional plots for different conditional on different τ and m under $N = 200$, $CV = 0.02$. Plots for the 1st, 3rd, 5th, and 8th group are shown conditional on τ and m, respectively. (**a**) Two-dimensional plots conditional on the center of the 1st group of τ; (**b**) 2D plots conditional on the center of the 3rd group of τ; (**c**) 2D plots conditional on the center of the 5th group of τ; (**d**) 2D plots conditional on the center of the 8th group of τ; (**e**) 2D plots conditional on the center of the 1st group of m; (**f**) 2D plots conditional on the center of the 3rd group of m; (**g**) 2D plots conditional on the center of the 5th group of m; (**h**) 2D plots conditional on the center of the 8th group of m.



(**a**) (**b**)

(**c**) (**d**)

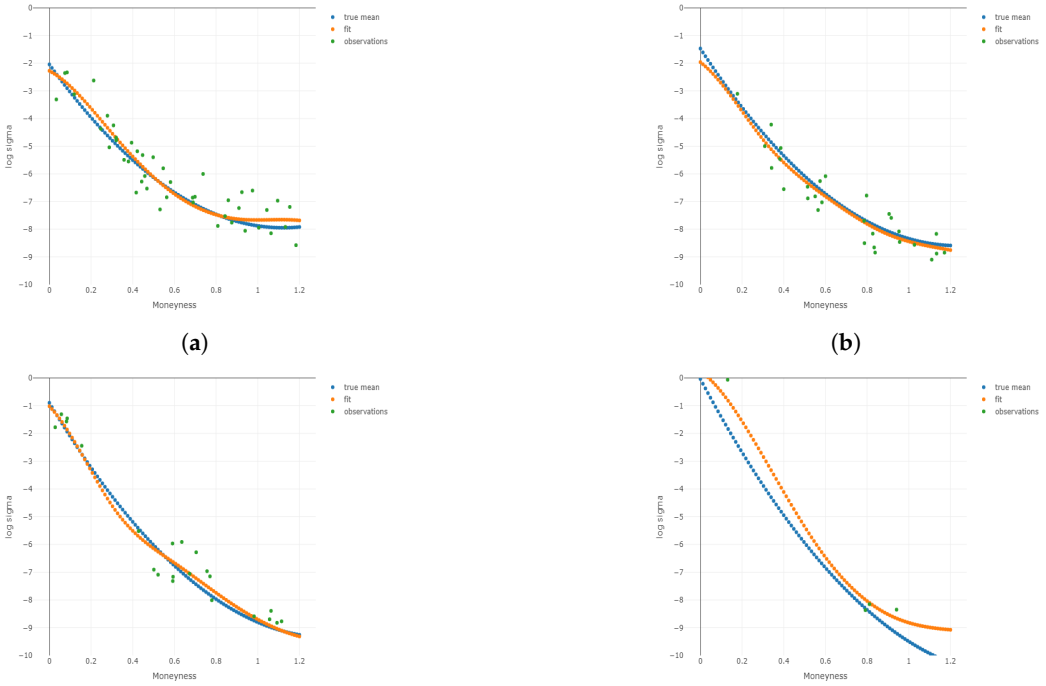
Figure 4. *Cont.*



(**e**) (**f**)

(**g**) (**h**)

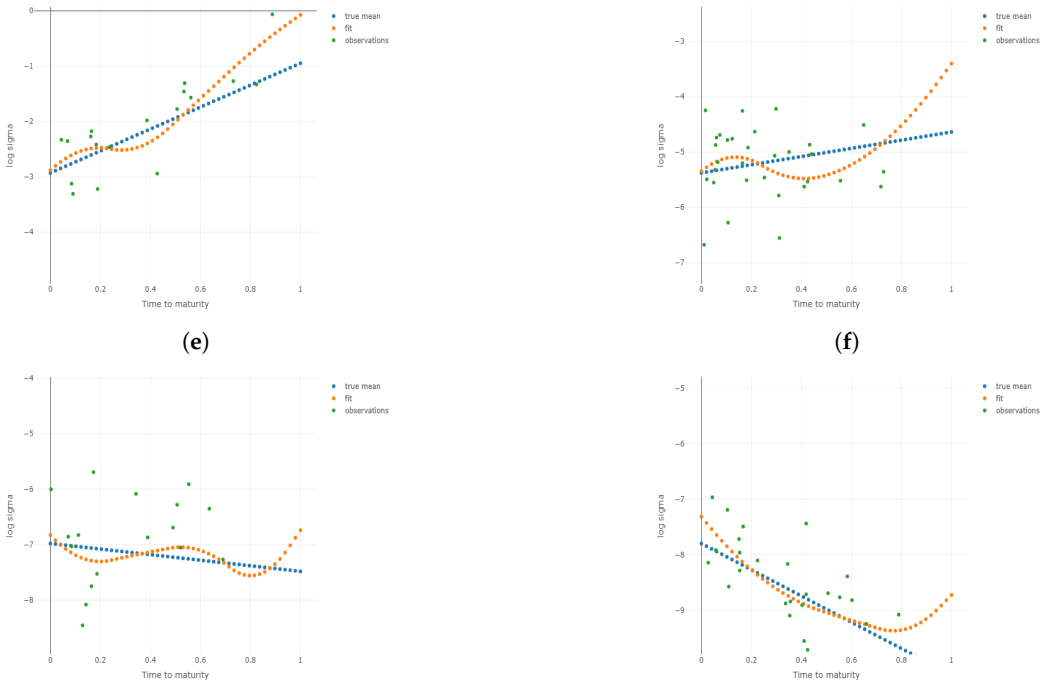
Figure 4. Two-dimensional plots for different conditional on different τ and m under $N = 1000$, $CV = 0.02$. Plots for the 1st, 3rd, 5th, and 8th group are shown conditional on τ and m, respectively. (**a**) Two-dimensional plots conditional on the center of the 1st group of τ; (**b**) 2D plots conditional on the center of the 3rd group of τ; (**c**) 2D plots conditional on the center of the 5th group of τ; (**d**) 2D plots conditional on the center of the 8th group of τ; (**e**) 2D plots conditional on the center of the 1st group of m; (**f**) 2D plots conditional on the center of the 3rd group of m; (**g**) 2D plots conditional on the center of the 5th group of m; (**h**) 2D plots conditional on the center of the 8th group of m.



(**a**) (**b**)

(**c**) (**d**)

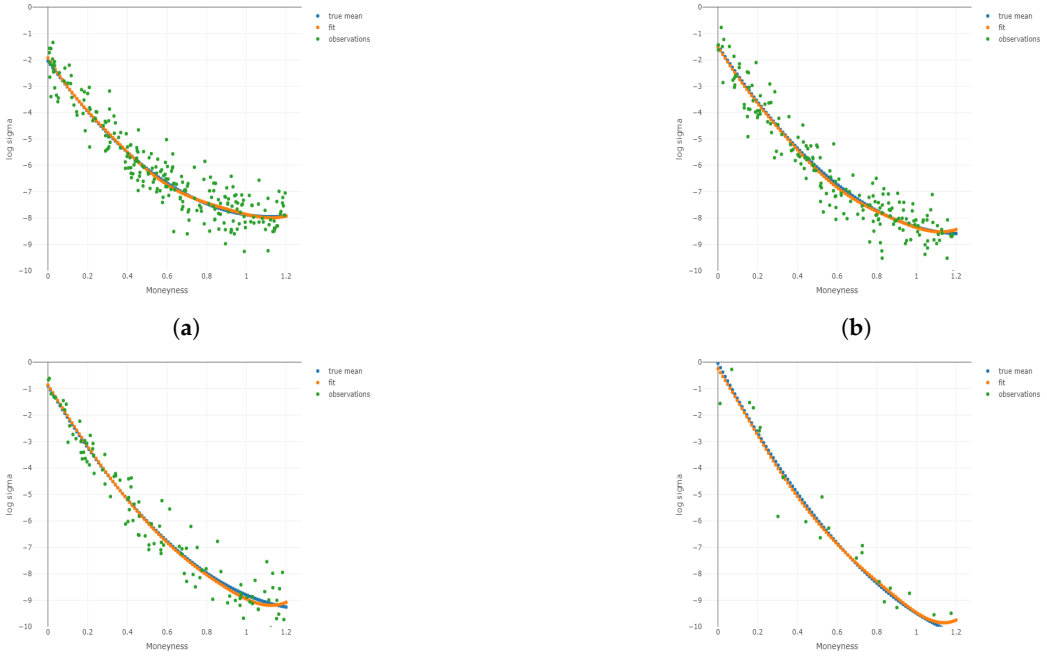
Figure 5. *Cont.*



(**e**) (**f**)

(**g**) (**h**)

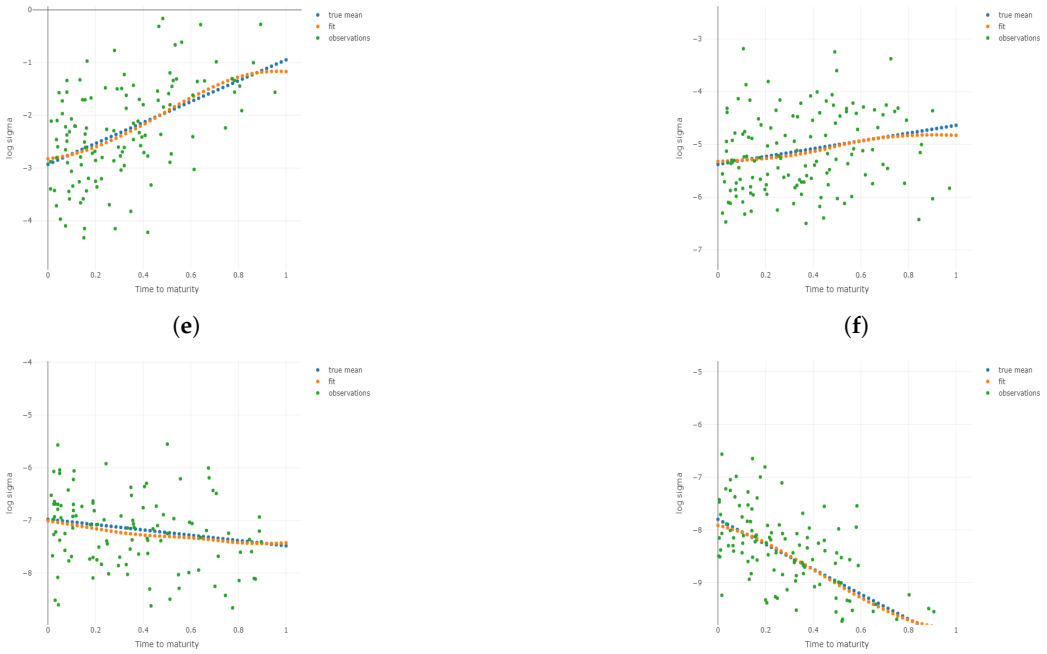
Figure 5. Two-dimensional plots for different conditional on different τ and m under $N = 200$, $CV = 0.1$. Plots for the 1st, 3rd, 5th, and 8th group are shown conditional on τ and m, respectively. (**a**) Two-dimensional plots conditional on the center of the 1st group of τ; (**b**) 2D plots conditional on the center of the 3rd group of τ; (**c**) 2D plots conditional on the center of the 5th group of τ; (**d**) 2D plots conditional on the center of the 8th group of τ; (**e**) 2D plots conditional on the center of the 1st group of m; (**f**) 2D plots conditional on the center of the 3rd group of m; (**g**) 2D plots conditional on the center of the 5th group of m; (**h**) 2D plots conditional on the center of the 8th group of m.



(**a**) (**b**)

(**c**) (**d**)

Figure 6. *Cont.*



(**e**) (**f**)

(**g**) (**h**)

Figure 6. Two-dimensional plots for different conditional on different τ and m under $N = 1000$, $CV = 0.1$. Plots for the 1st, 3rd, 5th, and 8th group are shown conditional on τ and m, respectively. (**a**) Two-dimensional plots conditional on the center of the 1st group of τ; (**b**) 2D plots conditional on the center of the 3rd group of τ; (**c**) 2D plots conditional on the center of the 5th group of τ; (**d**) 2D plots conditional on the center of the 8th group of τ; (**e**) 2D plots conditional on the center of the 1st group of m; (**f**) 2D plots conditional on the center of the 3rd group of m; (**g**) 2D plots conditional on the center of the 5th group of m; (**h**) 2D plots conditional on the center of the 8th group of m.

4. Empirical Study

In this section, we aim to demonstrate the effectiveness of the two steps using in-sample and out-sample performance, respectively. Section 4.1 introduces the option data. Section 4.2 introduces the additive B-spline model, a special case of BTPB, to incorporate the overidentification problem in practice. Section 4.3 provides the in-sample result to show the fitting accuracy of the proposed method. Section 4.4 highlights the forecasting ability based on the dynamic coefficients framework and the gradient boosted regression trees on a rolling scheme.

The construction of BTPB is implemented in `R(4.3.2)`, where the package `mgcv(v1.9)` can construct and fit a B-spline basis under a general additive model. BRT is implemented in `Python(3.9)`. The code for implementing the proposed two-step method can be found in GitHub (accessed on 20 March 2024) (YuyangLi0606/Dynamic-IVS/tree/main).

4.1. Data

We obtain daily option prices for S&P 500 that starts in January 2015 and ends in April 2022 following Gao et al. [25]. These options cover a large range of time to maturities from 1 to 1872 calendar days, as well as a large range of absolute moneyness $(\frac{K_{i,t}}{S_t} - 1)$ from -0.98 to 1.40. The implied volatility from the BS formula uses the interbank borrowing rates as the discount rate. It is a common practice to filter the biased option data according to certain criteria (Orosi [13], Kim [26]). Specifically, we only keep options with volume greater than 5, open interest greater than 5, best bid greater than USD 3.8, implied volatility greater than 0, and time to maturity greater than 6 days but less than 365 days. We also discard options with absolute moneyness greater than 0.1 or less than -0.1, i.e., the options with absolute moneyness in excess of 10%. We predict the IVS for call options and put options separately. Although the implied volatility surface could be different for call options and

put options, this paper focuses on the results based on call options since the put options tell a similar story. Table 3 shows the summary statistics for all call options. Notably, the time to maturity has the largest skewness and kurtosis, indicating the heavy tail distribution.

Table 3. This table shows summary statistics for call options in the real data. Note that the time to maturity has the largest skewness and kurtosis, indicating the heavy tail distribution.

	Mean	Volatility	Skewness	Kurtosis	Min	Max
Implied volatility	−1.9106	0.3827	0.4528	0.5749	−3.2729	0.1913
Moneyness	1.0086	0.0329	−0.0294	0.5207	0.8780	1.0998
Time to maturity	0.1439	0.1620	2.7109	8.4048	0.0164	1.0000

4.2. Additive B-Splines Model

We have introduced the bivariate tensor-product B-splines (BTPBs) and how the time-varying coefficients can be modeled with tree-based machine learning methods. However, because Equation (5) does not necessarily provide a full-rank model matrix, the existence of extreme coefficients due to multicollinearity becomes a problem when modeling the coefficients through the tree-based machine learning method gradient boosted trees using the lagged coefficients.

Therefore, instead of BTPB, we consider the additive B-splines in the empirical study. Define the additive B-spline as follows:

$$\sum_{1\leqslant k_m\leqslant K_m+d_m+1} \alpha_{k_m,1} B^1_{k_m,d_m}(m) + \sum_{1\leqslant k_\tau\leqslant K_\tau+d_\tau+1} \alpha_{k_\tau,2} B^2_{k_\tau,d_\tau}(\tau), \tag{12}$$

and Equation (6) becomes as follows:

$$y_{i,t} = G(m_{i,t},\tau_{i,t}) + e_{i,t} = \alpha_0 + f_1(m_{i,t}) + f_2(\tau_{i,t}) + e_{i,t}, \tag{13}$$

where $f_1(m_{i,t}) = \sum_{1\leqslant k_m\leqslant K_m+d_m+1} \alpha_{k_m,1} B^1_{k_m,d_m}(m_{i,t})$, $f_2(\tau_{i,t}) = \sum_{1\leqslant k_\tau\leqslant K_\tau+d_\tau+1} \alpha_{k_\tau,2} B^2_{k_\tau,d_\tau}(\tau_{i,t})$.

Define $\boldsymbol{\alpha}_{\text{additive}}$ to be a vector of length $(K_m+d_m+1)+(K_\tau+d_\tau+1)+1$ that contains all the coefficients, i.e., $\boldsymbol{\alpha}_{\text{additive}} = (\alpha_0, \alpha_{1,1}, \alpha_{2,1}, \ldots, \alpha_{k_m,1}, \ldots, \alpha_{1,2}, \alpha_{2,2}, \ldots, \alpha_{k_\tau,2})$. Our goal is to the estimator $\boldsymbol{\alpha}_{\text{additive}}$ by minimizing the loss function below:

$$\arg\min_{\boldsymbol{\alpha}} \sum_{t=1}^{T}\sum_{i=1}^{N_t} (y_{i,t} - (\alpha_0 + f_1(m_{i,t}) + f_2(\tau_{i,t})))^2 + \left[\lambda_1 \int_{a_m}^{b_m} (f_1''(m_{i,t}))^2 dm + \lambda_1 \int_{a_\tau}^{b_\tau} (f_2''(\tau_{i,t}))^2 d\tau\right], \tag{14}$$

where $y_{i,t}$ is the implied volatility on a log scale of the ith option at time t; the smoothing parameter λ_1, λ_2 selection is implemented in R package *mgcv*. Note that we consider the derivative-based penalty in Equation (14) because it consists of essentially two univariate smoothing B-splines and does not have heavy computation cost in integral.

To allow for time-varying coefficients, the two-step procedure is now applied to $\boldsymbol{\alpha}_{\text{additive}}$ accordingly. In the following sections, we consider $K_m = K_\tau = 3$, $d_m = d_\tau = 3$ for additive B-splines.

4.3. In-Sample Model Performance

In this subsection, we look at the in-sample performance as it is also of research interest for IVS calibration (Audrino and Colangelo [6]). We compare the additive B-splines model with the traditional parametric benchmark model. Specifically, each of these two models is used to fit the IVS for each trade date, and we aggregate the results across all trade dates.

Table 4 provides the RMSE and the relative RMSE ($\frac{\text{RMSE}}{\frac{1}{n}\sum|y_i|}$) as the measure of goodness of fit for two models. The additive B-splines model has significantly lower (relative) RMSE and hence provides a better fit of the IVS than the parametric benchmark model.

A call option is in-the-money (ITM) when $m > 1$ and out-of-the-money (OTM) when $m < 1$, and there is no at-the-money ($m = 1$) option in our data. The option is overvalued when the predicted implied volatility is greater than the observed implied volatility and

undervalued the other way around. Table 5 compares the RMSE in terms of moneyness, and Table 6 compares the RMSE between the overvalued options and undervalued options for both models. The additive B-splines model has consistently performed better than the parametric benchmark.

Table 4. The RMSE and relative RMSE are provided for the additive B-splines model and the parametric benchmark model. The additive B-splines model has significantly lower (relative) RMSE and hence provides a better fit of the IVS than the parametric benchmark model.

	Additive B-Splines	**Parametric Benchmark**
RMSE	0.0532	0.0702
Relative RMSE	0.0279	0.0368

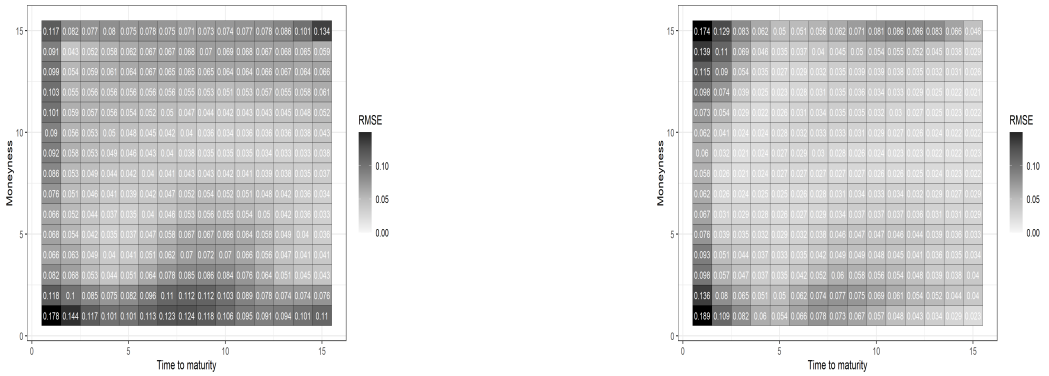
Table 5. Calculate the RMSE for the in-the-money (ITM) options and out-of-money options (OTM) under two models. The additive B-splines model has consistently smaller RMSE.

	Additive B-Splines	**Parametric Benchmark**
ITM ($m < 1$)	0.0424	0.0665
OTM ($m > 1$)	0.0665	0.0756

Table 6. Calculate the RMSE for the overvalued options and undervalued options under two models. The additive B-splines model has consistently smaller RMSE.

	Additive B-Splines	**Parametric Benchmark**
Overvalued	0.0564	0.0742
Undervalued	0.0502	0.0663

To examine the in-sample fit performance among different moneynesses and time to maturities, Figure 7 provides the heatmaps of RMSEs based on a 15 by 15 grid determined by equal-spaced percentiles in moneyness and time to maturity. The parametric benchmark model has a darker area than additive B-splines in the bottom, indicating a poorer fit when the moneyness is close its lower boundary. To further visualize a comparison, we focus on the date 5 February 2021. Figures 8 and 9 show 3D plots from four different angles to compare the performances of the parametric benchmark model and the additive B-splines model. Similarly, Figures 10 and 11 show 2D plots conditional on four different levels of τ that corresponds to the 20th, 35th, 65th, and 80th percentiles. Overall, the additive B-splines model has a better fit than the parametric benchmark model due to its flexibility.



(**a**) (**b**)

Figure 7. Two heatmaps are based on a 15 by 15 grid determined by equal-spaced percentiles in moneyness and time to maturity. The parametric benchmark model has a darker area than the additive B-splines model in the bottom, indicating a poorer fit when the moneyness is close its lower boundary. (**a**) Parametric benchmark; (**b**) additive B-spline.

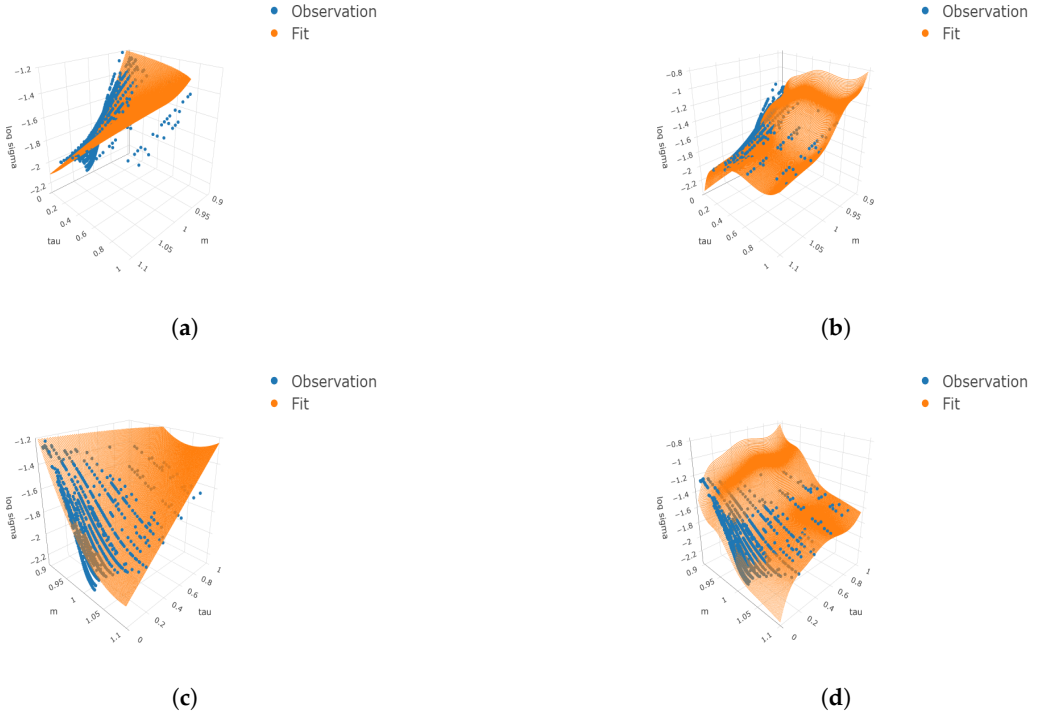


Figure 8. Four 3D plots with angle 1 and angle 2 compare the in-sample fit of IVS from the parametric benchmark and additive B-splines model on the date 5 February 2021. The additive B-splines model has a better fit to the data. (**a**) Angle 1 for parametric benchmark, (**b**) angle 1 for additive B-splines, (**c**) angle 2 for parametric benchmark, and (**d**) Angle 2 for additive B-splines.

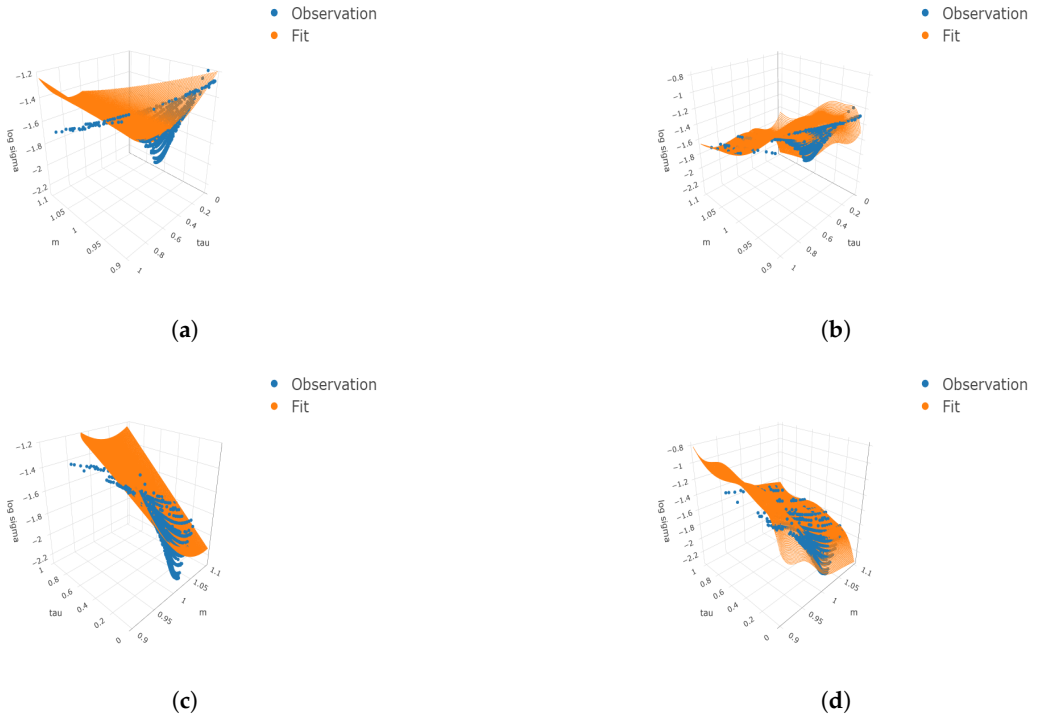


Figure 9. Four 3D plots with angle 3 and angle 4 compare the in-sample fit of IVS from the parametric benchmark and additive B-splines model on the date 5 February 2021. The additive B-splines model has a better fit to the data. (**a**) Angle 3 for parametric benchmark, (**b**) angle 3 for additive B-splines, (**c**) angle 4 for parametric benchmark, and (**d**) angle 4 for additive B-splines.

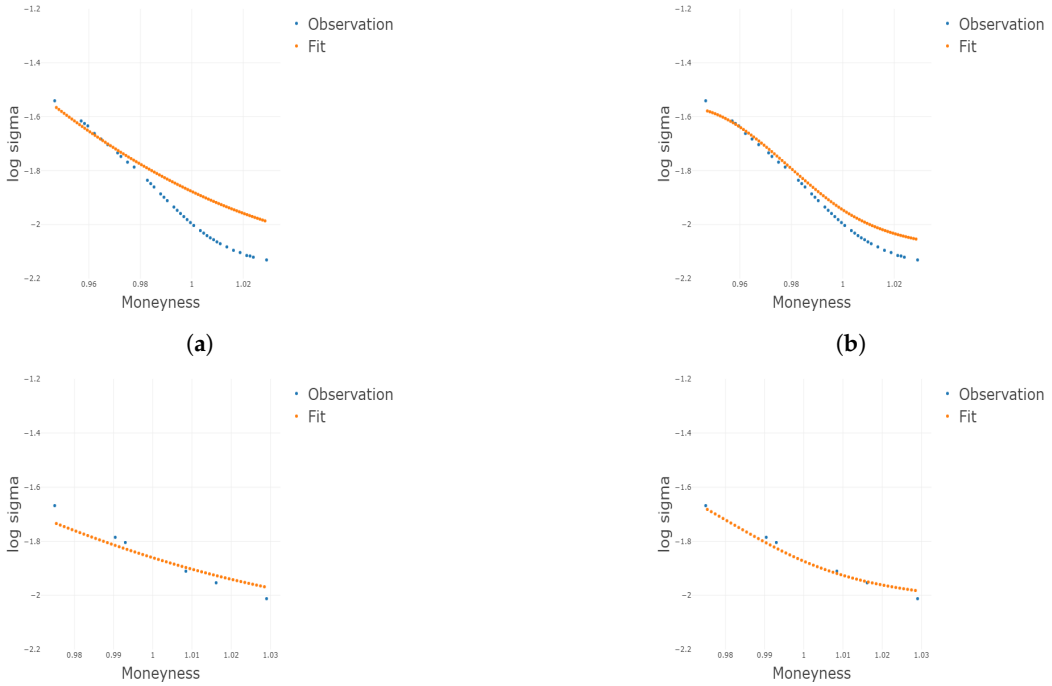


Figure 10. Four 2D plots conditional on the 20th and 35th percentile of τ compare the in-sample fit of IVS from the parametric benchmark and additive B-splines model on the date 5 February 2021. The additive B-splines model has a better fit to the data. (**a**) Parametric benchmark model fit conditional on the 20th percentile, (**b**) additive B-splines model fit conditional on the 20th percentile, (**c**) parametric benchmark model fit conditional on the 35th percentile, and (**d**) additive B-splines model fit conditional on the 35th percentile.

4.4. Out-of-Sample Model Performance

In this subsection, we compare the out-of-sample results for the additive B-splines model with time-varying coefficients (dynamic additive B-splines), parametric benchmark model, nonparametric benchmark model, and dynamic parametric benchmark model based on the rolling scheme.

We start the rolling scheme in March 2015. Figure 1 shows the first three rolling of the two-step procedure. We consider the training period to be 60 days, the validation period 20 days, and the test period 1 day.

Similar to in-sample results, Table 7 provides the RMSE and relative RMSE ($\frac{\text{RMSE}}{\frac{1}{n}\sum|y_i|}$) as the measure of goodness of fit for the four models. The dynamic additive B-splines model has the lowest (relative) RMSE, followed by the nonparametric benchmark, parametric benchmark, and dynamic parametric benchmark model, and hence provides the best fit of the IVS among the four models. Table 8 compares the RMSE in terms of the moneyness, and Table 9 compares the RMSE between the overvalued options and undervalued options. The dynamic additive B-splines model has consistently performed better than the other three benchmarks.

To examine the prediction performance among different moneynesses and time to maturities, Figure 12 provides the heatmaps of RMSEs based on a 15 by 15 grid determined by equal-spaced percentiles in moneyness and time to maturity. The dynamic parametric benchmark has the darkest heatmap, while the dynamic additive B-splines model has the brightest, indicating the best fit among the four models. Figure 13 compares the RMSE time series across the trade date for the four models. The dynamic additive B-splines model has the most stable time series with the narrowest band and fewest extremes, while the dynamic parametric benchmark has the largest spread. To further visualize a comparison, we focus on the date 4 February 2016. Figures 14–17 show 3D plots from four different

angles to compare their performance. Similarly, Figures 18–21 show 2D plots conditional on four different levels of τ that corresponds to the 20th, 35th, 45th, and 65th percentiles. Overall, the dynamic additive B-splines model has a better fit than the other three models. We notice that the nonparametric benchmark gives an irregular and uneven predicted implied volatility surface, and this can be due to the lack of constraints on the shape of IVS. For the parametric benchmark, because it is too restrictive on the shape and there is a risk of model mis-specification, it is not flexible enough to account for the spread of the data. Therefore, the estimated coefficients as predictors in dynamic parametric benchmarks are fundamentally bad predictors that worsen the prediction of the IVS. The risk of mis-specification and inaccurate predictions of the coefficients leads to the poorest performance of the dynamic parametric benchmark.

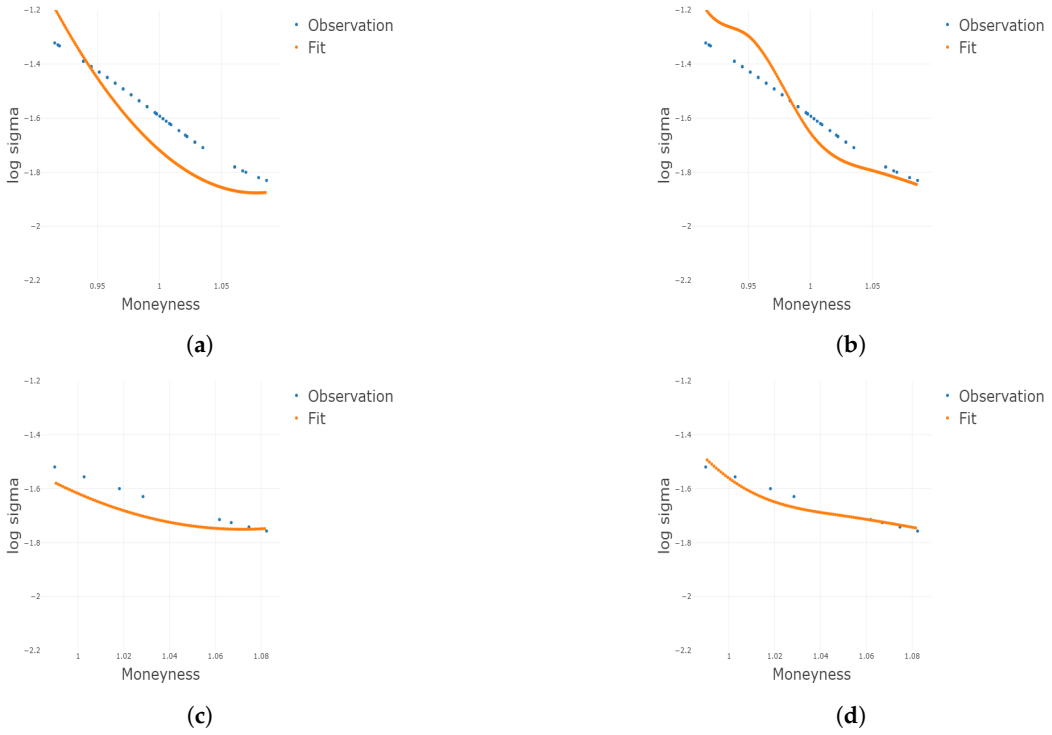


Figure 11. Four 2D plots conditional on the 65th and 80th percentile of τ compare the in-sample fit of IVS from the parametric benchmark and additive B-splines model on the date 5 February 2021. The additive B-splines model has a better fit to the data. (**a**) Parametric benchmark model fit conditional on the 65th percentile, (**b**) additive B-splines model fit conditional on the 65th percentile, (**c**) parametric benchmark model fit conditional on the 80th percentile, and (**d**) additive B-splines model fit conditional on the 80th percentile.

Table 7. The RMSE and relative RMSE are provided for the dynamic additive B-splines, nonparametric benchmark, parametric benchmark, and dynamic parametric benchmark model. The dynamic additive B-splines model has lowest (relative) RMSE, followed by the nonparametric benchmark, parametric benchmark, and dynamic parametric benchmark model.

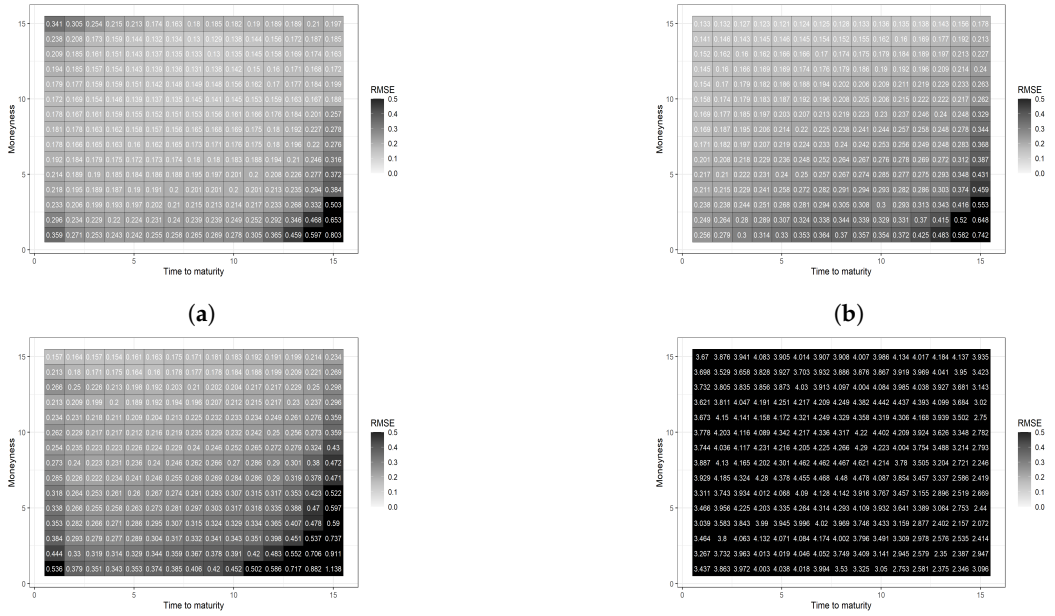
	Dynamic Additive B-Splines	**Nonparametric Benchmark**	**Parametric Benchmark**	**Dynamic Parametric Benchmark**
RMSE	0.2016	0.2455	0.2962	3.8606
Relative RMSE	0.1057	0.1287	0.1553	2.0242

Table 8. Calculate the RMSE for the in-the-money (ITM) options and out-of-money options (OTM) under the four models. The dynamic additive B-splines model consistently has the lowest RMSE.

	Dynamic Additive B-Splines	Nonparametric Benchmark	Parametric Benchmark	Dynamic Parametric Benchmark
ITM ($m < 1$)	0.2011	0.2588	0.30971	3.7824
OTM ($m > 1$)	0.2022	0.2234	0.2742	3.9781

Table 9. Calculate the RMSE for the overvalued options and undervalued options under two models. The dynamic additive B-splines model has a consistently smaller RMSE. The option is overvalued when the predicted implied volatility is greater than the observed implied volatility.

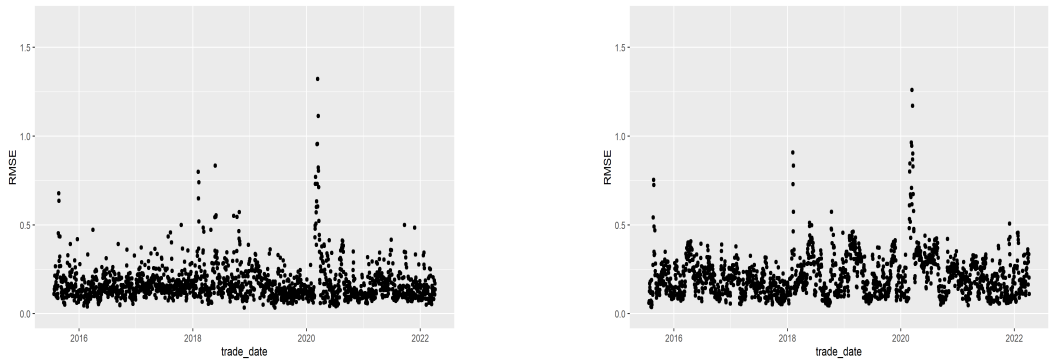
	Dynamic Additive B-Splines	Nonparametric Benchmark	Parametric Benchmark	Dynamic Parametric Benchmark
Overvalued	0.1813	0.2319	0.2193	3.8242
Undervalued	0.2303	0.2733	0.3669	3.9086



(**a**) (**b**)

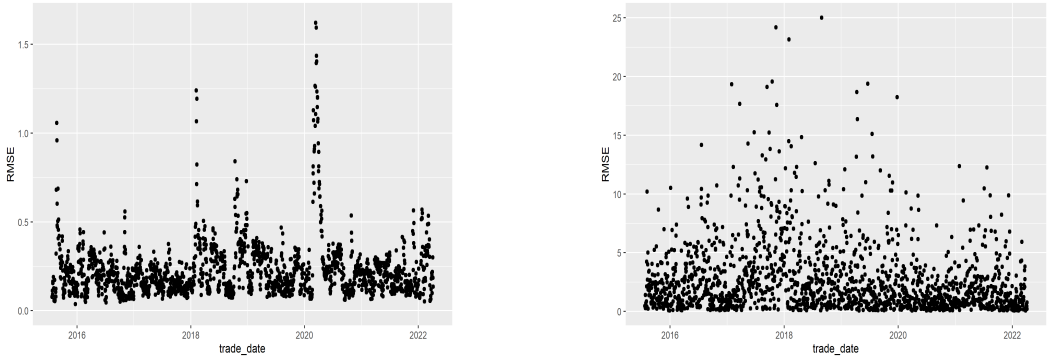
(**c**) (**d**)

Figure 12. Four heatmaps are based on a 15 by 15 grid determined by equal-spaced percentiles in moneyness and time to maturity. The dynamic parametric benchmark has the darkest heatmap, while the dynamic additive B-splines model has the brightest, indicating the best fit among the four models. (**a**) Dynamic additive B-splines model, (**b**) nonparametric benchmark, (**c**) parametric benchmark, and (**d**) dynamic parametric benchmark.



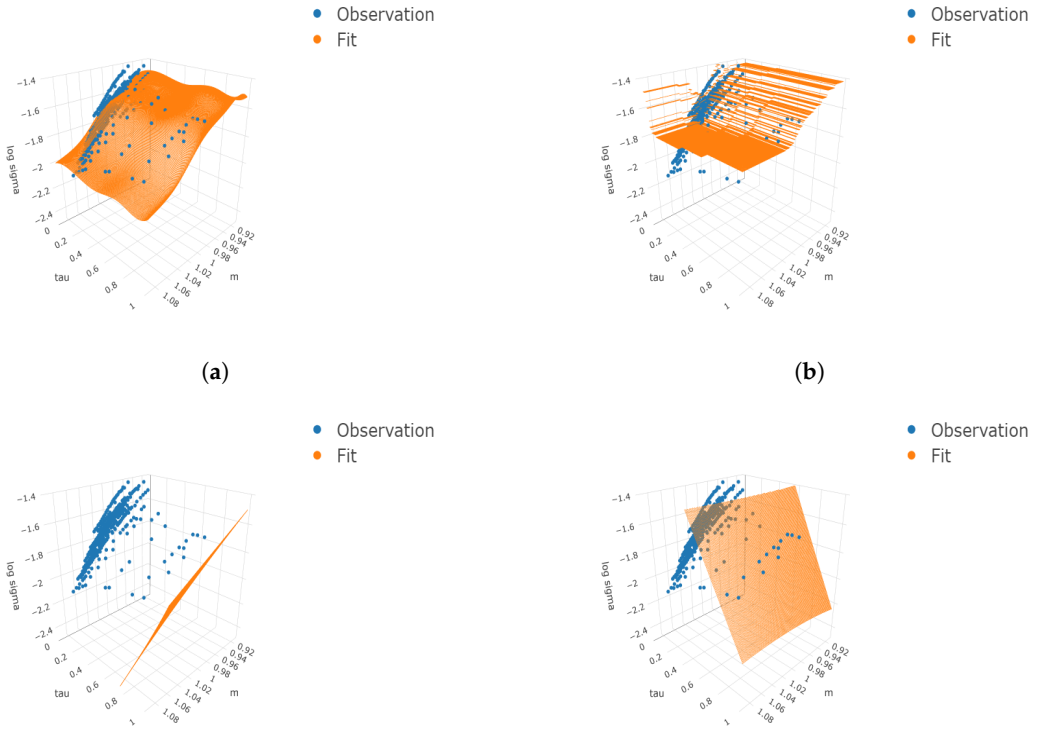
(**a**) (**b**)

Figure 13. *Cont.*



(**c**) (**d**)

Figure 13. Compare the RMSE time series across the trade date for the four models. The dynamic additive B-splines model has the most stable time series with the narrowest band and fewest extremes, while the dynamic parametric benchmark has the largest spread. (**a**) RMSE time series for dynamic additive B-splines, (**b**) RMSE time series for nonparametric benchmark, (**c**) RMSE time series for parametric benchmark, and (**d**) RMSE time series for dynamic parametric benchmark.



(**a**) (**b**)

(**c**) (**d**)

Figure 14. Four 3D plots with angle 1 compare the in-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit to the data. (**a**) Angle 1 for the dynamic additive B-splines model, (**b**) angle 1 for nonparametric benchmark, (**c**) angle 1 for parametric benchmark, and (**d**) angle 1 for dynamic parametric benchmark.

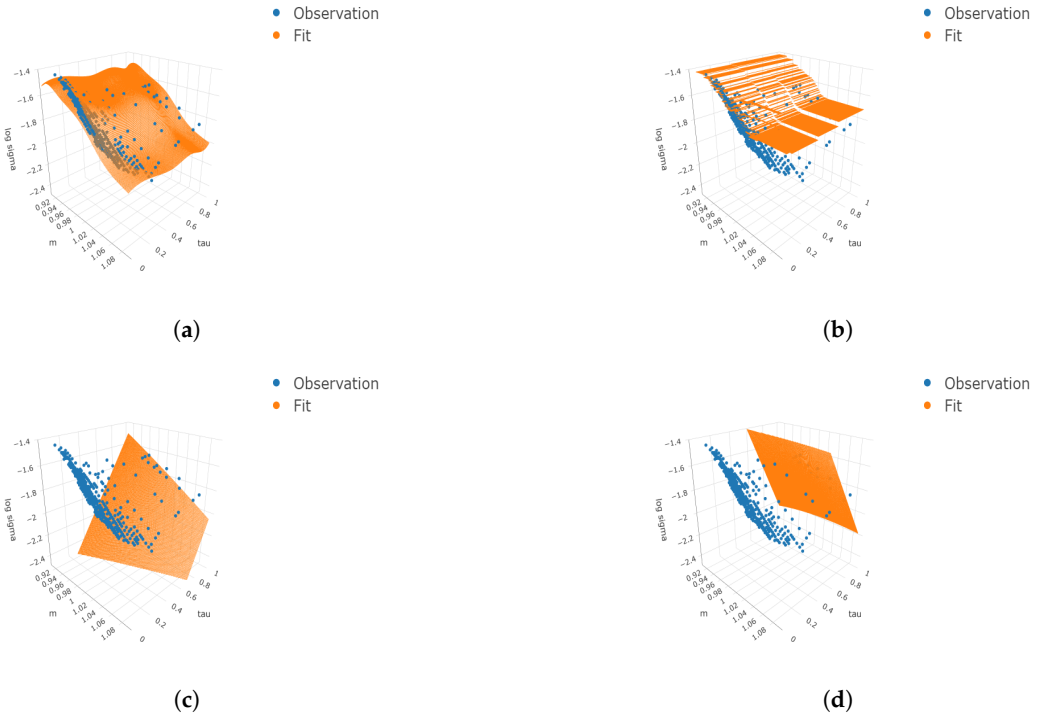


Figure 15. Four 3D plots with angle 2 compare the in-sample fit of IVS from four models on the date 24 February 2016. The additive B-splines model has the best fit to the data. (**a**) Angle 2 for the dynamic additive B-splines model, (**b**) angle 2 for nonparametric benchmark, (**c**) angle 2 for parametric benchmark, and (**d**) angle 2 for dynamic parametric benchmark.

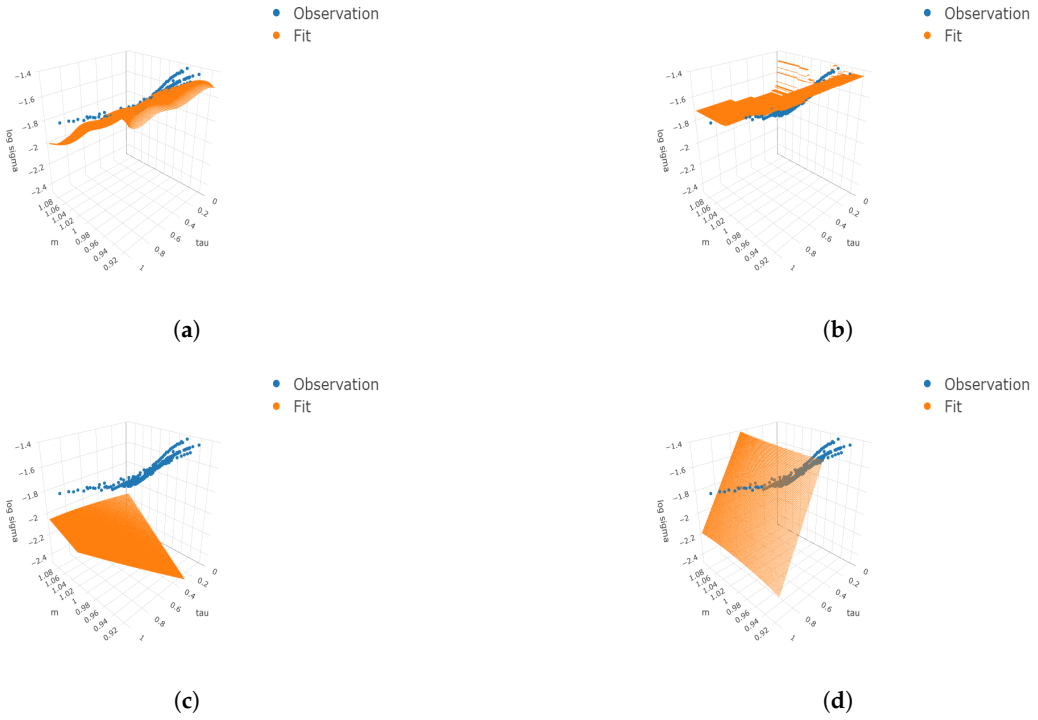


Figure 16. Four 3D plots with angle 3 compare the in-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit to the data. (**a**) Angle 3 for the dynamic additive B-splines model, (**b**) angle 3 for nonparametric benchmark, (**c**) angle 3 for parametric benchmark, and (**d**) angle 3 for dynamic parametric benchmark.

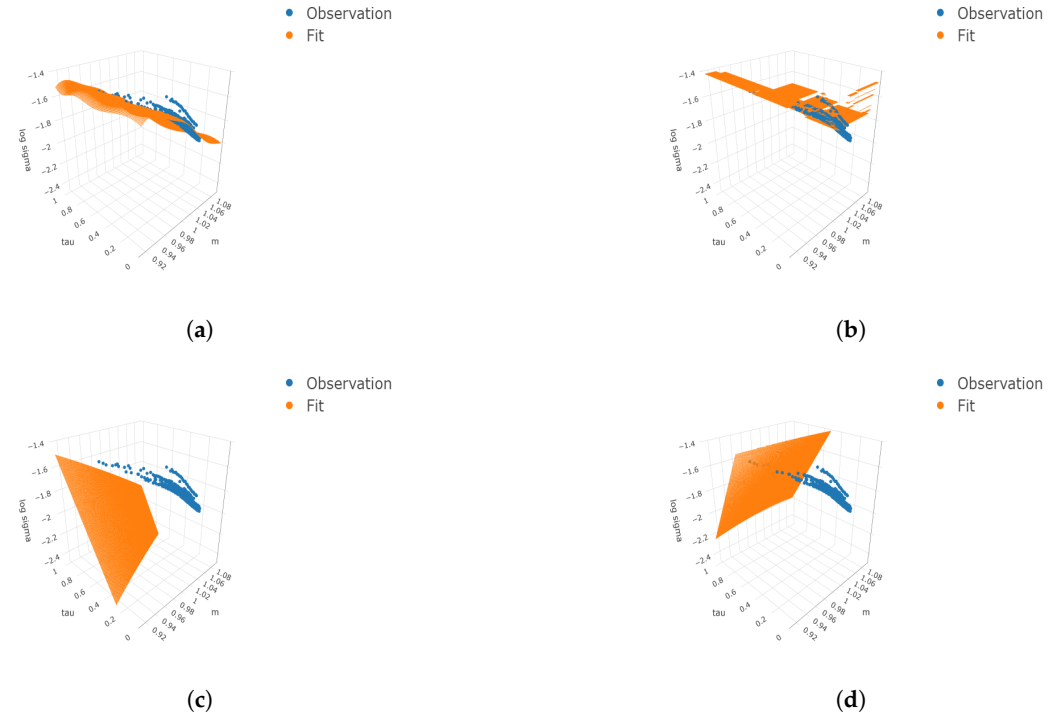


Figure 17. Four 3D plots with angle 4 compare the in-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit to the data. (**a**) Angle 4 for the dynamic additive B-splines model, (**b**) angle 4 for nonparametric benchmark, (**c**) angle 4 for parametric benchmark, and (**d**) angle 4 for dynamic parametric benchmark.

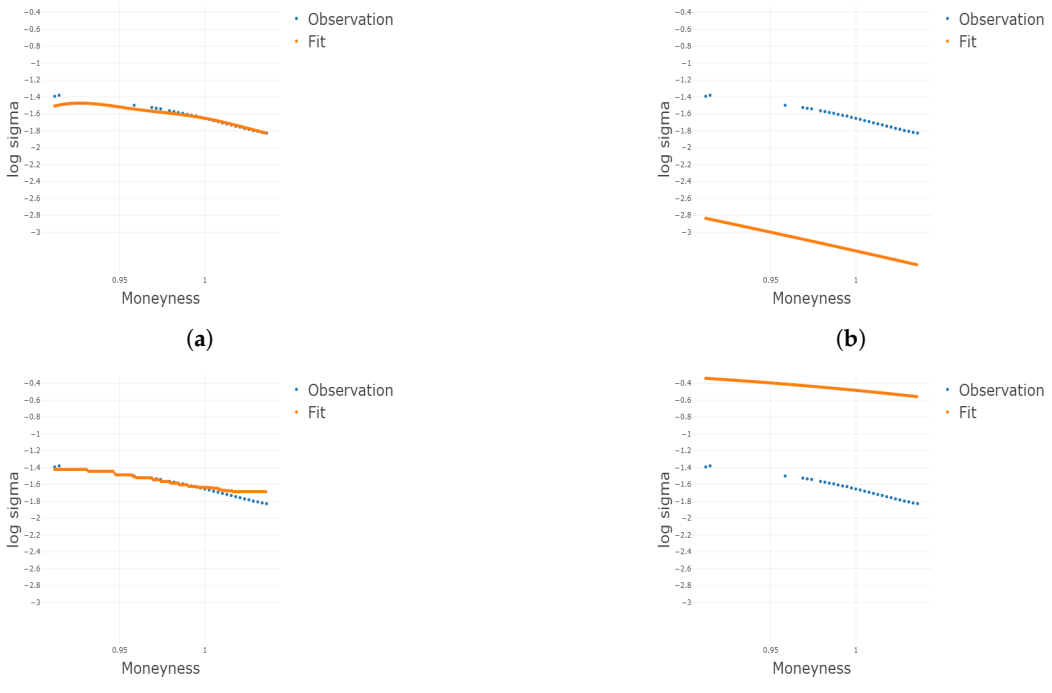


Figure 18. Four 2D plots conditional on the 20th percentile of τ compare the out-of-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit to the data. (**a**) Dynamic additive B-splines model fit conditional on the 20th percentile, (**b**) parametric benchmark model fit conditional on the 20th percentile, (**c**) nonparametric benchmark model fit conditional on the 20th percentile, and (**d**) dynamic parametric benchmark model fit conditional on the 20th percentile.

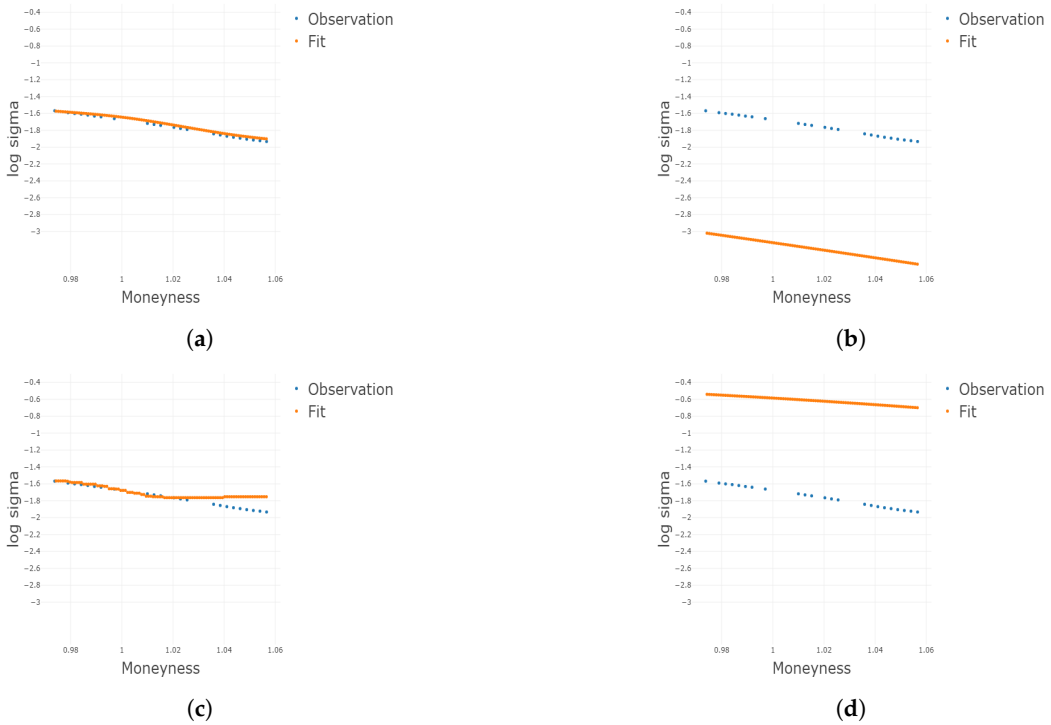


Figure 19. Four 2D plots conditional on the 35th percentile of τ compare the out-of-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit to the data. (**a**) Dynamic additive B-splines model fit conditional on the 35th percentile, (**b**) parametric benchmark model fit conditional on the 35th percentile, (**c**) nonparametric benchmark model fit conditional on the 35th percentile, and (**d**) dynamic parametric benchmark model fit conditional on the 35th percentile.

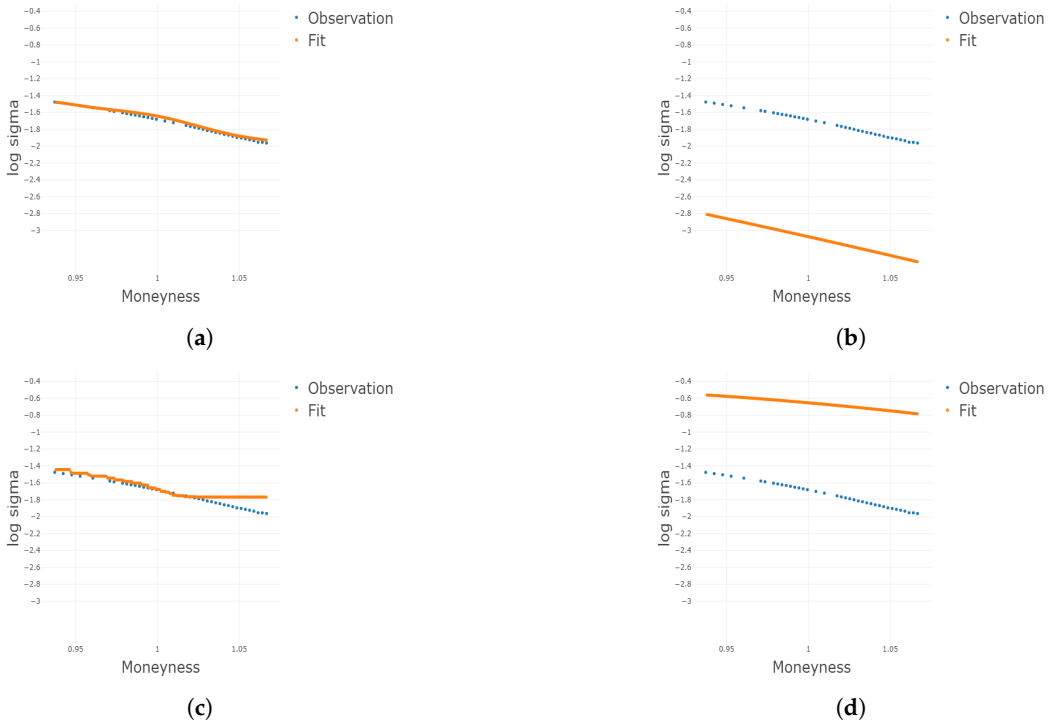


Figure 20. Four 2D plots conditional on the 45th percentile of τ compare the out-of-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit

to the data. (**a**) Dynamic additive B-splines model fit conditional on the 45th percentile, (**b**) parametric benchmark model fit conditional on the 45th percentile, (**c**) nonparametric benchmark model fit conditional on the 45th percentile, and (**d**) dynamic parametric benchmark model fit conditional on the 45th percentile.

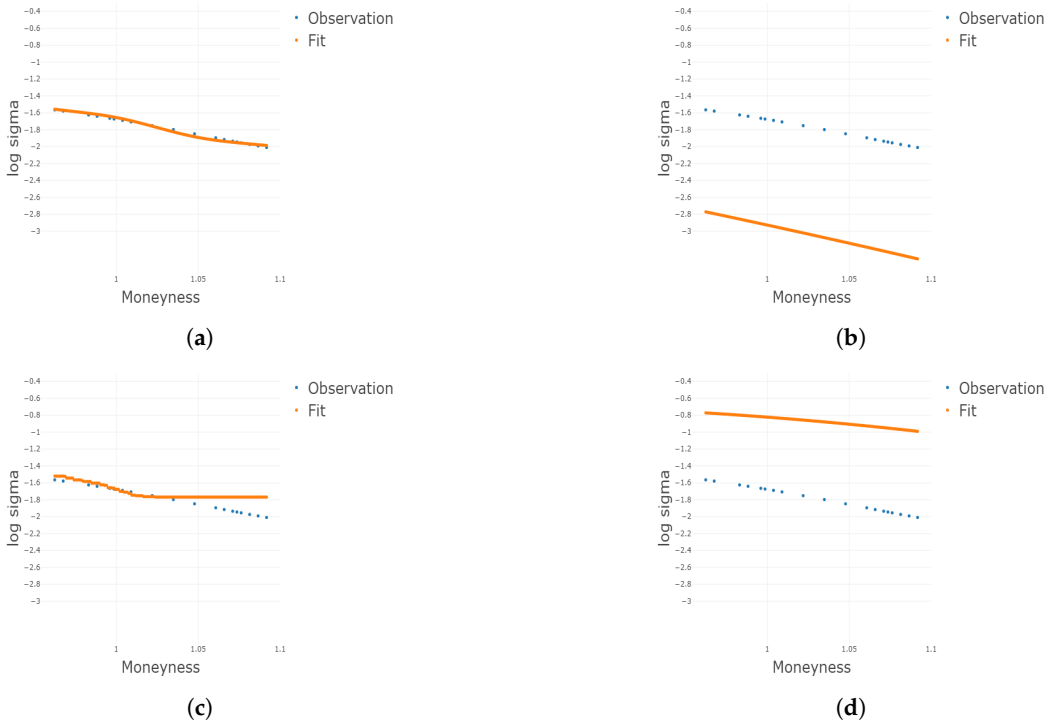


Figure 21. Four 2D plots conditional on the 65th percentile of τ compare the out-of-sample fit of IVS from four models on the date 24 February 2016. The dynamic additive B-splines model has the best fit to the data. (**a**) Dynamic additive B-splines model fit conditional on the 65th percentile, (**b**) parametric benchmark model fit conditional on the 65th percentile, (**c**) nonparametric benchmark model fit conditional on the 65th percentile, and (**d**) dynamic parametric benchmark model fit conditional on the 65th percentile.

5. Conclusions and Future Work

Traditional parametric models for IVS are too restrictive on the form of the model, while the nonparametric model using machine learning provides more flexibility but fails to take the volatility smile structure into account. In this paper, we propose a dynamic semiparametric approach that combines the use of tensor-product B-splines and tree-based machine learning methods that allow for not only the flexibility and the constraints of the shape of IVS but also the dynamics of the IVS over time by considering the time-dependent coefficients.

The simulation study shows that BTPB is able to recover the classical parametric benchmark model under different cases, and the increase in sample size improves the prediction of the IVS. In the empirical study, we show that our dynamic semiparametric model, whose coefficients are modeled by gradient boosting trees, has the best performance in predicting the IVS for the *S&P* 500 index options, followed by the nonparametric benchmark, parametric benchmark, and dynamic parametric benchmark model.

In the future, we will consider the arbitrage opportunity by constructing portfolios based on the directions of the predicted implied volatility; i.e., if we predict that the implied volatility for an option is to increase in the next trade date, we are going to be long on the option and short on the underlying stock because the increase in implied volatility causes the increase in option price.

Author Contributions: Conceptualization, Z.C., Y.L. and C.Y.; methodology, Z.C., Y.L. and C.Y.; software, Z.C. and Y.L.; validation, Z.C., Y.L. and C.Y.; formal analysis, Z.C. and Y.L.; investigation, Z.C. and Y.L.; resources, Z.C. and Y.L.; data curation, Z.C. and Y.L.; writing—original draft preparation, Z.C. and Y.L.; writing—review and editing, Z.C., Y.L. and C.Y.; visualization, Z.C. and Y.L.; supervision, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data were obtained from OptionMetrics and are available at https://optionmetrics.com/ with the permission of OptionMetrics.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Bayesian optimization is used for solving black-box optimization problems where the objective function is unknown or difficult to evaluate. Bayesian optimization replaces the objective function by a Gaussian process so that we can use a sequence of optimizations to approximate the original problem.

The hyperparameter tuning in machine learning can be treated as a Bayesian optimization problem where we want to maximize the metric on the validation set with respect to the hyperparameters. In the context of hyperparameter tuning in machine learning, the response variable is the metric value on the validation set and covariates are the hyperparameters of machine learning methods. The meanings of notations are as follows:

$$
\begin{aligned}
v_i &= i \text{ th value of metric on validation set where } i = 1, 2, \ldots, n.\\
\mathbf{v} &= (v_1, v_2, \ldots, v_n)'.\\
\omega_{ij} &= i \text{ th value of hyperparameter } j \text{ where } i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, q.\\
\boldsymbol{\omega}_j &= (\omega_{1j}, \omega_{2j}, \ldots, \omega_{nj})'.\\
\boldsymbol{\omega}_i &= (\omega_{i1}, \omega_{i2}, \ldots, \omega_{iq})'.\\
\boldsymbol{\Omega} &= n \times q \text{ matrix with } ij \text{ th element } \omega_{ij}.\\
f(\boldsymbol{\omega}) &= \text{the true underlying function for the metric on validation set w.r.t. hyperparameters.}
\end{aligned}
$$

We denote as n the number of grids in grid search or the number of iteration in a Bayesian approach, and q is the number of hyperparameters to tune. For example, in principal component regression, the number of components that selected M is the only hyperparameter, and hence, we have one covariate, i.e., $q = 1$. If we use R^2 as the metric, then R^2 on the validation set is the response variable.

A Gaussian process is a collection $\{f(\boldsymbol{\omega}), \boldsymbol{\omega} \in \mathbb{R}^q\}$ where any point $\boldsymbol{\omega} \in \mathbb{R}^q$ is assigned a random variable $f(\boldsymbol{\omega})$ and the joint distribution of a finite number $n \in \mathbb{N}$ of these variables $\mathbf{f} = (f(\boldsymbol{\omega}_1), \ldots, f(\boldsymbol{\omega}_n))$ satisfies a Gaussian distribution Williams and Rasmussen [27]:

$$\mathbf{f}|\boldsymbol{\omega} \sim N(\boldsymbol{\mu}, \mathbf{K}), \tag{A1}$$

where $\boldsymbol{\mu} = (m(f(\boldsymbol{\omega}_1)), \ldots, m(f(\boldsymbol{\omega}_n)))$ and $\mathbf{K}_{i,j} = cov(f(\boldsymbol{\omega}_i), f(\boldsymbol{\omega}_j))$. For $i = 1, 2, \ldots, n$, $m(\boldsymbol{\omega}_i) = \mathbb{E}[f(\boldsymbol{\omega}_i)]$ is the mean function, and it is common to use $m(\boldsymbol{\omega}_i) = \mathbf{0}$ without loss of generality. $\mathbf{K}$ is the covariance matrix (kernel) that defines the shape of the Gaussian process. One popular kernel is the Matern kernel, given by the following:

$$K(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j) = \frac{1}{\Gamma(\rho)2^{\rho-1}}\left(\frac{\sqrt{2\rho}}{\kappa} d(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j)\right)^{\rho} k_{\rho}\left(\frac{\sqrt{2\rho}}{\kappa} d(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j)\right), \tag{A2}$$

where $d(.,.)$ is the Euclidean distance, $k_{\rho}(.)$ is the modified Bessel function, and ρ, κ are two positive parameters that control the smoothness, with default values $\rho = 1.5$, $\kappa = 1.0$.

For data $\{(\boldsymbol{\omega}_i, v_i)\}_{i=1}^n$, the goal of Gaussian process regression (GPR) is to predict $\mathbf{f}^* \equiv f(\boldsymbol{\Omega}^*)$ given new inputs $\boldsymbol{\omega}^* \in \mathbb{R}^{n^* \times q}$ Gonzalvez et al. [28]. Assume $\mathbf{v} = \mathbf{f} + \boldsymbol{\epsilon}$, where $\mathbf{f} = f(\boldsymbol{\Omega}) = (f(\boldsymbol{\omega}_1), \ldots, f(\boldsymbol{\omega}_n))$ and $\boldsymbol{\epsilon} \sim N(\mathbf{0}_n, \sigma^2 \mathbf{I}_n)$. Then the joint distribution of observed $\mathbf{v}$ and prediction $\mathbf{f}^*$ is as follows:

$$\begin{pmatrix} \mathbf{v} \\ \mathbf{f}^* \end{pmatrix} \sim N(\mathbf{0}_{n+n^*}, \begin{pmatrix} \mathbf{K}(\boldsymbol{\Omega}, \boldsymbol{\Omega}) + \sigma^2 \mathbf{I}_n & \mathbf{K}(\boldsymbol{\Omega}, \boldsymbol{\Omega}^*) \\ \mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega}) & \mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega}^*) \end{pmatrix}),$$

where $\mathbf{K}(\boldsymbol{\Omega}, \boldsymbol{\Omega})$ is a $n \times n$ matrix, $\mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega})$ is a $n^* \times n$ matrix, and $\mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega}^*)$ is a $n^* \times n^*$ matrix.

A Gaussian prior $p(\mathbf{f}|\boldsymbol{\Omega})$ can be converted into a Gaussian posterior $p(\mathbf{f}|\boldsymbol{\Omega}, \mathbf{v})$ after having observed $\{(\boldsymbol{\omega}_i, v_i)\}_{i=1}^n$. The posterior $p(\mathbf{f}|\boldsymbol{\Omega}, \mathbf{v}) \propto p(\mathbf{f}|\boldsymbol{\Omega}) p(\mathbf{v}|\boldsymbol{\Omega}, \mathbf{f})$ can then be used to make predictions for a given new input $\boldsymbol{\Omega}^*$ through (Williams and Rasmussen [27]):

$$p(\mathbf{f}^*|\boldsymbol{\Omega}^*, \boldsymbol{\Omega}, \mathbf{v}) = \int p(\mathbf{f}^*|\boldsymbol{\Omega}^*, \mathbf{f}) p(\mathbf{f}|\boldsymbol{\Omega}, \mathbf{v}) d\mathbf{f}, \tag{A3}$$

i.e., the posterior predictive distribution. By the rules of conditional distribution of two Gaussian distributions, the predictive distribution is as follows:

$$p(\mathbf{f}^*|\boldsymbol{\Omega}^*, \boldsymbol{\Omega}, \mathbf{v}) = N(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*), \tag{A4}$$

where $\boldsymbol{\mu}^* = \boldsymbol{\mu}^*(\boldsymbol{\Omega}^*) = \mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega})(\mathbf{K}(\boldsymbol{\Omega}, \boldsymbol{\Omega}) + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{v}$ and $\boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}^*(\boldsymbol{\Omega}^*) = \mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega}^*) - \mathbf{K}(\boldsymbol{\Omega}^*, \boldsymbol{\Omega})(\mathbf{K}(\boldsymbol{\Omega}, \boldsymbol{\Omega}) + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{K}(\boldsymbol{\Omega}, \boldsymbol{\Omega}^*)$. The deduced prediction is the conditional expectation $\boldsymbol{\mu}^*$.

Bayesian optimization with a Gaussian process takes two steps to find $\boldsymbol{\omega}$ that maximizes $f(\boldsymbol{\omega})$. In the first step, construct a surrogate model through a Gaussian process prior; i.e., we impose a Gaussian prior on $f(\boldsymbol{\omega})$ and have a posterior predictive distribution for $f(\boldsymbol{\omega})$ after incorporating the observed samples from $f(\boldsymbol{\omega})$. Because the posterior of a Gaussian process is still a Gaussian distribution, Gaussian process is one of the popular surrogate priors. In the second step, we pick a new point $\boldsymbol{\omega}^*$ based on the so-called acquisition function (or utility function) under the posterior distribution from the first step. The objective function $f(\boldsymbol{\omega})$ is then evaluated at the new point $\boldsymbol{\omega}^*$ with the potential of achieving a higher objective function value. Incorporate the new sample to the sample data and update the posterior predictive distribution. The second step is repeated until a desired maximum of the objective function is approximated. Algorithm A1 provides the details.

Algorithm A1 Bayesian optimization with a Gaussian process

Initialize a data sample set D_1 of size n_1 where n_1 is a fixed number.
for $k = 1, 2, \ldots$ **do**
Find the next $\boldsymbol{\omega}_{k+1}$ as:

$$\boldsymbol{\omega}_{k+1} = \arg max_{\boldsymbol{\omega}} a_k(\boldsymbol{\omega})$$

Augment the observed sample by including $\{(\boldsymbol{\omega}_{k+1}, v_{k+1})\}$

$$D_{k+1} = D_k \cup \{(\boldsymbol{\omega}_{k+1}, v_{k+1})\}$$

end for
return D_{k+1}

An acquisition function directs the sampling to areas where there is a potential improvement over the current best observation. In Equation (A4), we show that the posterior predictive distribution is $p(\mathbf{f}^*|\boldsymbol{\Omega}^*, \boldsymbol{\Omega}, \mathbf{v}) = N(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ for new inputs $\boldsymbol{\Omega}^*$.

The acquisition function proposes new sampling locations in an iterative way. Denote the sample data at iteration k as D_k. Let $\mathbf{f}_k^* = f_k(\boldsymbol{\Omega}^*) \sim N(\boldsymbol{\mu}_k^*, \boldsymbol{\Sigma}_k^*)$ denote the prediction whose posterior distribution is obtained after incorporating the observed sample D_k. Let

$a_k(\boldsymbol{\omega})$ be the acquisition function associated with D_k. The goal is to find a new point $\boldsymbol{\omega}_{k+1}$ through the acquisition function that maximizes $a_k(\boldsymbol{\omega})$:

$$\boldsymbol{\omega}_{k+1} = \arg max_{\boldsymbol{\omega}} a_k(\boldsymbol{\omega}), \tag{A5}$$

and then use $D_{k+1} = D_k \cup \{(\boldsymbol{\omega}_{k+1}, v_{k+1})\}$ to update the posterior predictive distribution of $\mathbf{f}^*_{k+1}$. The posterior distribution is then used for the evaluation of the acquisition function.

One popular choice of acquisition function is expected improvement Mockus et al. [29], Snoek et al. [30]:

$$EI(\boldsymbol{\omega}) = \mathbb{E}(max(f(\boldsymbol{\omega}) - v_k^+), 0), \tag{A6}$$

where v_k^+ is the highest value in current samples. The EI is trying to find a new point with a higher objective function value. In our context, denote $(\boldsymbol{\omega}_k^+, v_k^+)$ as the optimal in the observed samples D_k; i.e., v_k^+ takes the maximum response value in D_k. We pick $\boldsymbol{\omega}_{k+1}$ as follows:

$$\boldsymbol{\omega}_{k+1} = \arg\max_{\boldsymbol{\omega}} EI_k(\boldsymbol{\omega}) = \arg\max_{\boldsymbol{\omega}} \mathbb{E}(max(f_k(\boldsymbol{\omega}) - v_k^+), 0). \tag{A7}$$

It can be shown as follows:

$$EI_k(\boldsymbol{\omega}) = (\boldsymbol{\mu}_k^*(\boldsymbol{\omega}) - v_k^+))\Phi(\frac{\boldsymbol{\mu}_k^*(\boldsymbol{\omega}) - v_k^+)}{\sqrt{\boldsymbol{\Sigma}_k^*(\boldsymbol{\omega})}}) + \sqrt{\boldsymbol{\Sigma}_k^*(\boldsymbol{\omega})}\phi(\frac{\boldsymbol{\mu}_k^*(\boldsymbol{\omega}) - v_k^+)}{\sqrt{\boldsymbol{\Sigma}_k^*(\boldsymbol{\omega})}}), \tag{A8}$$

where $\Phi(.)$ and $\phi(.)$ represent the cumulative distribution function and density function of a standard normal distribution, respectively.

It is also likely to consider a parameter ξ that balances the exploration and exploitation Jones et al. [31]:

$$EI_k(\boldsymbol{\omega}) = \begin{cases} (\boldsymbol{\mu}_k^*(\boldsymbol{\omega}) - v_k^+) - \xi)\Phi(C) + \sqrt{\boldsymbol{\Sigma}_k^*}\phi(C) & \boldsymbol{\Sigma}_k^*(\boldsymbol{\omega}) > 0 \\ 0 & \boldsymbol{\Sigma}_k^*(\boldsymbol{\omega}) = 0 \end{cases}, \tag{A9}$$

where $C = \begin{cases} \frac{\boldsymbol{\mu}_k^*(\boldsymbol{\omega}) - v_k^+ - \xi)}{\sqrt{\boldsymbol{\Sigma}_k^*(\boldsymbol{\omega})}} & \boldsymbol{\Sigma}_k^*(\boldsymbol{\omega}) > 0 \\ 0 & \boldsymbol{\Sigma}_k^*(\boldsymbol{\omega}) = 0 \end{cases}$.

The first term in Equation (A9) is the exploitation term, and the second term in Equation (A9) is the exploration term. Parameter ξ determines the amount of exploration during optimization, and higher ξ values lead to more exploration. With increasing ξ values, the importance of potential improvements from the GP posterior mean decreases relative to the importance of potential improvements in regions of high prediction uncertainty, which is represented by a large covariance value Σ_k^*.

The grid search of hyperparameters is completely uninformed by past evaluations, and hence spends a significant amount of time evaluating "unpromising" hyperparameters. Bayesian optimization with GP, however, attempts to solve the black-box problem by imposing a Gaussian prior on the unknown objective function and proposing more "promising" points that can yield a higher value of the objective function through the acquisition function. Therefore, a Bayesian optimization approach updates the hyperparameter in a more appropriate direction with the knowledge of past observations based on the correlation imposed by a covariance structure, thus locating the optimal hyperparameter more efficiently. More importantly, because the observations (metric values on a validation set in our context) are noisy, i.e., $\mathbf{v} = f(\boldsymbol{\omega}) + \epsilon$, it is difficult to find the $\boldsymbol{\omega}$ that maximizes the underlying function $f(\cdot)$ directly using $\mathbf{v}$, which is contaminated with error. Bayesian optimization with GP creates a way of approximating the true underlying $f(\cdot)$ and produces a posterior predictive distribution that is also Gaussian such that the problem becomes tractable.

References

1. Dumas, B.; Fleming, J.; Whaley, R.E. Implied volatility functions: Empirical tests. *J. Financ.* **1998**, *53*, 2059–2106. [CrossRef]
2. Goncalves, S.; Guidolin, M. Predictable dynamics in the S&P 500 index options implied volatility surface. *J. Bus.* **2006**, *79*, 1591–1635.
3. Heston, S.L.; Nandi, S. A closed-form GARCH option valuation model. *Rev. Financ. Stud.* **2000**, *13*, 585–625. [CrossRef]
4. Aït-Sahalia, Y.; Lo, A.W. Nonparametric estimation of state-price densities implicit in financial asset prices. *J. Financ.* **1998**, *53*, 499–547. [CrossRef]
5. Fengler, M.R.; Härdle, W.K.; Villa, C. The dynamics of implied volatilities: A common principal components approach. *Rev. Deriv. Res.* **2003**, *6*, 179–202. [CrossRef]
6. Audrino, F.; Colangelo, D. Semi-parametric forecasts of the implied volatility surface using regression trees. *Stat. Comput.* **2010**, *20*, 421–434. [CrossRef]
7. Das, S.P.; Padhy, S. A new hybrid parametric and machine learning model with homogeneity hint for European-style index option pricing. *Neural Comput. Appl.* **2017**, *28*, 4061–4077. [CrossRef]
8. Hahn, J.T. *Option Pricing Using Artificial Neural Networks: An Australian Perspective*; Citeseer: Princeton, NJ, USA, 2013.
9. Luo, J.; Yu, C.L. The Application of Symbolic Regression on Identifying Implied Volatility Surface. *Mathematics* **2023**, *11*, 2108. [CrossRef]
10. Shimko, D. Bounds of probability. *Risk* **1993**, *6*, 33–37.
11. Bliss, R.R.; Panigirtzoglou, N. Option-implied risk aversion estimates. *J. Financ.* **2004**, *59*, 407–446. [CrossRef]
12. Figlewski, S. Estimating the Implied Risk Neutral Density. 2008. Available online: https://ssrn.com/abstract=1256783 (accessed on 27 August 2008).
13. Orosi, G. Empirical performance of a spline-based implied volatility surface. *J. Deriv. Hedge Funds* **2012**, *18*, 361–376. [CrossRef]
14. Eilers, P.H.; Marx, B.D. Flexible smoothing with B-splines and penalties. *Stat. Sci.* **1996**, *11*, 89–121. [CrossRef]
15. Eilers, P.H.; Marx, B.D. Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemom. Intell. Lab. Syst.* **2003**, *66*, 159–174. [CrossRef]
16. Marx, B.D.; Eilers, P.H. Multidimensional penalized signal regression. *Technometrics* **2005**, *47*, 13–22. [CrossRef]
17. Wood, S.N. Thin plate regression splines. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2003**, *65*, 95–114. [CrossRef]
18. Wood, S.N. Low-rank scale-invariant tensor product smooths for generalized additive mixed models. *Biometrics* **2006**, *62*, 1025–1036. [CrossRef] [PubMed]
19. Xiao, L.; Li, Y.; Ruppert, D. Fast bivariate P-splines: The sandwich smoother. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2013**, *75*, 577–599. [CrossRef]
20. Price, M.J.; Yu, C.L.; Hennessy, D.A.; Du, X. Are actuarial crop insurance rates fair?: An analysis using a penalized bivariate B-spline method. *J. R. Stat. Soc. Ser. C Appl. Stat.* **2019**, *68*, 1207–1232. [CrossRef]
21. Black, F.; Scholes, M. The pricing of options and corporate liabilities. *J. Political Econ.* **1973**, *81*, 637–654. [CrossRef]
22. Fengler, M.; Hardle, W.; Mammen, E. *A Dynamic Semiparametric Factor Modelfor Implied Volatility String Dynamics*; Technical Report, SFB 649 Discussion Paper; Humboldt University Berlin: Berlin, Germany, 2005.
23. Yoshida, T. Asymptotics for penalized spline estimators in quantile regression. *Commun. in Stat. Theory Methods* **2013**, *52*, 4815–4834. [CrossRef]
24. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009.
25. Gao, G.P.; Lu, X.; Song, Z. Tail risk concerns everywhere. *Manag. Sci.* **2019**, *65*, 3111–3130. [CrossRef]
26. Kim, H.J. Characterizing the Conditional Pricing Kernel: A New Approach. 2021. Available online: https://ssrn.com/abstract=4605072 (accessed on 14 November 2023)
27. Williams, C.K.; Rasmussen, C.E. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006; Volume 2.
28. Gonzalvez, J.; Lezmi, E.; Roncalli, T.; Xu, J. Financial applications of Gaussian processes and Bayesian optimization. *arXiv* **2019**, arXiv:1903.04841.
29. Mockus, J.; Tiesis, V.; Zilinskas, A. The application of Bayesian methods for seeking the extremum. *Towards Glob. Optim.* **1978**, *2*, 2.
30. Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1–9.
31. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [CrossRef]