Deep Surrogates for Finance: With an Application to Option Pricing

Hui Chen   Antoine Didisheim   Simon Scheidegger*

First draft: March 12, 2021
This draft: February 22, 2023

Abstract

We introduce "deep surrogates" – high-precision approximations of structural models based on deep neural networks, which speed up model evaluation and estimation by orders of magnitude and allow for a variety of compute-intensive applications that were previously intractable. As an application, we build a deep surrogate for a high-dimensional option pricing model. With the surrogate, we can re-estimate the model frequently and construct a measure of time-varying parameter instability, which we find to be positively associated with options market illiquidity empirically. The surrogate also makes it efficient to evaluate the structural model's out-of-sample pricing and hedging performances and helps reveal its advantages and weaknesses against a non-parametric model. Finally, the surrogate helps us extract model-implied conditional tail risk measures and predictive distributions of option returns.

Keywords: surrogate, deep neural network, Heston model, Bates model, parameter instability, illiquidity, option return, VaR

*Chen: MIT Sloan and NBER. Email: huichen@mit.edu. Didisheim: Department of Finance, HEC Lausanne, University of Lausanne and Swiss Finance Institute. Email: antoine.didisheim@unil. Scheidegger: Department of Economics, HEC Lausanne, University of Lausanne and Enterprise for Society. Email: simon.scheidegger@unil. We thank Lorenzo Bretscher, Xiaohong Chen, Mathieu Fournier, Andrew Lo, Semyon Malamud, Dmitriy Muravyev, John Rust, Olivier Scaillet, Fabio Trojani, seminar participants at the University of Geneva, the University of Lausanne, MIT, and participants of the 2021 CICF, the 2021 Econometrics Society Winter Meeting, the SIAM conference 2021, and the PASC 2021 conference, and the Econometric Society Summer School in Dynamic Structural Econometrics 2021 at Bonn for comments. This work is generously supported by grants from the Swiss National Supercomputing Center (CSCS) under project ID 995, and the Swiss National Science Foundation under project IDs "Can Economic Policy Mitigate Climate-Change?", and "New methods for asset pricing with frictions."

1 Introduction

Driven by advances in theory, the availability of big data, and gains in computing power, contemporary models in economics have become increasingly complex. State-of-the-art structural models often impose a substantial roadblock to researchers as they increasingly suffer from the curse of dimensionality, that is, the computational burden grows exponentially with every additional degree of freedom. The problem becomes even more challenging when our analysis calls for a large number of model evaluations. An example is structural estimations, especially with simulation-based methods such as simulated method of moments or particle filtering, or when the estimation needes to be done repeatedly to study parameter heterogeneity in a large cross section.

To tackle these challenges, we introduce *deep surrogates*, a new framework to evaluate and estimate structural models. Conceptually, our method re-purposes the concept of surrogate models commonly applied in physics and engineering for the context of economic models. By taking advantage of the strong approximating power of deep neural networks, we create high-precision, cheap-to-evaluate surrogates of high-dimensional models. The surrogate takes the same input from the original model, including the states as well as model parameters, which are treated as pseudo-states, and produces the same output at a significantly lower computational cost. Simply put, it is a deep-neural-network-based look-up table.

Not only does a deep surrogate help speed up function valuation, but it also helps alleviate the curse of dimensionality problem in many cases. Consider the task of approximating a univariate function by visiting n locations along the input state. Generalizing to a function with d states, this procedure requires evaluating the original function at $\mathcal{O}(n^d)$ locations in the d-dimensional state space. Even when a single function evaluation is relatively inexpensive, approximation of a high-dimensional function in this way can quickly become infeasible.

In contrast, one can train a deep neural network to accurately approximate a wide class of high-dimensional functions with substantially fewer observations (the training sample) than one would use with traditional methods based on a Cartesian grid. This is due to the approximation power of deep neural networks, as formally established by the universal approximation theorem, and their ability to achieve high accuracy with a relatively simple network structure (as typically measured by the total number of trainable parameters).[1] Intuitively, provided that the target function is sufficiently smooth, from a bounded domain, and the training examples are reasonably well scattered across the domain, a neural network

[1]See Gühring, Raslan, and Kutyniok (2020) for a recent survey on the large body of work of approximation results for deep neural networks.

is capable of "learning" its features without that many examples.[2]

As an application, we use the deep surrogate methodology to study a popular option pricing model, a double-exponential jump-diffusion model with stochastic volatility (which we refer to as the "Bates model" for simplicity, since it is a direct extension of Bates, 1996). The model features 14 states and parameters ($d = 14$). We train a surrogate using a deep neural network with seven hidden layers, 400 neurons in each layer, corresponding to nearly 10^6 trainable parameters. We achieve the desired approximation accuracy using a training sample of size $N = 10^9$, which is quite sparse given the number of dimensions.[3] Through simulation, we verify that the deep surrogate not only prices options and produces the greeks accurately, but also helps identify structural parameters accurately in GMM estimations. Moreover, it helps reduce the average time for estimating the parameters on a cross-section of 1,000 options from over 40 minutes using traditional methods to less than one second in our numerical experiments.[4]

The deep surrogate enables us to carry out a variety of analyses that would have incurred prohibitive computation costs otherwise. We study four of them in this paper: i) out-of-sample performance analysis; ii) parameter stability analysis; iii) exploring the information in the model-implied tail risk measures; iv) characterizing model-predicted conditional distributions of option returns.

First, we examine the out-of-sample performance of the Bates model and compare it against a reduced-form counterpart, which models the implied volatility surface using random forests (see, e.g., Breiman, 2001, we refer to it as the "RF model"). Conceptually, structural models can help guard against over-fitting by imposing restrictions between the dynamics of option prices and the underlying states but likely suffer from misspecification compared to the flexible machine learning models. The tradeoff between the two considerations determines the out-of-sample performances, which have been understudied in the option pricing literature. This is largely due to a computational challenge: in order to assess the out-of-sample performance of a structural model in the time series, we need to frequently update the structural parameters (typically by re-estimating them in a moving or expanding window) to capture potential parameter changes while avoiding any look-ahead bias. The deep surrogate substantially alleviates this problem.

Our empirical results confirm the tradeoff discussed above. While the RF model consis-

[2]For instance, neural networks have been shown to break or lessen the curse of dimensionality for compositional or polynomial functions (Bach, 2017; Petersen and Voigtlaender, 2018), functions in the Korobov spaces (Montanelli and Du, 2019), generalized bandlimited functions (Montanelli, Yang, and Du, 2021), and Black-Scholes PDEs (Berner, Grohs, and Jentzen, 2020).

[3]The trained surrogate is available at: `https://github.com/DeepSurrogate/OptionPricing`.

[4]The exact difference in computing time heavily depends on expert knowledge and code optimization.

tently produces smaller pricing errors (based on mean squared error) than the Bates model in sample, the out-of-sample performances vary depending on the type of options and market conditions. The RF model consistently outperforms the Bates model for short-dated options (with time-to-maturity of 7 days or less), while the opposite is true for longer-dated options (with maturity of 30 days or more), and mixed results for options with maturity between 7 and 30 days. Moreover, the Bates model tends to outperform the RF model when market volatility and jump risk are elevated, and it underperforms when option liquidity worsens (as measured by bid-ask spreads).

Next, we turn to the models' out-of-sample hedging performance. Indeed, one of the important tasks of option pricing models is to estimate the option delta: an option's price sensitivity to changes in the price of the underlying asset. This key estimation allows market makers and liquidity providers to hedge their position dynamically. For each option, we perform delta hedging at a daily frequency based on the model-specific theoretical deltas calculated using parameter values estimated on that day.

We show that the Bates model's replicating portfolio outperforms the RF on average. Unlike pricing errors, the difference in hedging is more evenly spread across maturities and moneyness. Furthermore, a regression controlling for other *Greeks*,[5] shows that the Bates' delta are less biased, and that they can explain 10% more of the prices variation compared to the one produced by the non-parametric benchmark.

These results highlight a fundamental weakness of non-parametric models: While parametric models aim to provide a simplified description of the world and, therefore, can be used for multiple purposes, non-parametric models tend to be task-oriented and thus can not trivially be applied to multiple objectives without rethinking the model itself.

The second question we examine is parameter stability. Using the statistical test proposed by Andersen, Fusari, and Todorov (2015), we construct a daily measure of parameter instability for the Bates model. As this measure only exists for parametric models, we use the HM instead of RF as a benchmark. The measure reveals that the estimated parameters for both structural models are unstable: the parameter estimates from two consecutive trading days differ significantly (at the 1% level) 41.6% of the time for the Bates and 60.7% of the time for the HM. Furthermore, the degree of parameter instability fluctuates significantly over time. More interestingly, we show that the parameter instability of the Bates models is significantly positively related to the options' bid-ask.

Third, through the daily re-estimation of the Bates model, we extract a model-implied tail risk measure. Previous research has shown that tail risks are priced by the market and

[5]The *Greeks* are the partial derivative of the options.

can be used to predict market returns (see, e.g., Kelly and Jiang, 2014; Andersen, Fusari, and Todorov, 2020). We use the weekly average of the parameters of the Bates model to predict market returns. These regressions with the model-implied tail risks produce higher in- and out-of-sample R^2 than a set of tail risks estimators from the extant literature.

Finally, we adapt the methodology by Israelov and Kelly (2017) to estimate the distribution of future option returns from our Bates and HM surrogate. In the first step, we use a bootstrap method on a vector autoregressive model to randomly draw 5000 pairs of volatility and stock prices. We then plug these values into our surrogates to compute an empirical distribution of the future option prices and returns. This methodology requires pricing each option 5000 times daily, making it computationally very costly in the absence of the surrogate technology. We use the empirical distributions to evaluate the Bates and HM surrogate models' performance at managing risks from a portfolio of options. We observe that Bates significantly outperforms the simpler HM in this application, especially for call options with long times to maturity.

While we use a relatively large neural network and train it for a large number of epochs, we do not use network shrinkage, dropout, or early stopping (see, e.g., Goodfellow, Bengio, Courville, and Bengio (2016)). Instead, we rely on the recently discovered double descent phenomenon (see, e.g., Stephenson and Lee, 2021; Nakkiran, Kaplun, Bansal, Yang, Barak, and Sutskever, 2021): the surprisingly high performance of a very complex network that completely interpolates the training sample. Indeed, with neural networks, the well-known bias-variance trade-off can be expressed as a function of model complexity or epochs of training. As training is refined or model complexity increases, the out-of-sample performance first increases. Then, as the training sample is over-fitted, the out-of-sample performance starts decreasing. Double-descent states that the out-of-sample performance increases again once the model complexity increases significantly after the in-training sample is fully interpolated. To the extent of our knowledge, we are among the first to use this recently described phenomenon in economics and finance research.

From a methodological perspective, this paper makes two contributions. First, we present the *deep surrogate* methodology and discuss how to use it in practical applications in economics and finance.[6] We demonstrate how to adopt the deep neural network architecture,

[6]In physics and engineering, Gaussian processes regression (see, e.g., Williams and Rasmussen, 2006; Tripathy, Bilionis, and Gonzalez, 2016; Bilionis and Zabaras, 2012a; Bilionis, Zabaras, Konomi, and Lin, 2013; Chen, Zabaras, and Bilionis, 2015), radial basis functions (Park and Sandberg, 1991), or relevance vector machines (Bilionis and Zabaras, 2012b) are often used to build surrogate models. More recently, following the rapid developments in the theory of stochastic optimization and artificial intelligence as well as the advances in computer hardware leading to the widespread availability of graphic processing units (GPUs; see, e.g., Scheidegger, Mikushin, Kubler, and Schenk (2018); Aldrich, Fernández-Villaverde, Gallant, and Rubio-Ramírez (2011), and references therein), researchers have turned their attention towards *deep neural*

the activation functions, the training procedure, and the simulated training sample to a given model's complexity, that is, the number of parameters and states, as well as the nonlinearity of the model. Second, we demonstrate the usefulness of cheap-to-evaluate surrogate models in controlled environments. With simulated data, we show that the *deep surrogate* can help us swiftly and precisely estimate the parameters of complex structural models.

Our paper contributes to three strands of the literature: (i) methods for constructing surrogate models in general and their application to high-dimensional models in economics and finance; (ii) applications of deep learning in finance and economics; and (iii) empirical option pricing.

Model estimation, calibration, and uncertainty quantification can be daunting numerical tasks because of the need to perform sometimes hundreds of thousands of model evaluations to obtain converging estimates of the relevant parameters and converging statistics (see, e.g., Fernández-Villaverde, Rubio-Ramírez, and Schorfheide, 2016; Fernández-Villaverde and Guerrón-Quintana, 2020; Iskhakov, Rust, and Schjerning, 2020; Igami, 2020, among others). To this end, a broad strand of literature in engineering, physics (see, e.g., Tripathy and Bilionis, 2018b), but also in finance and economics has long tried to replace expensive model evaluations that suffer from the curse of dimensionality with cheap-to-evaluate surrogate models that mitigate the said curse.[7] Heiss and Winschel (2008) for instance, approximate the likelihood by numerical integration on Smolyak sparse grids, whereas Scheidegger and Treccani (2018) apply adaptive sparse grids[8] to approximate high-dimensional probability density functions (PDFs) in the context of American option pricing. Scheidegger and Bilionis (2019) propose a method to carry out uncertainty quantification in the context of discrete-time dynamic stochastic models by combining the active subspace method (Constantine, 2015) with Gaussian processes to approximate the high-dimensional policies as a function of the endogenous and exogenous states as well as the parameters. Kaji, Manresa, and Pouliot (2020) propose a simulation-based estimation method for structural models in economics using a generative adversarial neural network.

In a related paper, Norets (2012) extends the state-space by adding the model parameters as "pseudo-states" to efficiently estimate finite-horizon, dynamic discrete choice models. He uses shallow artificial neural networks to approximate the dynamic programming solution

networks (see, e.g., Tripathy and Bilionis, 2018a; Liu, Borovykh, Grzelak, and Oosterlee, 2019).

[7]Over the course of the past two decades, there have been significant advances in the development of algorithms and numerical tools to compute global solutions to high-dimensional dynamic economic models (see Maliar and Maliar (2014) for a thorough review). This strand of literature is loosely related to the work presented here in that also, in the context of computing global solutions to dynamic models, high-dimensional functions have to be approximated repeatedly.

[8]See, e.g., Brumm and Scheidegger (2017); Pflueger, Peherstorfer, and Bungartz (2010), and references therein for more details on adaptive sparse grids.

as a function of parameters and state variables prior to estimation. The main difference in our methodology is the application of deep neural networks. Compared to shallow neural networks and some of the other popular approximation methods, the benefits that deep neural networks offer include approximating power, alleviating the curse of dimensionality by reducing the amount of training data required, and the ability to make efficient use of GPUs and big data. In particular, we show that a deep network can reproduce the parameter values significantly more accurately than shallow networks, which is crucial for structural estimation. We also confront our surrogates with real data in the context of empirical option pricing.

Secondly, our paper is part of the emergent literature on the applications of deep learning to economics and finance. In the seminal work of Hutchinson, Lo, and Poggio (1994), shallow neural nets are used for pricing and hedging derivative securities. Chen and White (1999) establish improved approximation error rate and root-mean-squared convergence rate for the non-parametric estimation of the conditional mean, conditional quantile, conditional densities of nonlinear time series using shallow neural networks and obtain root-n asymptotic normality of smooth functionals. Chen and Ludvigson (2009) use neural networks to solve habit-based asset pricing models. More recently, Farrell, Liang, and Misra (2021) derive non-asymptotic high probability bounds for deep feed-forward neural nets in the context of semiparametric inference without a pseudo-states approach. Buehler, Gonon, Teichmann, and Wood (2019) present a methodology called *deep hedging*, which uses deep neural networks and reinforcement learning to compute optimal hedging strategies directly from simulated data with transaction costs. Didisheim, Karyampas, and Scheidegger (2020) introduce the concept of *deep replication* to extract the implied risk aversion smile from options data. Chen, Pelger, and Zhu (2019) use deep neural networks to estimate an asset pricing model for individual stock returns that takes advantage of the vast amount of conditioning information. Becker, Cheridito, and Jentzen (2018) introduce *deep optimal stopping* for pricing Bermudan options. In contrast, several authors apply various formulations of deep neural networks to solve a broad range of high-dimensional dynamic stochastic models in discrete and continuous-time settings, but do not directly deal with estimation.[9]

We relate to this strand of literature in that we apply neural networks to tackle problems in finance. In particular, and to the best of our knowledge, we are the first to show that very deep neural networks combined with the *Swish* activation function (Ramachandran,

[9]For an incomplete list of research, see, e.g., Azinovic, Gaegauf, and Scheidegger (2022); Fernandez-Villaverde, Hurtado, and Nuno (2019); Villa and Valaitis (2019); Maliar, Maliar, and Winant (2021); Duarte (2018); Fernandez-Villaverde, Nunu, Sorg-Langhans, and Vogler (2020); Duarte, Fonseca, Goodman, and Parker (2021); Bretscher, Fernandez-Villaverde, and Scheidegger (2022); Han, Yang, et al. (2021). Recently, building on Maliar et al. (2021), Kase, Melosi, and Rottner (2022) estimate a HANK model by approximating the likelihood function via neural networks.

Zoph, and Le, 2017) and double descent phenomenon (see, e.g., Stephenson and Lee, 2021; Nakkiran et al., 2021) are necessary to construct high-dimensional surrogate models to estimate dynamic models in finance. Indeed, very deep models are necessary to replicate complex structural models, and *Swish* activation functions are necessary to obtain a smooth gradient when using the surrogate model on real data.

Thirdly, in the option pricing application, we study two popular structural models in the literature: the stochastic volatility model of Heston (1993) and the double-exponential jump-diffusion model extended from Bates (1996). Our method allows us to re-estimate the structural parameters and hidden states at high frequency and compare the models' out-of-sample performance at various time horizons. We apply the test statistic developed by Andersen et al. (2015) to investigate the stability of the risk-neutral dynamics of the two models. Christoffersen and Jacobs (2004) also compare the pricing performance of a "standard" Practitioner's Black-Scholes (PBS) model against the Heston model. They emphasize the importance of using the same loss function to estimate and evaluate the models, and they find that in doing so, the PBS model outperforms the Heston model out-of-sample. Our finding of the superior hedging performance of structural models echoes the finding of Schaefer and Strebulaev (2008) in the corporate bond market, which shows that structural credit models are informative about out-of-sample hedge ratios even though they might imply large pricing errors.

The remainder of this article is organized as follows. In Section 2, we present the *deep surrogate* methodology and discuss how to apply it to option pricing models. Section 3 presents the applications. In section 4, we confront our methodology in the context of real data. Section 5 concludes.

2 Methodology

In this section, we introduce the *deep surrogate* methodology, discuss the advantages of applying deep neural networks in the context of surrogate models, and summarize the procedure to create a surrogate model in algorithm 1.

2.1 Deep surrogates

Consider an economic model as represented by function f, which maps state variables into equilibrium objects such as optimal policies, equilibrium prices, or their moments. Formally, we assume

$$\mathbf{y}_t = f(\mathbf{s}_t|\theta), \tag{1}$$

where $\mathbf{s}_t \in \mathbb{R}^n$ is a vector of states (which could be observable or hidden), $\theta \in \mathbb{R}^p$ is a vector of model parameters, and $\mathbf{y}_t \in \mathbb{R}^k$ is a vector of model outputs.

By treating the model parameters θ as pseudo-states, we can define the augmented state vector $\mathbf{x}_t$ as

$$\mathbf{x}_t \equiv [\mathbf{s}_t, \theta], \tag{2}$$

and rewrite (1) as $\mathbf{y}_t = f(\mathbf{x}_t)$. We denote the admissible augmented state space by $\mathcal{X} \in \mathbb{R}^m$, where $m = n + p$.

In situations where the function $f(\cdot)$ is computationally expensive to evaluate, one may wish to build a cheap-to-evaluate surrogate, $\widehat{f}(\cdot)$, for the original function $f(\cdot)$. A familiar example of such a surrogate is the standard normal table, which stores pre-computed values for the CDF function of the standard normal distribution on a grid. We propose to use deep neural networks (DNNs) to construct the said surrogate $\hat{f}(\cdot)$, or *deep surrogate*, which takes advantage of the flexibility of DNN and can approximate high-dimensional functions with high accuracy.

Below, we briefly outline the key elements of a deep neural network. For a textbook treatment of DNNs, see Goodfellow et al. (2016). Neural networks consist of multiple stacked-up layers of neurons. A given layer ℓ takes a vector I_ℓ as input and produces a vector O_ℓ as output,

$$O_\ell = \sigma\left(W_\ell I_\ell + b_\ell\right), \tag{3}$$

where W_ℓ and b_ℓ consist of unknown parameters for the network; $\sigma(\cdot)$ is a non-linear function applied element-wise and is commonly termed an *activation function*. For a neural network that consists of L layers, I_1 represents the network's inputs; inputs for the intermediate (or "hidden") layers are the outputs from the previous layers, that is, $I_\ell = O_{\ell-1}$ for $\ell = 2, \cdots, L$; O_L represents the output of the final layer.

Popular choices for the activation function $\sigma(\cdot)$ include the sigmoid function and rectified linear unit (ReLU). In our numerical experiments, we use the *Swish* activation function (Ramachandran et al., 2017),[10] which is given by

$$\sigma(x) = \frac{x}{1 + \exp(-\gamma x)}, \tag{4}$$

[10]It has been shown empirically that the performance of very deep neural nets in combination with *Swish* activation functions outperform architectures that rely on ReLu (see, e.g., Tripathy and Bilionis (2018b)).

where γ is either a constant or a trainable parameter. The *Swish* activation function can be viewed as a smooth version of the ReLu function, and possesses two particular qualities that make it appropriate for the surrogate application: 1) unlike the *sigmoid*, it does not suffer from the vanishing gradient problem and, as our application shows, complex structural models require surrogate networks with a deep architecture, and 2) unlike the *ReLu* activation function, the *Swish* is smooth, which means that the gradients of the trained surrogate model will also be smooth across the state-space. This latter property substantially enhances the performance when the gradients of the surrogate are needed (e.g., for estimation).

Let $\phi(\mathbf{x}_t|\xi)$ be a neural network that takes the same input vector $\mathbf{x}_t$ as does $f(\cdot)$ and generates an output of dimension k, where ξ is the vector of all the trainable parameters for the network (W_ℓ and b_ℓ from (3) for all layers $\ell = 1, 2, \cdots, L$). To train the hyper-parameters ξ, we need a training set, which is a random sample composed of pairs $(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)$, where $\tilde{\mathbf{y}}_i = f(\tilde{\mathbf{x}}_i)$. The training examples are generated by evaluating the original function f at $\tilde{\mathbf{x}}_i$ randomly drawn from the input space $\mathcal{X}$. Different from typical machine learning applications, where the data-generating process is unknown, we know the true f from the economic model and thus can generate as much data as desired. This step can be costly, especially when the size of the sample required for training the neural network is large. However, as we explain below in Section 2.2, we can have significant gains when applying the surrogate model in exchange for this one-time upfront cost.

Generating the training sample. We obtain a random draw $\tilde{\mathbf{x}}_i$ from the input space $\mathcal{X}$ as follows. Assume $\mathcal{X}$ is a hypercube. For each element $x_t^{(j)}$ in $\mathbf{x}_t$, let $\underline{x}^{(j)}$ and $\bar{x}^{(j)}$ be it's lower bound and upper bound, respectively, and

$$\underline{\mathbf{x}} = [\underline{x}^{(1)}, \underline{x}^{(2)}, \cdots, \underline{x}^{(m)}], \quad \overline{\mathbf{x}} = [\bar{x}^{(1)}, \bar{x}^{(2)}, \cdots, \bar{x}^{(m)}]. \tag{5}$$

We then draw $\tilde{\mathbf{x}}_i$ according to:

$$\tilde{\mathbf{x}}_i = \underline{\mathbf{x}} + \tilde{R}_i(\bar{\mathbf{x}} - \underline{\mathbf{x}}), \tag{6}$$

where $\tilde{R}_i = \mathrm{diag}(\tilde{\mathbf{r}}_i)$, with $\tilde{\mathbf{r}}_i \in \mathbb{R}^m$ following an appropriate multivariate distribution. For simplicity, we use the multivariate uniform distribution, and denote the resulting training sample by $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$.

Surrogate training. Next, we train the neural network $\phi(\mathbf{x}_t|\xi)$ by

$$\xi^* = \arg\min_{\xi} L(\mathbf{x}, \mathbf{y}; \xi), \tag{7}$$

where the loss function is given by

$$L(\mathbf{x}, \mathbf{y}; \xi) \equiv \frac{1}{N} \sum_{i=1}^{N} \|\phi(\mathbf{x}_i|\xi) - \mathbf{y}_i\|_1, \tag{8}$$

which is the mean ℓ_1-norm of the difference between the actual model outcome and the prediction from the neural network in the training sample.

To perform the minimization, we use stochastic gradient descent. The neural network's gradients can be computed exactly through the backpropagation algorithm, which we can view as a recursive application of the standard chain rule (Chauvin and Rumelhart, 1995). In standard machine learning applications, the stochastic gradient descent is often stopped before convergence to avoid over-fitting. In contrast, we rely on double-descent (see, e.g., Stephenson and Lee, 2021; Nakkiran et al., 2021), a very recently discovered phenomenon that appears to be particularly strong in the case of *deep surrogate* thanks to two unique distinctions: (i) our target y_i is noise-free (or nearly noise-free when $f(\cdot)$ can only be evaluated numerically), and (ii) the training sample size N can be made arbitrarily large with the knowledge of the true model.

Surrogate validation. After training, we evaluate the accuracy of the deep surrogate model $\phi(\mathbf{x}_t|\xi^*)$ to see whether its accuracy is sufficiently high. The acceptance of the surrogate is based on the following convergence criterion:

$$\sup_{\mathbf{x}_t \in \mathcal{X}} d\left(\phi(\mathbf{x}_t|\xi), f(\mathbf{x}_t)\right) \leq \varepsilon, \tag{9}$$

where $d(\cdot)$ is a distance metric between the two functions, and ε is some small positive constant. To check the above condition, we generate a new random sample of size N^o, $\left(\mathbf{x}_j^o, \mathbf{y}_j^o\right)_{j=1}^{N^o}$, and evaluate a finite-sample version of (9) under the ℓ_1-norm,

$$\sup_j \|\phi(\mathbf{x}_i^o|\xi^*) - \mathbf{y}_i^o\|_1 \leq \varepsilon. \tag{10}$$

If Condition (10) is satisfied, $\phi(\mathbf{x}_t|\xi^*)$ is accepted as a deep surrogate of the model $f(\cdot)$. Otherwise, we can add flexibility to $\phi(\cdot)$ by modifying, for instance, its architecture, and increase the size of the training sample to further reduce training error. We then re-validate the model. The two steps are iterated until Condition (10) is met.

Algorithm 1 Training a deep surrogate

```
Input:
  Some economic model $\mathbf{y}_t = f(\mathbf{s}_t \mid \theta)$, $\mathbf{s}_t \in \mathbb{R}^n$, $\theta \in \mathbb{R}^p$, $\mathbf{y}_t \in \mathbb{R}^k$.
  A neural network $\phi(\mathbf{x}_t \mid \xi) : \mathbb{R}^{n+p} \to \mathbb{R}^k$, where $\mathbf{x}_t \equiv [\mathbf{s}_t, \theta]$.
  Upper and lower bound of the pseudo-state-space in which the surrogate is define: $\underline{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R}^{n+p}$.

Output:
  The neural network's trained parameters $\xi*$, such that $\phi(x_i, \xi^*)$ satisfies Eq. (10).
  ———————— Part 1
  Generating the training samples
  for $i \leftarrow 1$ to $N$ do          # Embarrassingly parallelizable
    Draw $\bar{R}_i \sim U(0,1)$
    $\tilde{\mathbf{x}}_i = \underline{\mathbf{x}} + \bar{R}_i(\overline{\mathbf{x}} - \underline{\mathbf{x}})$
    $\mathbf{y}_i = f(\tilde{\mathbf{x}}_i \mid \theta)$
  end for
  ———————— Part 2
  Surrogate training
  $\xi^* = \arg\min_{\xi} \frac{1}{N} \sum_{i=1}^{N} \|\phi(\mathbf{x}_i \mid \xi) - \mathbf{y}_i\|_1$

  Surrogate validation
  for $i \leftarrow 1$ to $N^o$ do
    $\tilde{\mathbf{x}}_i^o = \underline{\mathbf{x}} + \bar{R}_i(\overline{\mathbf{x}} - \underline{\mathbf{x}})$
    $\mathbf{y}_i^o = f(\tilde{\mathbf{x}}_i^o \mid \theta)$
  end for

  if $\sup_{\mathbf{x}_t \in \mathcal{X}} d(\phi(\mathbf{x}_t \mid \xi), f(\mathbf{x}_t)) \leq \varepsilon$ then
    Validate surrogate
  end if
```

2.2 Benefits of the deep surrogate

Having explained the algorithm for building a deep surrogate model, we now turn to the advantages a deep surrogate offers relative to conventional methods of model evaluation, including surrogate models based on alternative approximators.

The deep surrogate acts as a (high-dimensional) lookup table, effectively "interpolating" the precalculated mappings between $\mathbf{x}_t$ and $\mathbf{y}_t$ in the training sample using a deep neural network. Once trained, it is easy to see the gain in performance for evaluating the surrogate $\phi(\mathbf{x}_t|\xi^*)$ relative to $f(\mathbf{x}_t)$. Moreover, the evaluation of a deep surrogate is highly parallelizable. Computationally, it can be reduced to a succession of matrix multiplications and the application of activation functions. Therefore, it is well suited to take advantage of modern hardware such as GPUs. Finally, thanks to the backward propagation algorithm, we can get

Table 1: Performance Improvement from Deep Surrogates

This table compares the estimated computing time for pricing or getting the gradient of all the SPX options on an average day and daily estimation over a year using a traditional method based on Fast Fourier Transform (FFT) versus a Deep Surrogate model. We use the Bates model for illustration. We estimate the gradient of the FFT method through numerical differentiation, while the gradient of the Deep Surrogate can be obtained directly through backward propagation. To transform the daily gradient estimation into a yearly estimation of the model's parameters, we assume an average of 10 iterations of the optimization algorithm and 252 business days in a year.

	FFT	Deep Surrogate	Deep Surrogate + GPU
pricing, 1-day	10s	0.6s	0.06s
gradient, 1-day	180s	3.2s	0.3s
estimation, 1-year	125h	2.2h	0.2h

the gradient of a surrogate model for little to no computational costs. This can be particularly useful for model estimation (see Section 2.3) and sensitivity analysis (see, e.g., Chen, Dou, and Kogan, 2022).

To illustrate the magnitude of gains in computing speed, we compare the approximate computing time for pricing 4,000 European options (roughly the daily number of SPX options traded on the CBOE) with the Bates model based on three methods: direct evaluation of the pricing formula using the Fast Fourier Transform (FFT), a deep surrogate using a single CPU, and a deep surrogate using a GPU. We also compare the computing time for estimating the model parameters from option prices on a single day, as well as the time required to perform daily estimation for an entire year. The results are shown in Table 1. Although the estimates can vary depending on hardware and implementation, the advantage of the deep surrogate relative to ordinary FFT, which suffers from the curse of dimensionality, is evident, especially when we take advantage of its parallelizability on a GPU. In summary, if we assume that a *true* f can be evaluated at arbitrary points in $\mathcal{X}$, and it is *expensive* to evaluate, which means that the explicit numerical evaluation of f might take much time, then replacing f by a surrogate that approximates f can lead to substantial performance gains.

Upfront vs. recurring cost. The comparison provided in Table 1 is not entirely fair since it does not take into account the time needed to train the surrogate. In exchange for faster model evaluation, the deep surrogate requires a large upfront cost for generating the training sample and fitting the DNN. However, this tradeoff is worthwhile for several reasons.

First, while the one-time upfront cost for building the surrogate can be large, it leads to

Table 2: A Comparison of Approximation Methods

This table provides a comparison of different approximation methods. "High-dimensional input" refers to problems with state space dimensions higher than four. "Capturing local features" refers to the ability to deal with strong nonlinearity, such as kinks. "Irregularly-shaped domain" refers to problems with geometries other than a hypercube. "Large amount of data" refers to the ability to deal with more than $10,000$ observations.

	Polynomials	Splines	(Adaptive) sparse grids	Gaussian processes	Deep neural networks
High-dimensional input	✓	✗	✓	✓	✓
Capturing local features	✗	✓	✓	✓	✓
Irregularly-shaped domain	✓	✗	✗	✓	✓
Large amount of data	✓	✓	✓	✗	✓

a massive improvement in computing speed in all future model evaluations. Thus, in cases where a large number of model evaluations is needed (e.g., when pricing a large panel of options or doing parameter estimation), the benefits of a deep surrogate post-training can easily outweigh the upfront costs.

Second, deep surrogate models are highly portable. One only needs to store the parameters for the neural network, which is substantially smaller than the amount of data storage needed when using a grid-based method. This feature makes it easy to share a deep surrogate among users, which further helps to justify the upfront cost.

Third, the main upfront cost comes from generating the training and validation samples. This process is ideally suitable for parallelization. Although the overall computational cost can be high in node hours, it can be distributed across the compute nodes of contemporary high-performance computing hardware, thus reducing the runtime by orders of magnitude.[11]

Why DNNs? One can build a surrogate model using a variety of function approximators, such as Chebyshev polynomials or splines. Deep neural networks have several desirable properties which make them ideal for our purposes. Following Azinovic et al. (2022), Table 2 offers a comparison of DNN and other standard approximation methods.

[11]For illustration, consider estimating the Bates model. When using the true pricing function, the estimation takes roughly 40 minutes on a single compute node of the "Piz Daint," a Cray XC50 system at the Swiss National Supercomputing Centre (its compute nodes are equipped with a 12-core Intel(R) Xeon(R) E5-2670 with 64GB of DDR3 memory and an NVIDIA Tesla P100 GPU with 16GB.). On the same hardware, we are able to reduce the computing time to less than one second with the *deep surrogate*. The computing time could be further reduced with expert knowledge and appropriate code optimization.

Neural networks are universal function approximators (see, e.g., Hornik, Stinchcombe, and White, 1989). They can handle irregularly shaped domains and approximate functions with kinks and ridges.[12] In contrast, polynomials, if coupled with Smolyak's algorithm (see, e.g., Krueger and Kubler, 2006; Judd, Maliar, Maliar, and Valero, 2014), are well suited for approximating high-dimensional smooth functions on irregularly shaped domains, but have difficulties resolving kinked features. Adaptive sparse grids can approximate functions with local features using a large amount of high-dimensional data but are inefficient on irregular domains (see, e.g., Brumm and Scheidegger, 2017). Gaussian processes can approximate high-dimensional functions with local features on irregularly shaped domains, but become prohibitively slow when using many training points (see, e.g., Tripathy et al., 2016; Scheidegger and Bilionis, 2019). In addition, DNNs can alleviate the curse of dimensionality, as was shown both formally (see, e.g., Montanelli and Du, 2019), as well as in a broad range of empirical applications. Consequently, one can typically train a deep network to accurately approximate the high-dimensional function we are concerned with using substantially fewer observations than one would use with traditional methods based on Cartesian grids. Furthermore, a key advantage of DNNs is their ability to handle high-dimensional input. A densely connected feed-forward neural network is a sequence of matrix-vector multiplications, vector-vector additions, and element-wise applications of the activation functions. Hence, the computational complexity of evaluating the neural network is inherited from these operations. Furthermore, the computational complexity of computing the gradients with automatic differentiation is of a similar magnitude to that of a single evaluation of the neural network. Therefore, neural networks can handle very high-dimensional inputs in both training and inference without becoming prohibitively slow.

2.3 Deep surrogates for estimation

An important application of a deep surrogate model is in structural estimation. Due to the significant gain in speed for evaluating the model and its gradients, the application of popular estimators such as GMM or MLE can also become significantly more efficient, thereby rendering the estimation of complex models numerically tractable.

For example, suppose we want to estimate the economic model parameters θ using GMM.

[12]To accurately approximate local features, such as kinks, it is essential that the approximating function can fit strong nonlinearities in some areas of the domain without deteriorating the approximation quality in others. While some grid-based methods, like adaptive sparse grids (see, e.g., Brumm, Krause, Schaab, and Scheidegger (2022), and references therein) or linear interpolation from a dense grid (see Judd, 1996, for a survey), can resolve local features well, it is not the case for polynomial interpolation (approximating a kinked function by interpolating points with global polynomials often leads to undesired "wiggles", known as Runge's phenomenon, which also occur away from the kink).

Eq. (1) in the economic model implies the following moment condition,

$$E[f(\mathbf{s}_t|\theta) - \mathbf{y}_t] = 0. \tag{11}$$

Assume $k > p$ and the system of moment restrictions is over-identified. With a sample of T observations, we define

$$g_T(\theta) \equiv \frac{1}{T}\sum_{t=1}^{T}(\phi(\mathbf{x}_t(\theta)) - \mathbf{y}_t), \tag{12}$$

where $x_t(\theta) = [\mathbf{s}_t, \theta]$. With a $k \times k$ weighting matrix W, the GMM estimator reads

$$\hat{\theta} = \arg\min_{\theta} g_T(\theta)^T W g_T(\theta). \tag{13}$$

Furthermore, the first-order condition is given by

$$d_T(\theta)^T W g_T(\theta) = 0, \tag{14}$$

where

$$d_T(\theta) \equiv \frac{\partial g_T(\theta)}{\partial \theta^T} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial \phi(\mathbf{x}_t(\theta))}{\partial \theta^T}. \tag{15}$$

Notice that both $\phi(\mathbf{x}_t)$ and its gradient are available analytically through the deep surrogate model. Thus, Eq. (14) is a system of p non-linear equations, which can be solved efficiently. This is in contrast to having to estimate d_T through numerical differentiation schemes, which further adds to the large number of costly model evaluations entailed in a typical parameter search.[13] Afterward, the asymptotic variance of the estimator is readily available as well.

In the remainder of the paper, we will perform a thorough out-of-sample analysis of the Bates model, using either the HM or the RF as a benchmark. This analysis demonstrates

[13]Numerical differentiation requires multiple estimations of the base model. As an example, take the simple first-order finite difference scheme $(f(x+h) - f(x))/h_i$ where $f(x)$ is the original model as a function of a single pseudo-state x, and h_i is a small positive number. To approximate the Jacobian numerically, one has to apply this or a similar (computationally potentially even more expensive) discretization once per pseudo-state. Thus, estimating the Jacobian with finite differences comes at the cost of at least two numerically expensive function evaluations $f(\cdot)$ per dimension of θ. In contrast, the backward propagation algorithm we use to compute the gradients of a deep surrogate model does not query the original model and needs to be estimated once to obtain the gradient of all the states and pseudo-states. Consequently, as long as the original model is costly enough, the optimization scheme presented here will be significantly faster with a surrogate model. Table 1 demonstrates that, in our application, this condition is well met.

how t the deep surrogate methodology opens the door for model analysis that would have been computationally infeasible otherwise.

3 Application to Option Pricing

In this section, we apply the *deep surrogate* technology to option pricing. We choose this field of applications because it has well-developed numerical solutions to benchmark performance and since requires frequent estimation and out-of-sample analysis. We use a leading option pricing models to demonstrate how to build surrogate models and how to use them for structural estimation and other model analysis.

3.1 The Bates Model

The example we choose to illustrate the deep surrogate methodology is an extended version of the model of Bates (1996), which features stochastic volatility and jumps. The log jump size is modeled as double-exponential. While the deep surrogate methodology can be readily applied to more sophisticated option pricing models (for example, Duffie, Pan, and Singleton, 2000; Bates, 2000; Pan, 2002; Andersen et al., 2015), we choose this example for a good balance between transparency and complexity (with a total of 14 augmented states).

Under the risk-neutral measure $\mathbb{Q}$, the stock price S_t and the conditional variance v_t are assumed to follow the process:

$$\frac{dS_t}{S_{t^-}} = (r - d - \lambda\bar{\nu})dt + \sqrt{v_t}dW_1 + dZ_t, \tag{16a}$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v}dW_2, \tag{16b}$$

Here, r is the (constant) instantaneous risk-free rate, and d is the dividend yield. The conditional variance v_t follows a Feller process under $\mathbb{Q}$, with the speed of mean reversion κ, long-run mean θ, and volatility parameter σ. The two standard Brownian motions under $\mathbb{Q}$, $W_{1,t}$ and $W_{2,t}$, are correlated with $E[dW_{1,t}dW_{2,t}] = \rho dt$. Finally, Z is a pure jump process with constant arrival intensity λ. The log jump size has distribution ω with average jump size $\bar{\nu}$. Different from Bates (1996), who assumes ω is normal, we model the log jump size with an asymmetric double exponential distribution,

$$\omega(J) = p\frac{1}{\nu_u}e^{-\frac{1}{\nu_u}J}1_{\{J>0\}} + (1-p)\frac{1}{\nu_d}e^{\frac{1}{\nu_d}J}1_{\{J<0\}}, \quad \text{with} \;\; p \geq 0, \tag{17}$$

where $0 < \nu_u, \nu_d < 1$ are the average log jump size for positive and negative jumps, respectively, and p is the probability of a positive jump conditional on having a jump. Under this specification, the average jump size is $\bar{\nu} = \frac{p}{1-\nu_u} + \frac{1-p}{1+\nu_d} - 1$. The double-exponential assumption allows for heavier tails in jumps than the normal distribution, a feature supported by the evidence in Bollerslev and Todorov (2011).

The HM (Heston, 1993), which we use as a benchmark in some applications, can be viewed as the special case where $\lambda = 0$—that is, a model with time-varying volatility but no jumps.

A European option with maturity T and strike price K can be priced as the expected payoff under $\mathbb{Q}$ discounted using the risk-free rate. For example, the price of a European put option at time t is

$$P(S_t, t, v_t) \equiv E^{\mathbb{Q}}[e^{-r(T-t)}(K - S_T)^+].$$

The solution is based on the Fourier inversion of the characteristic function for affine processes.[14] Thanks to the put-call parity, it suffices to focus on the pricing of put options.

To facilitate a comparison of the pricing errors for options with different moneyness and maturities, we map option prices to the Black-Scholes implied volatility (BSIV). Moreover, following Andersen, Fusari, and Todorov (2017), we define a normalized moneyness measure m_t as

$$m_t = \frac{\ln\left(\frac{K}{S_t F_{t,T}}\right)}{\sqrt{T-t}\sigma_{atm}}, \tag{18}$$

where $F_{t,T}$ is the forward price of the stock for the same maturity as that of the option, while σ_{atm} is a constant volatility measure, which we set to be the average BSIV of at-the-money options from the entire sample.

When mapping the Bates model into our deep surrogate framework, there are three state variables, $\mathbf{s}_t = [m_t, T-t, v_t]$, where v_t is latent, and ten parameters $\theta = [r, d, \kappa, \theta, \sigma, \rho, \lambda, \nu_u, \nu_d, p]$.

3.2 Deep surrogates for option pricing

We now go over the details for generating the training sample, network architecture design, and the surrogate training procedure we use for the option pricing models.

[14]We apply the state-of-the-art option pricing library QuantLib (`https://www.quantlib.org`) to compute prices and the corresponding BSIV.

Training sample and DNN architecture. To generate the training sample, we first obtain random draws $\tilde{\mathbf{x}}_i$ from the input space $\mathcal{X}$. To do so, we set the minimum and maximum values for each state variable and all parameters (see Table A.1 in Appendix A), where the values for these upper and lower bounds are chosen based on a combination of model restrictions and domain knowledge. For example, the intensity of the Poisson process λ in the Bates model is, by definition, positive; the correlation between volatility shocks and price shocks ρ is restricted to be negative based on the so-called leverage effect documented in the literature.

We build the training samples and surrogate models following an iterative process described in Algorithm 1, increasing the training sample size and depth of the deep surrogate gradually until the validation criterion based on mean absolute pricing error (cf. equation (7)) is met. At each step, we run a mini-batch stochastic gradient descent algorithm to determine the neural network's parameters.[15] With this procedure, we obtain a training sample of size 10^9. The final feedforward network for the Bates model for instance has 7 hidden layers, 400 neurons each. This architecture yields a total of 967,201 trainable parameters.

Note that a small validated pricing error is a necessary but not sufficient condition to consider the DNN good enough to be used as a surrogate for the option pricing model. It only shows that the surrogate can price options accurately, but does not imply that it can approximate the function gradients sufficiently accurately so that we can derive reliable greeks or use the surrogate to accurately estimate model parameters. To verify these properties, we need controlled test cases based on simulation, which we discuss in Section 3.3.

Despite the seemingly large sample size, notice that the training sample only sparsely populates the 14-dimensional augmented state space.[16] However, thanks to the smoothness of the pricing function in the Bates model, the deep surrogate still manages to accurately approximate the true model. It is also worth noting that we build the training sample using a naive scheme by randomly drawing points from the augmented state space based on the uniform distribution. An adaptive scheme (see, e.g. Krause and Guestrin, 2007; Deisenroth, Rasmussen, and Peters, 2009; Renner and Scheidegger, 2018), that is, one that samples more intensively in regions where nonlinearity is more pronounced and/or where initial approximation errors are large) is likely to be much more efficient. We leave this question for

[15]Specifically, we run the ADAM optimization algorithm with an initial learning rate of $0.5 * 10^{-4}$ for 15 epochs, that is, we use mini-batches of size 256 until we have used the whole data set 15 times. After each epoch, we save the model and use a validation set of 10,000 points to estimate the surrogate model's performance. In the end, we use the network's parameters after the epoch with the lowest validation error.

[16]Consider a naive discretization scheme, where one places n points along each axis. The naive generalization to d dimensions would yield n^d points. Consequently, under the traditional Cartesian grid-based methods (see, e.g., Press, Teukolsky, Vetterling, and Flannery (2007)), those $N = 10^9$ points would translate into $N \approx 4$, clearly a low-resolution representation of the function to be approximated.

future research.

A deep network structure is critical to approximate the true model with sufficient accuracy, and it is intuitive that the depth of the network should increase further as the complexity of the model increases (such as the addition of states and model parameters) in order to keep the surrogate performance at a constant level. We examine this point in Section 3.3.

3.3 Accuracy of the surrogates

In this section, we examine the accuracy of the deep surrogate models. We start by investigating how accurately the surrogate model captures the sensitivity of option value to various states and parameters, that is, the pseudo-states. Such sensitivities are critical for hedging as well as parameter estimation. For illustration, we vary the states or parameters one at a time while setting the rest at their respective averages and then compare the BSIVs or their partial derivatives produced by the target model versus those by the surrogate of the Bates model. The results are shown in Figure A.1 and Figure A.2 in Appendix A. These figures highlight the complexity of the Bates model – there is significant nonlinearity along multiple dimensions. In particular, even though we do not explicitly target the partial derivatives for option prices, the surrogate almost perfectly replicates these features.

Next, using simulated data, we systematically examine the ability of the surrogate technology to recover parameter values in structural estimations. The procedure is as follows. First, we randomly draw volatility v_t and parameters θ according to Eq. (6). Second, we compute the option prices implied by the Bates model on 1,000 options, the moneyness, and maturity of which are drawn from a uniform distribution. These option prices are then converted into BSIVs. Third, we use the deep surrogate function $\phi(\cdot)$ to run GMM estimation of the hidden state v_t and parameters θ from the cross-section of option data (see Section 2.3). For each option, x_t depends on the hidden state v_t and parameters θ, while the BSIVs form our $\mathbf{y}_t$. Under the identity weighting matrix, the objective function minimizes squared pricing errors:

$$\hat{v}_t, \hat{\theta} = \arg\min_{v,\theta} (\phi(\mathbf{x}(v,\theta)) - \mathbf{y}_t)^T (\phi(\mathbf{x}(v,\theta)) - \mathbf{y}_t). \tag{19}$$

At the end of the estimation, we compute the estimation errors for each element of θ,

$$e_i = \frac{|\theta_i - \hat{\theta}_i|}{\bar{\theta}_i - \underline{\theta}_i}, \tag{20}$$

where $\bar{\theta}_i$ and $\underline{\theta}_i$ represent the maximum and minimum values in the training sample of the respective surrogate model (cf. Table A.1). The same error measure applies to the estimate

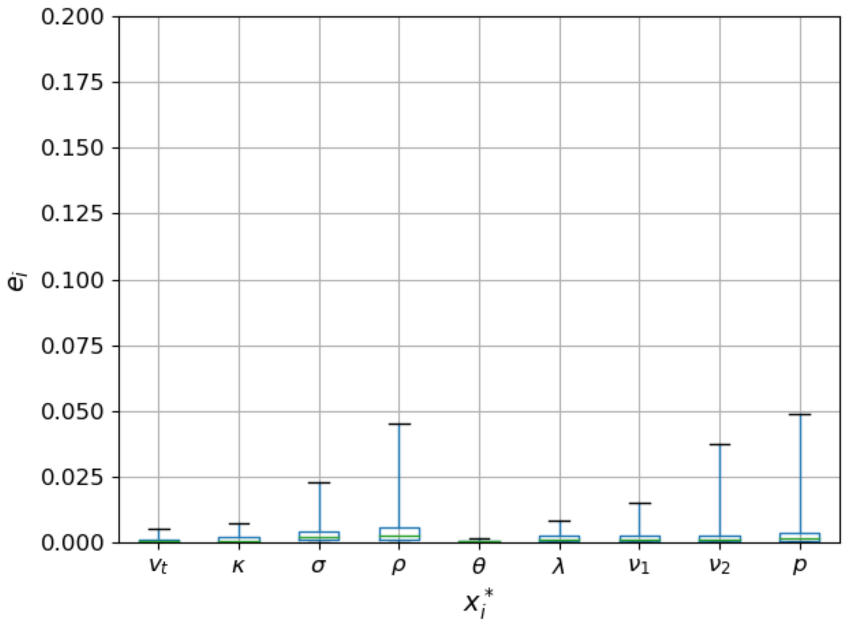


Figure 1: The figures above show that the Bates surrogate can be used to estimate the parameters of the models on a small data sample. We simulated market days with the original pricing models and used our surrogates to estimate the parameters and states simultaneously. Above, we show the resulting prediction errors (see Eq. (20)) for each hidden state and parameter of both models. The green vertical line represents the median prediction error computed across 1000 simulations. The box shows the interquartile range, while the whiskers show the 1st and 99th percentile errors across the simulations.

of the conditional variance, v_t.

We repeat the above procedure 1,000 times and present the distribution of estimation errors for the hidden state and different parameters in Figure 1.[17] Overall, the estimation errors are quite small. The parameters that are relatively more difficult to estimate include ρ, the correlation between shocks to stock prices and volatility, ν_2, the average size of negative jumps, and p, the probability of a positive jump conditional on a jump occurring. However, the estimation errors are small even for these parameters. For example, the estimation error of the parameter ρ is lower or equal to 0.049 in 99% of our simulations.

Put together, the above results on model sensitivity and parameter estimation suggest that the *deep surrogate* can reliably replace the target Bates model when it comes to pricing, hedging, or parameter estimation. However, unlike the original model, querying the surrogate, or estimating the surrogate's gradients comes at a negligible computational cost.

We conclude this section by demonstrating the importance of the neural network's

[17]In the boxplot, the (green) horizontal line represents the median error. The whisker indicates the 1st and 99th percentile, whereas the box shows the inter-quartile range across all simulations.

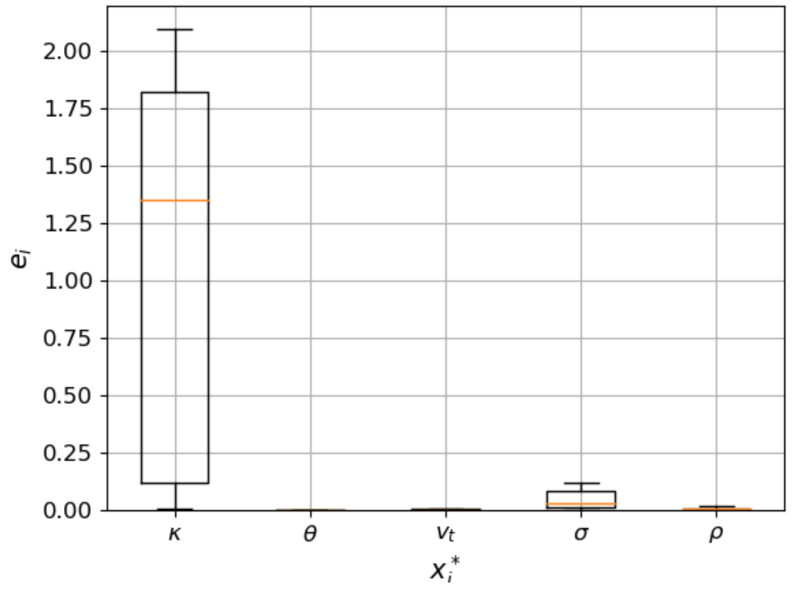


Figure 2: The figure above shows the prediction error for the call model of the HM using a shallow network (1 layer of 400 neurons).

.

architecture, in particular, its depth, in the surrogates models' performances. To do so, we first investigate the performance of shallow networks as surrogate models. In particular, we define a neural network with 400 neurons and train it to be a surrogate for the HM call options. We apply the procedure described at the start of this section to estimate the estimation error and how well the shadow network sensitivities replicate the true model sensitivities. We measure the sensitivities of a surrogate model generated by a shallow neural network, as previously done (cf. Figure A.2). We use our shallow surrogate model to compute estimation errors and produce a boxplot of those errors. Figure 2 highlights the importance of depth to build a surrogate for complex option models. Indeed, even with the HM, which is significantly simpler than the Bates, the surrogate model can not estimate all the parameters correctly in our simulation. Our vast numerical experiments showed that four 400-neuron hidden layers are necessary to build a surrogate for the HM, and seven hidden 400-neuron-layers are required to build a surrogate of good quality for the Bates.

These additional numerical results complete the tests for our method within a controlled environment and highlight the importance of choosing an appropriate network architecture to create efficient surrogate models. Note that, even though these *bad* surrogates can not estimate the models' parameters reasonably, the prediction error of these surrogates on a validation set was still virtually 0, that is, the surrogate BSIV versus the models BSIV for a given set of parameters. With complex structural financial models, some parameters have a relatively small impact on the target state. For example, on a cross-section of options on a given day, the κ parameter does not have a significant impact on the options BSIV. Therefore, we can reach a low prediction error while almost ignoring the effect of the κ. An appropriate network architecture can ensure that an adequate degree of precision is reached, even for marginally important parameters. Furthermore, this interesting result demonstrates that the prediction error alone is not a sufficiently accurate measure to assess the quality of a surrogate.

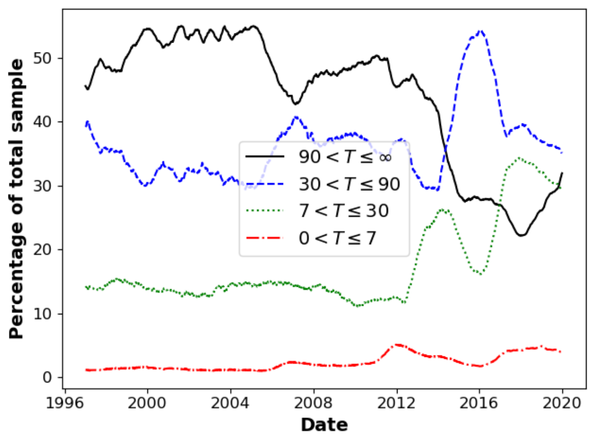
4 Empirical Results

In this section, we use the deep surrogate technology to study the structural option pricing model introduced in Section 3.1 in real data. We conduct three sets of analyses, all of which take advantage of the significant gains in the efficiency of model evaluation provided by the surrogate model, First, we conduct an in-depth analysis of the out-of-sample performance of the Bates model and compare it against a reduced-form model built on random forests. This comparison helps reveal the strengths and weaknesses of structural option pricing models relative to reduced-form models. Second, we build a measure of parameter instability for the Bates model. Empirically, this measure is positively linked to options market liquidity. Third, we characterize the model-implied option return distributions and confront them with the data. These return distributions are important for computing risk metrics (e.g., value-at-risk) and provide new dimensions to test our models.

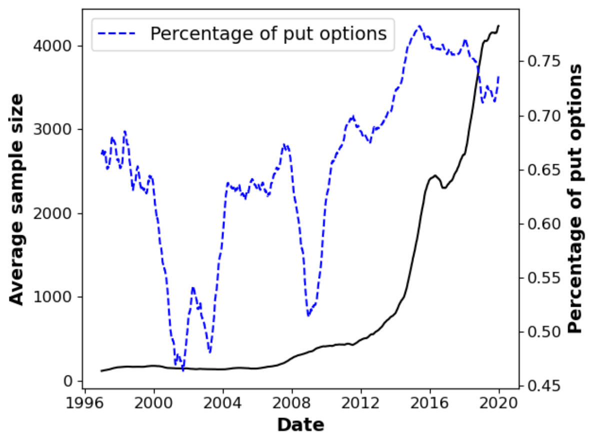
Data We use daily data on the S&P500 index options (SPX) from OptionMetric. Our sample includes European call and put options between 2001 and 2019. Following the literature (see e.g., Andersen et al., 2017), we remove options with maturity over 250 days as well as options with extreme moneyness (those with Black-Scholes delta outside the range of $[0.1, 0.9]$) since these options tend to be thinly traded.

Figure 3 summarizes our sample. Panel A shows the evolution across time for the sample's composition per maturity brackets. Panel B shows a dramatic increase in the number of SPX options traded in the past decade (left axis). Between 45% and 78% of and the percentage of put options in the sample (right axis). We smooth all numbers with a rolling average over the last 252 days.

In addition, we obtain the volatility index VIX, the tail index of estimate of the risk of



(a) Maturities in sample



(b) Number of options

Figure 3: The two figures above show the evolution of the sample composition across time. Panel (a) shows the percentage of options in the sample across time per maturity brackets. Panel (b) shows the total number of options per day (left axis) and the percentage of put options in the sample (right axis). Through all plots, we smooth the results with a rolling average over the last 252 business days.

jumps in the S&P500 index, from the Todorov and Andersen website.[18] The variable $JUMP_t$ represents the weekly probability of a negative jump of at least 10% for the S&P500 index. Finally, we obtain the risk-free rate from the Fama and French data library.[19]

4.1 Out-of-sample performance

In this section, we leverage the cheap-to-evaluate surrogate to re-estimate the Bates model daily and examine its in- and out-of-sample pricing performances. To measure out-of-sample pricing errors, we estimate the Bates model's parameters and hidden state at some given time t and use these states and parameters to predict options' BSIVs at time $t + \tau$, where τ is the forecasting horizon defined in numbers of business days. We make the predictions assuming we can see the observable states of time $t + \tau$, that is, the options' maturity, the options' moneyness, and the risk-free rate. As such, the performance out-of-sample can be viewed as a measure of parameter stability.

Non-parametric benchmark. When analyzing the empirical performance of the structural Bates model, we consider a non-parametric benchmark in the form of a random forest

[18] https://tailindex.com/volatilitymethodology.html
[19] `https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`

regressor.[20] We choose the random forest over other non-parametric benchmarks for two main reasons: (i) random forest regressors require little to no tuning to perform well out-of-sample, and (ii) random forest regressors have a relatively short training time, which significantly facilitates our analysis.

We define the RF model's input, i.e., the vector of observable states $X_i^{(R)}$ the random forest uses to predict the BSIV of option i, as

$$X_i^{(R)} = [1, \mathbb{1}_{C,i}, \hat{K}_i, T, \hat{K}_i T, \mathbb{1}_{\hat{K}>100}], \tag{21}$$

where 1 is a constant term, $\mathbb{1}_{C,i}$ is a binary indicator equal to 1 if the contract is a call option, $\hat{K}_i$ is the moneyness measure, T_i is the option's days to maturity, and $\mathbb{1}_{\hat{K}>100}$ is a binary indicator equal to 1 if the option's strike is above the underlying's asset price. The last two dimensions of the vector $X_i^{(rf)}$ provide redundant information, which we include to speed up the algorithm training.

Let $R(\cdot)$ be a random forest predictor. We train a random forest to minimize the following loss function:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(R(X_i^{(R)}) - \hat{y}_i)\right)^2, \tag{22}$$

where N is the number of options in the training sample, and $\hat{y}_i$ is the option's BSIV.

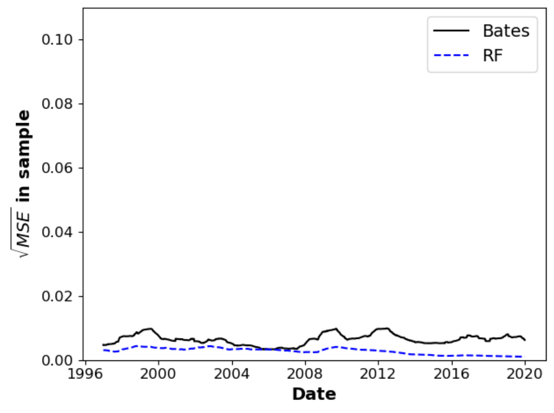
To make the comparison fair, the random forest minimizes the mean squared error, much like the surrogate models discussed above. In addition, our non-parametric benchmark does not use any additional information other than that used to estimate the option pricing model's hidden states and parameters.

We first examine the in-sample performance across time. In Figure 4(a), we display the daily $\sqrt{MSE}$ on the y-axis, smoothed over a rolling average of the previous 252 days.[21] Note that the Bates model's in-sample MSE varies across time, with peaks in 1998 and during the 2008 crisis. We also observe that both parametric models' in-sample errors are larger after the 2008 crisis. Finally, we note that the non-parametric model strongly outperforms both structural models in-sample.

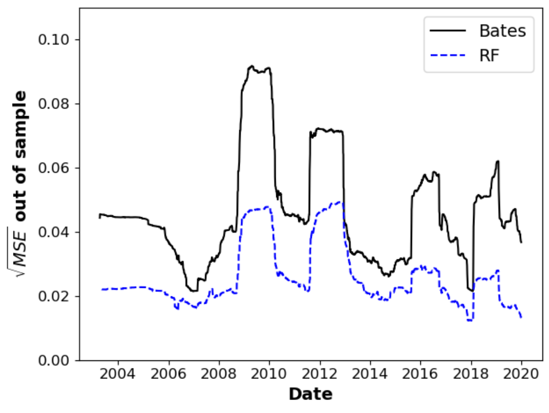
Panels (b), (c), and (d) in Figure 4 explore the out-of-sample performance. Each panel shows the average daily $\sqrt{MSE}$ estimated on a different subsample defined by maturity. In

[20]A random forest works by constructing a multitude of decision trees, with the algorithm's prediction defined as the mean output of each individual tree. This ensemble approach drastically reduces the risk of over-fitting. For a general introduction to random forests, see Liaw, Wiener, et al. (2002), and for random forests applied to finance, see Gu, Kelly, and Xiu (2018).

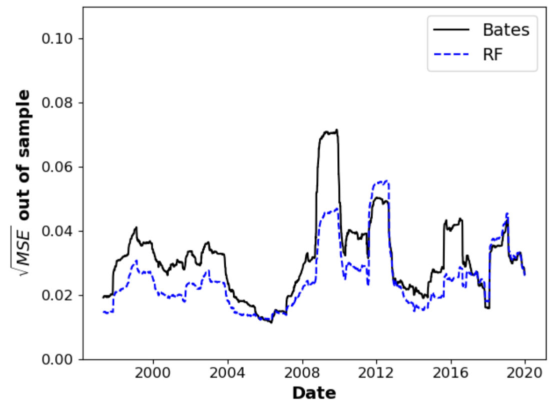
[21]To facilitate the interpretation of Figure 4(a), we additionally display the square root of the MSE here as well as throughout the remainder of this section.



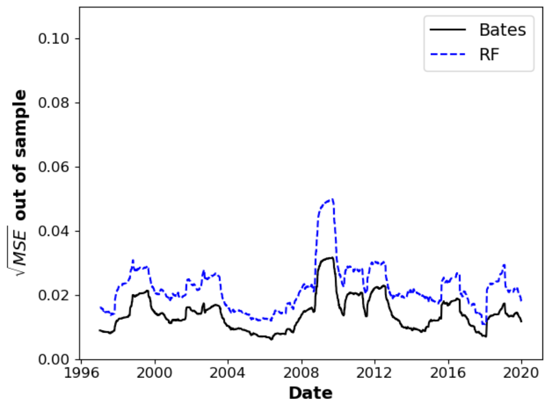
(a) IS: all T



(b) OOS: $0 < T \leq 7$



(c) OOS: $7 < T \leq 30$



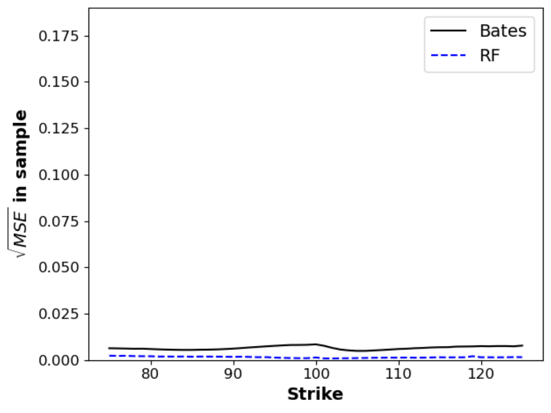
(d) OOS: $30 < T \leq 90$

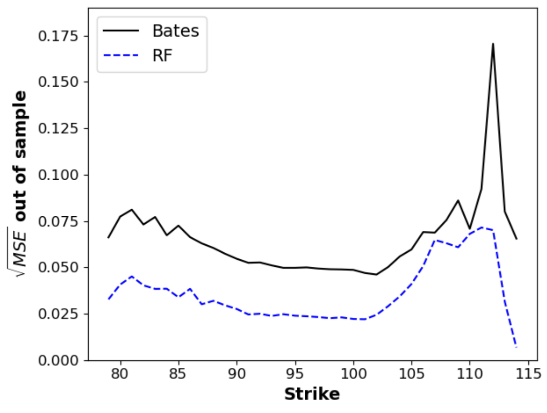
Figure 4: The figures above show the model's in- and out-of-sample performance across time in subsamples defined by maturity brackets. All panels show the performance for a forecasting horizon of one week ($\tau = 5$).

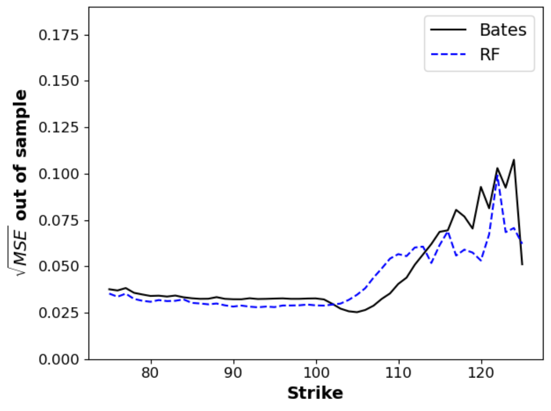
all panels, the out-of-sample performance is measured with a forecasting horizon of one week ($\tau = 5$).

All four panels reveal several interesting patterns. First, we note that, while the non-parametric is outperformed on average, the RF does outperform the Bates model on most days for options with short maturities ($0 < T \leq 7$ and $7 < T \leq 30$). Second, the average out-of-sample errors of all models on options with a long time to maturity are larger after the 2008 crisis.

We propose two non-exclusive hypotheses to explain the relatively poor performance of parametric models in Figure 4(b) and Figure 4(c). First, options with small maturities are often less liquid than their counterparts, and while liquidity can be an important pricing factor, it is assumed insignificant by parametric models. Second, most of the academic


(a) IS: all T


(b) OOS: $0 < T \leq 7$


(c) OOS: $7 < T \leq 30$

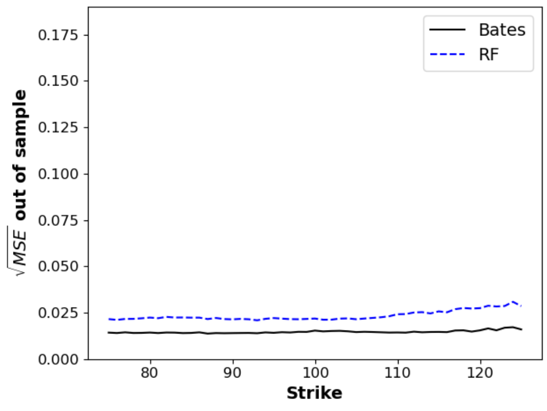

(d) OOS: $30 < T \leq 90$

Figure 5: The figures above show the model's in- and out-of-sample performance across moneyness on subsamples defined by maturity brackets. All panels show the performance for a forecasting horizon of one week ($\tau = 5$)

literature removes short maturity options from samples when testing and discussing models. To do so, academics argue that the illiquidity makes these option prices unreliable (see, e.g., Andersen et al., 2015, 2017; Kadan and Tang, 2020). However, the relatively good performance of our RF shows that there is some reliable pattern in the implied volatility smile.

Figure 5 further explores the distribution of the out-of-sample performance by showing the average $\sqrt{MSE}$ per standardized strike. As in Figure 4, we split the sample in each panel into maturity brackets. fig. 5(b) and Figure 5(d) show that for some maturity ranges, all models tend to underperform significantly more with deep-out-of-the-money call options than the rest of the sample.[22] In Figure 5(c), we see that the parametric models significantly

[22]Recall that the sample does not contain in-the-money options. Therefore, the left-hand side of the graphs

outperform the RF for options with a maturity contained between one and three months. Finally, when looking at Figure 5(a) and Figure 5(b), the relative overperformance of the RF is stable across moneyness on options with a very short time to maturity but the relative overperformance is concentrated on put options for options with maturities between one week and one month.

In Appendix D, we complement the analysis discussed in this section with additional figures showing the models' performance on other data subsamples.

4.1.1 The explaining factors of the out-of-sample performance

We now investigate what drives the differences in performance among the option pricing models. To do so, we compute the difference between the Bates model and the non-parametric benchmark, $(\sqrt{MSE_{Bates}} - \sqrt{MSE_{RF}})$. In addition, we look at the difference in performance at the individual option level. For simplicity, we focus on a forecasting horizon of a week $(\tau = 5)$.

We investigate potential factors that could explain the relative daily performance among the models. To do so, we compute the mean daily bid-ask spread at time t $((b-a)_t)$ as a proxy of liquidity risk. We capture the relative volatility risk with the VIX index VIX_t. We test the risk of jumps in the underlying asset's price with the variable $JUMP_t$. Finally, because the jump and volatility risk are highly correlated (0.84), we construct a decorrelated version of the volatility risk. We define $V\tilde{I}X_t$ as the residual from the regression $VIX_t = a + b \cdot JUMP_t + \epsilon_t$.

The coefficient associated with the risk of jumps is negative, meaning that the Bates outperforms significantly more on days with large risks of jumps. This effect is to be expected, given that the Bates model was explicitly designed to model such risks. Similarly, the Bates model overperforms on days with high volatility risk.

Table 6 shows the same regressions when we analyze the difference in performance between the Bates model and RF $(y_t = 100(MSE_{Bates,i,t} - MSE_{RF,i,t}))$. When controlling for the yearly fixed effect, the coefficient associated with liquidity risks is positive and statistically significant. These results suggest that the RF relative performance is higher when the liquidity risk is high. This mechanism is coherent because, unlike the RF, the parametric Bates model was specifically designed under the assumption that liquidity had no impact on prices. The coefficient associated with the risk of jumps is significant and negative, as it is associated with volatility risk. These two results imply that the Bates model has a small advantage when modeling days with steep implied volatility smile and high risk of jumps.

$(\hat{K} < 100)$ contains only put-options.

Table 3: This table reports the results of regressions of the relative performance of the Bates model on various market factors. The relative performance measure is the difference in pricing error between the Bates model and RF model: $MSE_{Bates,i,t} - MSE_{RF,i,t}$), for contract i on date t. The market variables include the VIX index VIX_t, the market jump risk index $JUMP_t$, and the average bid-ask spread across contracts bid-ask$_t$. We estimated model performance with a forecasting horizon of one day $\tau = 5$. The values in parentheses are standard errors, while * denotes significance at the 10% level, ** at 5%, and *** at 1%.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
VIX_t	-0.0011***				-0.0037***			
	(0.0000)				(0.0001)			
$\tilde{VIX}_t$				-0.0023***				-0.0054***
				(0.0000)				(0.0001)
$JUMP_t$		-0.0078***		-0.0076***		0.0172***		-0.0091***
		(0.0005)		(0.0005)		(0.0011)		(0.0011)
bid-ask$_t$			-0.0102***	-0.0113***			0.0291***	0.0242***
			(0.0014)	(0.0014)			(0.0019)	(0.0019)
y_{t-1}	0.2308***	0.2312***	0.2313***	0.2305***	0.1497***	0.1512***	0.1511***	0.1489***
	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0004)
Contract FE	No	No	No	No	Yes	Yes	Yes	Yes
R^2	0.0530	0.0528	0.0527	0.0532	0.1246	0.1238	0.1238	0.1252
Obs	5,307,078	5,307,078	5,307,078	5,307,078	5,307,078	5,307,078	5,307,078	5,307,078

Table 4: This table reports the results of regressions of the relative performance of the Bates model on various market factors. The relative performance measure is the difference in pricing error between the Bates model and RF model: $MSE_{Bates,i,t} - MSE_{RF,i,t}$), for contract i on date t. The market variables include the VIX index VIX_t, the market jump risk index $JUMP_t$, and the average bid-ask spread across contracts bid-ask$_t$. We estimated model performance with a forecasting horizon of one day $\tau = 5$. The values in parentheses are standard errors, while * denotes significance at the 10% level, ** at 5%, and *** at 1%.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
VIX_t	-0.0015***				-0.0046***			
	(0.0)				(0.0001)			
VOL_t^{TA}				-0.0024***				-0.0053***
				(0.0)				(0.0001)
$JUMP_t$		-0.0104***		0.02***		0.0221***		0.0714***
		(0.0006)		(0.0008)		(0.0012)		(0.0013)
bid-ask$_t$			0.0076***	0.0058***			0.0351***	0.0271***
			(0.0014)	(0.0014)			(0.002)	(0.002)
y_{t-1}	0.2343***	0.2351***	0.2352***	0.2343***	0.1553***	0.1575***	0.1573***	0.1552***
	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0005)	(0.0005)	(0.0005)	(0.0005)
Contract FE	No	No	No	No	Yes	Yes	Yes	Yes
Observations	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522
R^2	0.0548	0.0543	0.0542	0.0548	0.1263	0.1251	0.1251	0.1267

Table 5: Orthogonalized vola

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
VIX_t	-0.0015***				-0.0046***			
	(0.0)				(0.0001)			
$\tilde{VOL}_t^{TA}$				-0.0024***				-0.0053***
				(0.0)				(0.0001)
$JUMP_t$		-0.0104***		-0.0094***		0.0221***		0.0053***
		(0.0006)		(0.0006)		(0.0012)		(0.0012)
bid-ask$_t$			0.0076***	0.0058***			0.0351***	0.0271***
			(0.0014)	(0.0014)			(0.002)	(0.002)
y_{t-1}	0.2343***	0.2351***	0.2352***	0.2343***	0.1553***	0.1575***	0.1573***	0.1552***
	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0005)	(0.0005)	(0.0005)	(0.0005)
Contract FE	No	No	No	No	Yes	Yes	Yes	Yes
Observations	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522
R^2	0.0548	0.0543	0.0542	0.0548	0.1263	0.1251	0.1251	0.1267

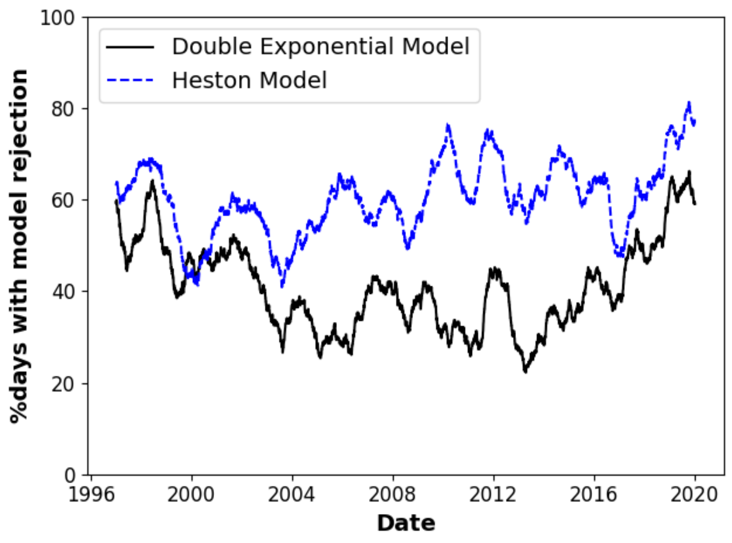
Table 6: Orthogonalized jump

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
VIX_t	-0.0015***				-0.0046***			
	(0.0)				(0.0001)			
VOL_t^{TA}				-0.0013***				-0.0017***
				(0.0)				(0.0001)
$\tilde{JUMP}_t$		0.0295***		0.02***		0.092***		0.0714***
		(0.0008)		(0.0008)		(0.001)		(0.0013)
bid-ask$_t$			0.0076***	0.0058***			0.0351***	0.0271***
			(0.0014)	(0.0014)			(0.002)	(0.002)
y_{t-1}	0.2343***	0.2348***	0.2352***	0.2343***	0.1553***	0.1556***	0.1573***	0.1552***
	(0.0004)	(0.0004)	(0.0004)	(0.0004)	(0.0005)	(0.0005)	(0.0005)	(0.0005)
Contract FE	No	No	No	No	Yes	Yes	Yes	Yes
Observations	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522	4,864,522
R^2	0.0548	0.0545	0.0542	0.0548	0.1263	0.1266	0.1251	0.1267

4.2 Parameter instability

Our fast-to-evaluate surrogates allow us to swiftly and cheaply create a time series of daily estimated parameters. If a structural model is correctly specified, we should expect to see stable parameter estimates over time. However, the plots of parameter estimates for both models show considerable variations over time (see Appendix B for more details).

We now formally test whether the models' parameters are stable across time. To do so, we apply the statistical tests developed by Andersen et al. (2015). Consider time-periods t



(a) $\alpha = 1\%$

Figure 6: The figure above shows, for each day, the percentage of the last 252 days for which the models' parameters are statistically significantly different from one day to the next. We show the average rejection of the test with a 1% significance level.

and $t+1$. Under the null that the pricing model is valid for the two distinct periods, we have

$$(\Theta_t - \Theta_{t+1})' \left(\widehat{\text{Avar}}\left(\Theta_t\right) + \widehat{\text{Avar}}\left(\Theta_{t+1}\right)\right)^{-1} (\Theta_t - \Theta_{t+1}) \xrightarrow{\mathcal{L}-s} \chi^2(q), \tag{23}$$

where Θ_t, and Θ_{t+1} denote the estimated parameters in the time periods t and $t+1$, respectively. $\widehat{\text{Avar}}\left(\Theta_t\right)$ and $\widehat{\text{Avar}}\left(\Theta_{t+1}\right)$ denote consistent estimates of the asymptotic variances of these parameters (cf. Andersen et al. (2015), equations (11)-(12)). Using the test statistic of Eq. (23), we can test whether the parameter estimates Θ_t on each date t are statistically different from those on date $t-1$.

Figure 6 shows the results of those daily tests for both the Bates and HM. On the y-axis, we show the average number of days over the past 252 days for which the hypothesis was rejected, that is, the percentage of days for which the models' parameters at time t and $t-1$ are statistically significantly different from one another ($\alpha = 1\%$). We observe a high rejection rate, which increases after the 2008 crisis. Across the whole sample, we reject with 1% confidence the hypothesis that the models' parameters are stable from one day to the next 41.6% of the time for the Bates and 60.7% of the time for the HM.

Next, we investigate the state variables that can predict parameter stability. To do so, we create a binary indicator $Unstable_{t,Bates}$, which is equal to 1 if we can reject the hypothesis of parameters stability of the Bates model at time t with a 5% probability. In

Table 7: The table below analyzes whether the state of the economy can predict the stability of the Bates model and HM. These logistic regressions predict $Unstable_{t,Bates}$ with various dependent variables.

	(1)	(2)	(3)
$Unstable_{t-1,Bates}$	0.6329***	0.5585***	0.5588***
	(0.053)	(0.0539)	(0.0539)
$JUMP_t - JUMP_{t-1}$			0.3647
			(0.4089)
$JUMP_{t-1}$		-0.3311***	-0.3256***
		(0.0494)	(0.0514)
$(\bar{b-a})_{t-1}$		-0.1935***	-0.1946***
		(0.0271)	(0.0271)
VIX_{t-1}		0.0419***	0.0417***
		(0.006)	(0.0064)
Observations	5,884	5,884	5,884
Pseudo R^2	0.0177	0.0313	0.0322

table 7, we use logistic regressions to predict $Unstable_{t,Bates}$ with various dependent variables. Not surprisingly, parameter instability is persistent, as indicated by the significant positive coefficient on the lagged indicator $Unstable_{t-1,Bates}$. Additionally, high VIX, low risk of jumps, and low bid-ask spreads at time $t-1$ are all associated with a high probability of parameter instability for the Bates model at time t.

Most interestingly, we find that model instability translates directly into liquidity—that is, illiquid markets correlate strongly with high parameters instability. To show this, we estimate the following panel regression:

$$\frac{Ask_{t,i} - Bid_{t,i}}{Bid_{t,i}} = \alpha + \beta_1 \log \chi^2(q)_{t,Bates} + \beta X_{i,t} + \epsilon_{i,t}, \tag{24}$$

Where $Ask_{t,i}$ and $Bid_{t,i}$ are the closing bid and ask price for option i at time t. The controls $X_{i,t}$ include contracts fixed effects, the daily volume at time t ($volume_t$), the level of volatility and risk jumps $VIX_t, JUMP_t$, the remaining time to maturity of the option $T_{i,t}$, as well as the interaction of the remaining time to maturity with a binary C_i equal to 1 if the option i is a call option.

The results displayed in table 8 show a significant positive relationship between parameter stability and option liquidity measured by the bid-ask spread.

Table 8: The table below shows of estimation of Eq. (24), which estimates the effect of parameter instability on the options bid-ask spread.

	(1)	(2)
$\log \chi^2(q)_{t-1,Bates}$		0.0287***
		(0.0017)
$C_iT_{i,t}$	0.0384***	0.0385***
	(0.0002)	(0.0002)
$T_{i,t}$	-0.0660***	-0.0660***
	(0.0001)	(0.0001)
$JUMP_t$	-1.2259***	-1.1806***
	(0.0229)	(0.0231)
VIX_t	-0.1742***	-0.1800***
	(0.0012)	(0.0012)
$volume_t$	-0.0000***	-0.0000***
	(0.0000)	(0.0000)
Contracts FE	YES	YES
Observations	4,948,103	4,948,103
R^2	0.616	0.616

4.2.1 Delta analysis

Market makers and traders often use models to set up hedging portfolios, that is, buying and selling other assets to offset or mitigate the risks inherent to holding options. To these agents, the quality of a model's predicted price sensitivity to the underlying assets is more important than the model's predictive power. In this section, we compare the Bates model and non-parametric models' hedging performance. We focus on the *delta*-hedging, that is, the option's price sensitivity to change in the underlying asset. In previous sections, we used the surrogate technology to re-estimate the parameters of the Bates model. We now use those daily sets of parameters and the original pricing model to estimate the first derivative with respect to the underlying asset $\delta_{i,t}^{(m)}$ for each model m and each option i on the day t.

Next, we compute a replication error assuming traders buy the option and counterbalance the *delta*-risk by taking an opposite position on the underlying assets,

$$\epsilon_{i,t}^{(m)} = \frac{(p_{i,t} - p_{i,t-1}) - \delta_{i,t}^{(m)}(S_t - S_{t-1})}{p_{t-1}}, \tag{25}$$

where S_t is the underlying asset at time t, $p_{i,t}$ is the mid quote of option i at time t. We standardize the error by the option price at time $t-1$. Therefore $\epsilon_{i,t}^{(m)}$ represents the replication error of the model m in percentage of the original price.[23]

Table 9 shows the mean, standard deviations, and important quantiles of the absolute replications errors across the whole sample ($|\epsilon_{i,t}^{(m)}|$). In addition to the random forest and Bates model, we compute the delta of the Black-Scholes model (BSM).[24] A perfect replicating portfolio would produce a replicating error of zero. Note that this result is not achievable here, as, for simplicity, we only hedge the sensitivity to changes in the underlying prices and ignore other well-documented sensitivities: time, volatility, etc. Nonetheless, the relative replicating errors of the various models give a measure of how well-suited these models are to hedging applications.

The Bates model produced an average absolute replication error of 0.14175, which beats both the BSM and RF average hedging performance. The percentile errors suggest that outliers do not produce this over-performance of the Bates model. Finally, we show in section 4.1 that the RF is surprisingly competitive when applied to price predictions, whereas table 9 shows that the RF is not well suited for hedging applications.[25]

Table 9: The table below shows the mean, standard deviation (std), and important percentiles of the absolute replication error for each model m ($|\epsilon_{i,t}^{(m)}|$) in the whole sample.

	BSM	RF	Bates
mean	0.19380	0.20133	0.14175
std	0.42104	0.32173	0.28051
5%	0.00423	0.00754	0.00595
50%	0.08686	0.12512	0.07914
95%	0.65096	0.59113	0.44869

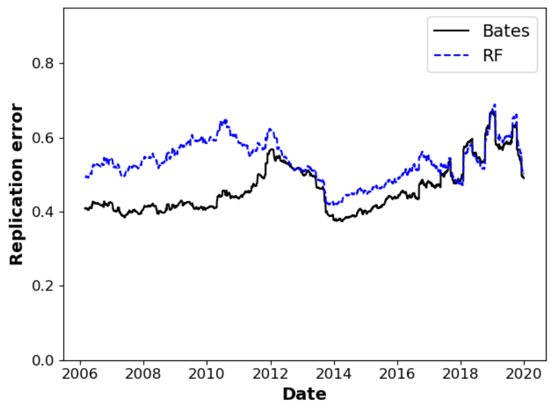
We further explore the relationship between replication error, time, and time to maturity in Figures 7. We show the time series of the average daily $\sqrt{MSE}$ on various subsamples defined by time to maturity. Figure 7(a) and Figure 7(b) show options with a relatively low time to maturity ($0 < T \leq 7$, and ($7 < T \leq 30$), while the last two show the performance on the subsample of options with a relatively long time to maturity ($30 < T \leq 90$, and $90 < T \leq \infty$).

These graphs highlight several interesting patterns. First, we see again that most of the

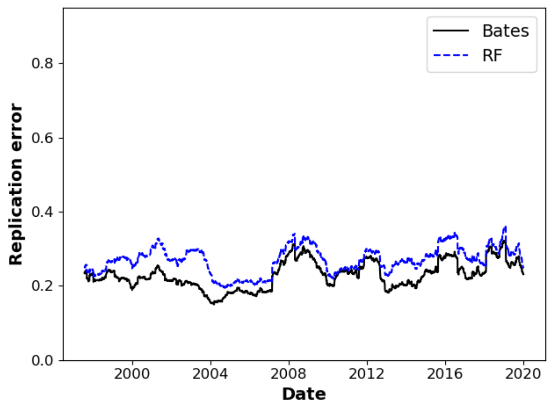
[23]Unlike the $\sqrt{MSE}$ of the previous section, we standardized the replication errors by the price of the options. We do this standardization here because the replication error is expressed in terms of prices instead of BSIV, and prices vary significantly across time to maturity and moneyness, which can introduce bias in the analysis.

[24]For each day, we took the BSIV of at-the-money options as our estimation of the Black-Scholes implied volatility for all options on that particular day.

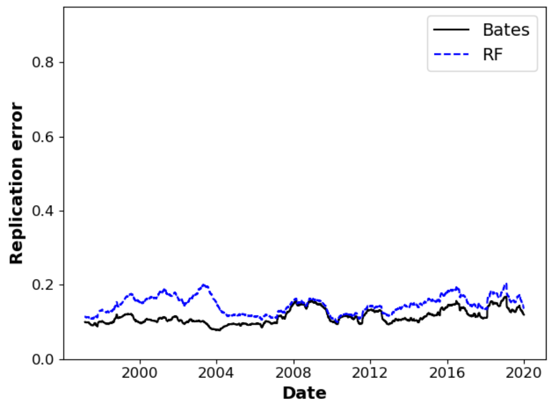
[25]Table A.2 in appendix D shows an extended version of this table.



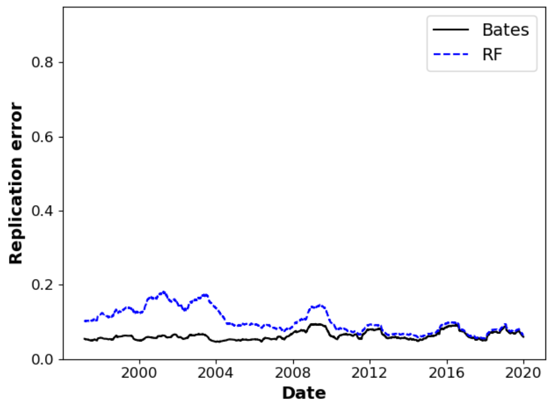
(a) $0 < T \leq 7$



(b) $7 < T \leq 30$



(c) $30 < T \leq 90$



(d) $90 < T \leq \infty$

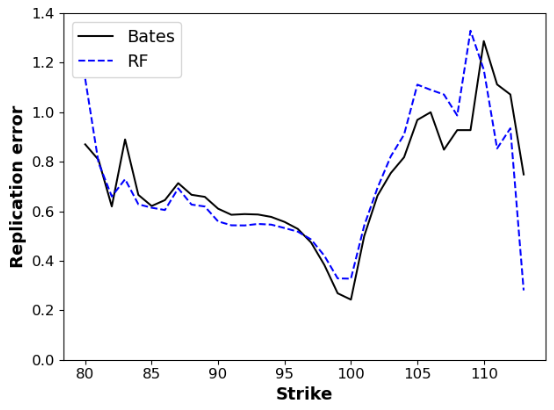
Figure 7: The figures above show the relative mean daily relative replication error smoothed by a 252-day rolling average on various subsamples defined by the options' time to maturity.

Bates model improvement on the non-parametric model concerns short-term options. This observation is in line with Andersen et al. (2017), who highlight that weekly options are very sensitive to the risk of jumps. Second, we see an upward trend in replication errors during 2008. Those results are in line with our analysis of the parameter stability in the last section.

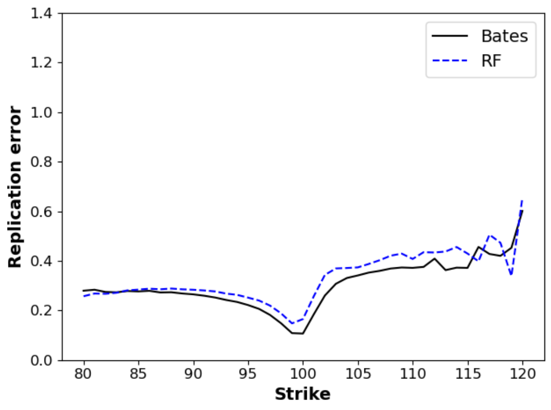
Finally, Figure 7(a) shows that the difference in hedging performance between the two models on options with a short time to maturity is very large at the beginning of the sample but converges toward zero in recent years. This trend can be explained by the number of options with short-term maturity.

Next, we discuss the relationship between replication error, time, and moneyness in figure 8. We show the average $\sqrt{MSE}$ per strike level on various subsamples defined by time to maturity.

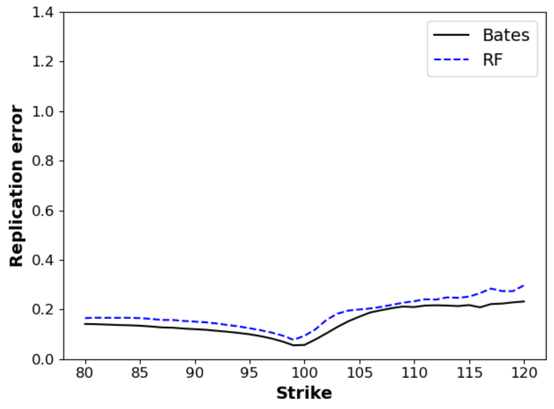
These graphs show that the replication error is smaller for at-the-money options across



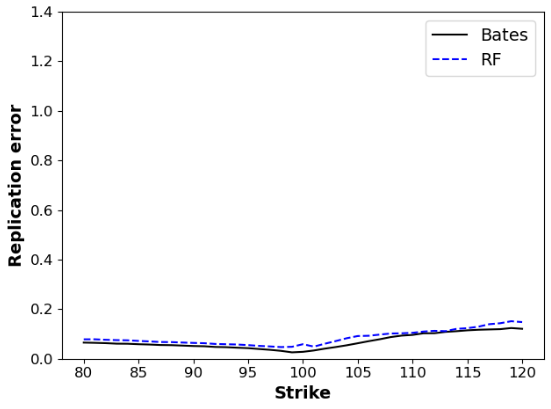
(a) $0 < T \leq 7$



(b) $7 < T \leq 30$



(c) $30 < T \leq 90$



(d) $90 < T \leq \infty$

Figure 8: The figures above show the relative mean daily relative replication error across standardized strikes on various subsamples defined by the options' time to maturity.

all maturity brackets. Indeed, a smile-like shape, reminiscent of the implied volatility smile, appears across all moneyness. Furthermore, we see that the relative overperformance of the Bates model is relatively well spread out across moneyness, with the notable exception of short-term options where the smile is more pronounced.

To conclude this analysis of hedging performance, we use the model states and parameters that we estimated in the previous section to compute the theoretical δ of each option of the subsamples and compute the following regression,

$$\begin{aligned} p_{i,t+1} - p_{i,t} = \beta_{Bates}\delta_{i,Bates}(S_{t+1} - S_t) &\quad + \\ \beta_T(T_{i,t+1} - T_{i,t}) + \beta_J(-\mathbb{1}_{put})(JUMP_{t+1} - JUMP_t) &\quad + \\ \beta_V(VIX_{t+1} - VIX_t), & \end{aligned} \tag{26}$$

Table 10: This table shows the estimations for various configurations of Eq. (26). The values in parentheses are standard errors, whereas * denotes significance at the 10% level, ** at 5%, and *** at 1%.

	(1)	(2)	(4)	(5)	(6)	(7)	(8)
$T_t - T_{t-1}$				-0.2178***	-0.1453***	-0.1561***	-0.1566***
				(0.0008)	(0.0004)	(0.0004)	(0.0004)
$(JUMP_t - JUMP_{t-1})(-\mathbb{1}_{call})$				10.101***	2.8129***	1.4542***	1.3527***
				(0.0664)	(0.029)	(0.0346)	(0.0331)
$VIX_t - VIX_{t-1}$				0.8189***	0.9727***	1.0647***	1.0683***
				(0.0009)	(0.0004)	(0.0005)	(0.0005)
$\delta_{i,t}^{(Bates)}(S_t - S_{t-1})$			0.879***				0.9158***
			(0.0003)				(0.0002)
$\delta_{BS}(S_t - S_{t-1})$		0.92***			0.9361***		
		(0.0002)			(0.0002)		
$\delta_{i,t}^{(RF)}(S_t - S_{t-1})$	82.5215***			80.4929***			
	(0.0603)			(0.056)			
Observations	5,021,299	5,021,299	5,021,299	5,021,299	5,021,299	5,021,299	5,021,299
R^2	0.2717	0.7455	0.6829	0.3768	0.8819	0.832	0.8457

where $p_{i,t+1} - p_{i,t}$ is the change in price for option i from time t to time $t+1$. $\delta_{i,Bates}$ is the theoretical delta for option i, predicted by the Bates model. $(S_{t+1} - S_t)$ is the change in the price of the underlying asset. $(T_{i,t+1} - T_{i,t})$ is the change in maturity in days of the asset.[26] $JUMP_t - JUMP_{t-1})(-\mathbb{1}_{call})$ denotes the change in the estimated risk of jumps. We multiply the change in jump risk by -1 for call options to account for the fact that risks of jump affect put and call options differently. $(VIX_{t+1} - VIX_t)$ is the change in the VIX index, which captures the change in the volatility premium.

We present the estimation of this regression's configurations in table 10. A perfect hedging model would get an R^2 of 100%, and a β_{Bates} of 1.0. While no tested model achieves this high standard, both the Bates and BS model significantly outperform the RF model in this exercise.

[26]Because t denotes trading days and not calendar days, the change in time to maturity of an option from time t to $t+1$ is not always equal to 1. For example, after a Friday, two calendar days separate two trading days.

4.2.2 Market prediction

Kelly and Jiang (2014) showed that tail risks estimated through a power law on the cross-section of returns have strong predictive power for aggregate market returns. Andersen et al. (2020) further showed similar results with tail risks estimated from index options prices.

This section shows that a measure of tail risk estimated through daily re-estimation of the Bates model S&P500 index options outperforms previous estimators of tail risks when predicting market returns both in- and out-of-sample.

First, we look at the predicting power of left tail jump risk. Kelly and Jiang (2014) define this risk as a single parameter $\lambda^{(K)}$, which we estimate following the methodology detail in the original paper. [27] On Todorov and Andersen's website tailindex.com, we find their option implied estimation of a "left tail probability" λ^{AT}. Finally, we define the Bates model measure of left tail jump risk as $(1-p_t^{(Bates)})\lambda_t^{(Bates)}$ where $p_t^{(Bates)}$ and $\lambda_t^{(Bates)}$ are the weekly average of the Bates parameters defined in Eqs. (16) and (17).

We then estimate the following equations in- and out-of-sample:

$$\log\frac{S_{t+h}}{S_t} = \alpha + \beta_1(1-p_t^{(Bates)})\lambda_t^{(Bates)} + \beta_2 v_t^{(Bates)} \tag{27}$$

$$\alpha + \beta_1\lambda_t^{(K)} + \beta_2 v_t^{(Bates)} \tag{28}$$

$$\log\frac{S_{t+h}}{S_t} = \alpha + \beta_1\lambda_t^{(AT)} + \beta_2 v_t^{(Bates)} \tag{29}$$

Where S_t is the price of the S&P500 at time t, and h is the forecasting horizon in number of weeks. [28] We include the Bates estimator $v_t^{(Bates)}$ to control for volatility. Note that the main results are unchanged when this controller is removed or replaced with an estimator of volatility downloaded from tailindex.com.

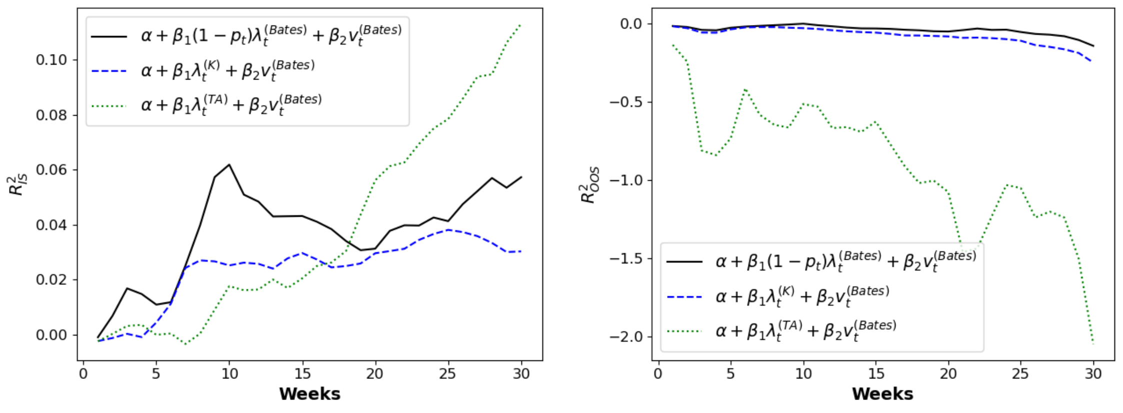
In-sample, we estimate those regressions once on the full sample. Out-of-sample, we re-estimate the regressions once per month with an expanding window and predict the next month's weekly returns out-of-sample. We carefully remove from the expanding window every overlapping return that would not be available at time t. [29]

Figure 9 shows the in-sample adjusted R^2 (Figure 9(a)) and out-of-sample R^2 (Figure 9(b)) for forecasting horizons between 1 and 30 weeks. In-sample, the adjusted R^2 increases with

[27]We differ from the original methodology in that Kelly and Jiang (2014) estimates $\lambda^{(K)}$ every month, and we estimate $\lambda^{(K)}$ every week with a four weeks rolling window.

[28]Following Andersen et al. (2020), we focus our analysis on weekly forecasts. We define a weekly return as the log return from Wednesday to Wednesday.

[29]Since the data provided on tailindex.com starts in 2008, we restrict our analysis to the subsample from 2008-2019. However, we use the full sample when available to calibrate the parameters of the out-of-sample regressions.



(a) (b)

Figure 9: The figures above show the in-sample adjusted R^2 (panel a) and out-of-sample R^2 (panel b) of regression (27), (28), (29), for forecasting horizons between 1 and 30 weeks.

the forecasting horizon, and the Bates-based tail risk predictors significantly outperform both $\lambda^{(K)}$ and $\lambda^{(AT)}$. Out-of-sample, no predictors manage to produce a significant R^2, but the Bates predictors still outperform their competitors.

To further investigate the predicting strength of those three measures of tail risk, we estimate a so-called "horse race" regression:

$$\log \frac{S_{t+h}}{S_t} = \alpha + \beta_1^{(Bates)}(1 - p_t^{(Bates)})\lambda_t^{(Bates)} + \beta_2^{(TA)}\lambda_t^{(TA)} + \beta_3^{(K)}\lambda_t^{(K)} \tag{30}$$

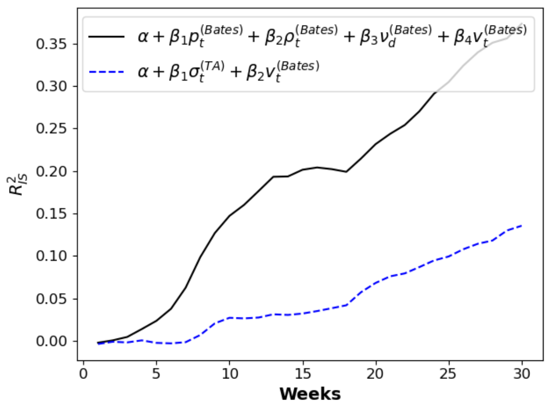
We compare the predicting power of the option-implied return volatility generated by the left tail downloaded on tailindex.com $\sigma_t^{(AT)}$ and the parameters of the Bates model defining the option-implied return volatility generated by the left tail which includes: $p_t^{Bates}, \rho_t^{Bates}, \nu_{d,t}^{Bates}$ as defined in equations (16) and (17).

We estimate, in- and out-of-sample:

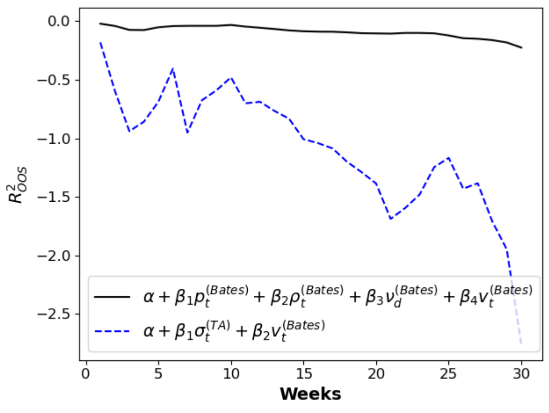
$$\log \frac{S_{t+h}}{S_t} = \alpha + \beta_1 p_t^{(Bates)} + \beta_2 \rho_t^{(Bates)} + \beta_3 \nu_d^{(Bates)} + \beta_4 v_t^{(Bates)} \tag{31}$$

$$\log \frac{S_{t+h}}{S_t} = \alpha + \beta_1 \sigma_t^{(AT)} + \beta_2 v_t^{(Bates)}. \tag{32}$$

Figure 10 shows the in- and out-of-sample R^2. As with figure 9, the Bates dominates the other estimation of the tail risk, but neither the Bates estimate nor the AT estimate does produce a positive R^2 out-of-sample.



(a)



(b)

Figure 10: The figures above show the in-sample adjusted R^2 (panel a) and out-of-sample R^2 (panel b) when predicting market returns with left-tail volatility estimators. On the x-axis, we vary the forecasting horizons between 1 and 30 weeks.

4.3 Empirical distribution and value-at-risk

In this section, we adopt the method proposed by Israelov and Kelly (2017) to estimate the distribution of option prices at time $t + 1$. This exercise is particularly interesting as it allows practitioners and academics to use complex parametric models to estimate important quantities for risk management, like the value-at-risk.

Israelov and Kelly (2017) argues that forecasting the option prices can be separated into two distinct tasks: a) forecasting the changes in the underlying asset price and b) forecasting changes in the implied volatility surface. They then used two model-free vector autoregressive models (VAR) two execute these two tasks. To estimate a distribution of future option prices, they used a bootstrap sample of historical errors from the VAR models to price a random sample of 5000 potential future prices per option with which they can estimate moments and quantiles of an empirical distribution.

Note that the computational cost of adapting this methodology to a structural model like the Bates model could be problematic without the deep-surrogate technology. Estimating the efficiency of the methodology on a reasonable subsample of 100 options per day for ten years would require the pricing of 126 million options.[30]

We use a method similar to that of Israelov and Kelly (2017) to estimate a distribution of changes in the price of the underlying S&P500 index (S_t) and the base volatility level v_t. We define the following VAR(1):

[30]256 business days x 10 years x 100 options x 5000 bootstrap samples.

$$X_t = \mu + \rho X_{t-1} + \epsilon_t, \tag{33}$$

where $X_t = \left[\frac{S_t}{S_{t-1}} - 1, \log \mathrm{VIX}_t, \mathrm{PC}_t'\right]'$, with PC_t' being the first components of the implied volatility surface as defined in Israelov and Kelly (2017).

For every day in the sample with at least 1000 observations, we estimate Eq. (33) and bootstrap $b = 1, \ldots, 5000$, residuals which we label $\hat{\epsilon}_{t+1}^b$ which we feed to Eq. (33) to estimate the $t+1$ bootstrap forecast at time $t+1$:

$$\hat{X}_{t+1}^b = \hat{\mu} + \hat{\rho} X_t + \hat{\Sigma}_t \hat{\epsilon}_{t+1}^b, \quad b = 1, \ldots, 5000, \tag{34}$$

where $X_{t+1}^b = \left[\frac{S_{t+1}^b}{S_t} - 1, \log \mathrm{VIX}_{t+1}^b, \mathrm{PC}_{t+1}'^b\right]'$. Finally, we use the VIX forecast to estimate the change in the base level of volatility v_t by assuming these changes are proportional and define $v_{t+1}^b = \frac{VIX_{t+1}^b}{VIX_t} v_t$.

Next, with our bootstrap estimation of the volatility level v_{t+1}^b, and index prices at time $t+1$, S_{t+1}^b, we estimate the bootstrap BSIV of an option i at time t as:

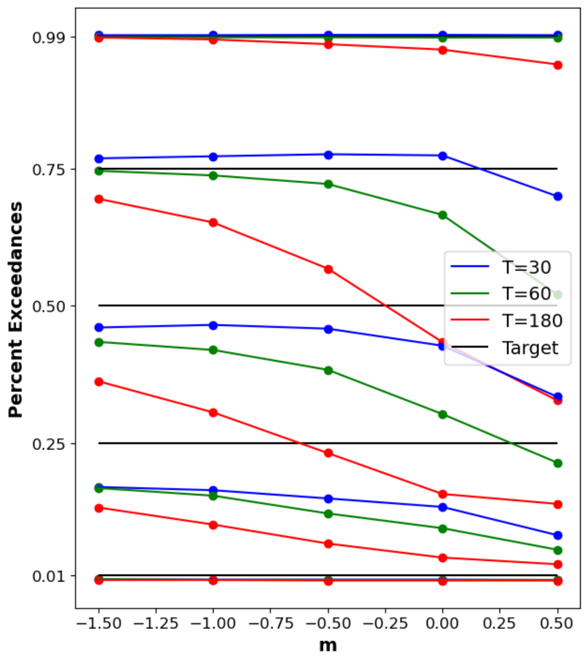
$$\hat{y}_i^b = \phi\left(\left[\Omega_{t+1}^{(b)}, H_{t+1}^{(b)}, \Theta_t\right] \mid \Theta_{NN}^*\right), \tag{35}$$

where Θ_t contains the parameters of the model estimated at time t as described in section in **??**, $H_{t+1}^{(b)} = [v_{t+1}^b]$ is the vector of forcasted hidden state, and $\Omega_{t+1}^{(b)} = \left[S_{t+1}^b, K, T - (t+1)\right]^T$ is the vector of forcasted observable states at time $t+1$.

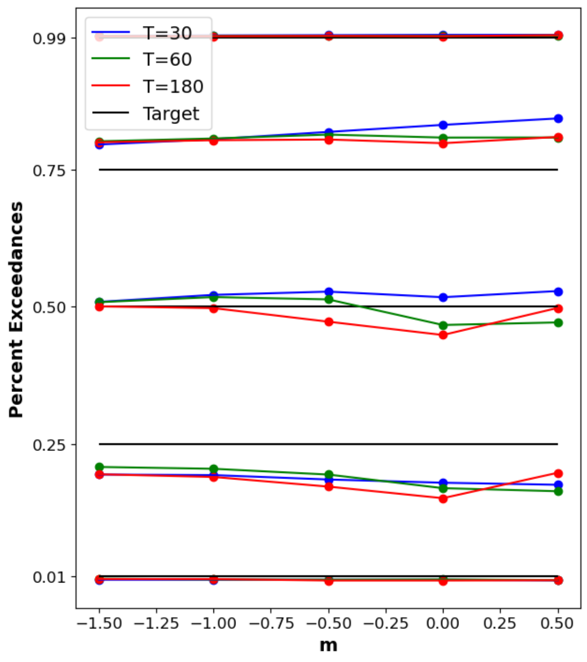
The main advantage of the methodology described above is that it produces an empirical distribution of the options' future prices as opposed to a point estimate. As non-parametric models are not well suited for this methodology, we use as a comparison the HM. To test the quality of this distribution for both models, we follow Israelov and Kelly (2017) and compare theoretical quantiles to realized price changes.

We group our options into three time to maturity brackets $T = [30, 60, 180]$ and 5 moneyness brackets $m = [-1.5, -1, -0.5, 0.0, 0.5]$. We keep all options that deviate by less than 10% of the moneyness and maturity target for each day of the sample and each option bracket. We then compute empirical quantiles for each option and report the percentage exceedance, that is, the average number of day-options in the sample for which the realized price was below the predicted quantile. If the empirical distribution is correct, this value should be equal to the predicted quantile.

Note that the methodology requires a parametric structure. Thus, the RF cannot be used as a benchmark of the Bates model. Hence, in the remainder of this section, we will use the



(a) HM



(b) Bates

Figure 11: The figures above show the performance of the HM (a) and Bates model (b) to estimate the empirical distribution of option prices at time $t + 1$. On the y-axis, we display the percent exceedance of both models as well as the target quantile. On the x-axis, we show the moneyness of the option brackets as defined by Eq. (18).

HM instead.

Figure 11 reports the results. On the y-axis, we display the target quantile (black line) and the percentage of day-options per maturity brackets for which the price at time $t + 1$ was below the quantile predicted at time t (blue, green, and red line). On the x-axis, we display the moneyness of the bracket. Figure 11(a) shows the results for the Heston model, and Figure 11(b). Figure A.3 further highlights the performance on extreme quantiles.

The Bates model strongly outperforms the HM model in this exercise. The 1% and 99% quantiles, which correspond to the 1% value-at-risk for an agent buying or selling the option, are fairly well estimated by both models, except for out-of-the-money call options with a long time to maturity for the Heston model. The Bates model also predicts the median fairly well, but both quantiles seem to underestimate the true value.

5 Conclusion

In this paper, we introduce *deep surrogates*: a generic framework to swiftly estimate complex structural models in economics and finance. We treat the model parameters as pseudo-state variables to create a deep-neural network surrogate replicating the target model. By alleviating the curse of dimensionality, this surrogate approach considerably lowers the computational cost for the prediction and parameter estimation. Such a speed gain is non-trivial, as high computational costs often prohibit important analyses of structural models, including (i) out-of-sample analysis, (ii) testing of parameter stability, and (iii) re-calibration and testing on multiple sub-samples. All three of these examples require a fast re-estimation of structural models, which is often infeasible without the surrogate technology.

As an application, we perform the first thorough analysis of a complex option pricing model: the Bates model with a double-exponential jump-diffusion. The complexity of this model rendered its calibration complex, and previous literature only performed one or two calibrations. Thanks to the surrogate technology, we re-estimated the model's parameters and hidden states at a daily frequency.

Our analysis reveals that, on the one hand, as a price predictor, the Bates model only outperforms a simple non-parametric benchmark on specific parts of the implied volatility surface. On the other hand, the Bates models completely outperform the benchmark as a tool to define optimal delta-hedging positions.

Comparing the Bates model to the simpler HM, we find that the Bates model's parameters are significantly more stable across time. Furthermore, the instability of the Bates parameters is positively correlated with the low liquidity of the options market. We also show that the tail risk estimated through the daily-re-estimation of the Bates models outperforms predictors of market risk by the extant literature. Finally, adapting the method proposed by Israelov and Kelly (2017), we show that the Bates models with daily re-calibration can be used to estimate the empirical distribution of options returns and their vale-at-risk. The quality of those estimates drastically drops when using the HM instead of Bates.

Taken together, the result of these sample analyses serve a dual purpose. First, it highlights the strength and weaknesses of the current option theory. Indeed, the out-of-sample approach made possible by the *deep-surrogate* technology shows applications where non-parametric models can match state-of-the-art option pricing theory and specific area of the implied volatility surface where the Bates model is outperformed by simple non-parametric alternative. Second, it highlights specific applications of structural models that couldn't be possible without deep-surrogate technology. Indeed, the estimation of value-at-risk or daily estimating of option-implied tail risk from the Bates model would be computationally extremely challenging without a *deep-surrogate*.

The *deep-surrogate* methodology opens the door to a vast number of previously infeasible applications of structural models in finance and economics. Furthermore, *deep-surrogates* can easily be shared by researchers to allow others to use their model. This last benefit of our methodology is non-negligible. Indeed, it has the potential to greatly increase the reproducibility of research and reduce the time and cost of future research.

References

Aldrich, Eric M, Jesús Fernández-Villaverde, A Ronald Gallant, and Juan F Rubio-Ramírez, 2011, Tapping the supercomputer under your desk: Solving dynamic equilibrium models with graphics processors, *Journal of Economic Dynamics and Control* 35, 386–393.

Andersen, Torben G, Nicola Fusari, and Viktor Todorov, 2015, Parametric inference and dynamic state recovery from option panels, *Econometrica* 83, 1081–1145.

Andersen, Torben G, Nicola Fusari, and Viktor Todorov, 2017, Short-term market risks implied by weekly options, *The Journal of Finance* 72, 1335–1386.

Andersen, Torben G, Nicola Fusari, and Viktor Todorov, 2020, The pricing of tail risk and the equity premium: Evidence from international option markets, *Journal of Business & Economic Statistics* 38, 662–678.

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger, 2022, Deep equilibrium nets, *International Economic Review* n/a.

Bach, Francis, 2017, Breaking the curse of dimensionality with convex neural networks, *Journal of Machine Learning Research* 18, 1–53.

Bates, David S, 1996, Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options, *The Review of Financial Studies* 9, 69–107.

Bates, David S., 2000, Post-'87 crash fears in the S&P 500 futures option market, *Journal of Econometrics* 94, 181–238.

Becker, Sebastian, Patrick Cheridito, and Arnulf Jentzen, 2018, Deep optimal stopping, *arXiv preprint arXiv:1804.05394* .

Berner, Julius, Philipp Grohs, and Arnulf Jentzen, 2020, Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations, *SIAM Journal on Mathematics of Data Science* 2, 631–657.

Bilionis, Ilias, and Nicholas Zabaras, 2012a, Multi-output local gaussian process regression: Applications to uncertainty quantification, *Journal of Computational Physics* 231, 5718–5746.

Bilionis, Ilias, and Nicholas Zabaras, 2012b, Multidimensional adaptive relevance vector machines for uncertainty quantification, *SIAM Journal on Scientific Computing* 34, B881–B908.

Bilionis, Ilias, Nicholas Zabaras, Bledar A Konomi, and Guang Lin, 2013, Multi-output separable gaussian process: Towards an efficient, fully bayesian paradigm for uncertainty quantification, *Journal of Computational Physics* 241, 212–239.

Bollerslev, Tim, and Viktor Todorov, 2011, Tails, fears, and risk premia, *The Journal of Finance* 66, 2165–2211.

Breiman, Leo, 2001, Random forests, *Machine Learning* 45, 5–32.

Bretscher, Lorenzo, Jesus Fernandez-Villaverde, and Simon Scheidegger, 2022, Ricardian business cycle, *Available at SSRN 4278274* .

Brumm, Johannes, Christopher Krause, Andreas Schaab, and Simon Scheidegger, 2022, Sparse grids for dynamic economic models .

Brumm, Johannes, and Simon Scheidegger, 2017, Using adaptive sparse grids to solve high-dimensional dynamic models, *Econometrica* 85, 1575–1612.

Buehler, H., L. Gonon, J. Teichmann, and B. Wood, 2019, Deep hedging, *Quantitative Finance* 0, 1–21.

Chauvin, Yves, and David E Rumelhart, 1995, *Backpropagation: theory, architectures, and applications* (Psychology press).

Chen, Hui, Winston Wei Dou, and Leonid Kogan, 2022, Measuring “dark matter” in asset pricing models, *Journal of Finance*, forthcoming.

Chen, Luyang, Markus Pelger, and Jason Zhu, 2019, Deep learning in asset pricing, Working paper.

Chen, Peng, Nicholas Zabaras, and Ilias Bilionis, 2015, Uncertainty propagation using infinite mixture of gaussian processes and variational bayesian inference, *Journal of computational physics* 284, 291–333.

Chen, Xiaohong, and Sydney C Ludvigson, 2009, Land of addicts? an empirical investigation of habit-based asset pricing models, *Journal of Applied Econometrics* 24, 1057–1093.

Chen, Xiaohong, and Halbert White, 1999, Improved rates and asymptotic normality for nonparametric neural network estimators, *IEEE Transactions on Information Theory* 45, 682–691.

Christoffersen, Peter, and Kris Jacobs, 2004, The importance of the loss function in option valuation, *Journal of Financial Economics* 72, 291 – 318.

Constantine, Paul G., 2015, *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies* (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA).

Deisenroth, Marc Peter, Carl Edward Rasmussen, and Jan Peters, 2009, Gaussian process dynamic programming, *Neurocomputing* 72, 1508–1524.

Didisheim, Antoine, Dimitrios Karyampas, and Simon Scheidegger, 2020, Implied risk aversion smile, *Available at SSRN 3533089* .

Duarte, Victor, 2018, Machine learning for continuous-time economics Working paper.

Duarte, Victor, Julia Fonseca, Aaron S Goodman, and Jonathan A Parker, 2021, Simple allocation rules and optimal portfolio choice over the lifecycle, Working Paper 29559, National Bureau of Economic Research.

Duffie, Darrell, Jun Pan, and Kenneth Singleton, 2000, Transform analysis and asset pricing for affine jump-diffusions, *Econometrica* 68, 1343–1376.

Farrell, Max H., Tengyuan Liang, and Sanjog Misra, 2021, Deep neural networks for estimation and inference, *Econometrica* 89, 181–213.

Fernández-Villaverde, J., J.F. Rubio-Ramírez, and F. Schorfheide, 2016, None, volume 2 of *Handbook of Macroeconomics*, 527 – 724 (Elsevier).

Fernandez-Villaverde, Jesus, Samuel Hurtado, and Galo Nuno, 2019, Financial Frictions and the Wealth Distribution, PIER Working Paper Archive 19-015, Penn Institute for Economic Research, Department of Economics, University of Pennsylvania.

Fernandez-Villaverde, Jesus, Galo Nunu, George Sorg-Langhans, and Maximilian Vogler, 2020, Solving high-dimensional dynamic programming problems using deep learning, Technical report.

Fernández-Villaverde, Jesús, and Pablo A Guerrón-Quintana, 2020, Estimating dsge models: Recent advances and future challenges, Working Paper 27715, National Bureau of Economic Research.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio, 2016, *Deep learning*, volume 1 (MIT press Cambridge).

Gu, Shihao, Bryan Kelly, and Dacheng Xiu, 2018, Empirical asset pricing via machine learning, Technical report, National Bureau of Economic Research.

Gühring, Ingo, Mones Raslan, and Gitta Kutyniok, 2020, Expressivity of deep neural networks, *arXiv preprint arXiv:2007.04759* .

Han, Jiequn, Yucheng Yang, et al., 2021, Deepham: A global solution method for heterogeneous agent models with aggregate shocks, Technical report, arXiv preprint arXiv:2112.14377.

Heiss, Florian, and Viktor Winschel, 2008, Likelihood approximation by numerical integration on sparse grids, *Journal of Econometrics* 144, 62 – 80.

Heston, Steven L, 1993, A closed-form solution for options with stochastic volatility with applications to bond and currency options, *The review of financial studies* 6, 327–343.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White, 1989, Multilayer feedforward networks are universal approximators, *Neural networks* 2, 359–366.

Hutchinson, James M, Andrew W Lo, and Tomaso Poggio, 1994, A nonparametric approach to pricing and hedging derivative securities via learning networks, *The Journal of Finance* 49, 851–889.

Igami, Mitsuru, 2020, Artificial intelligence as structural estimation: Deep Blue, Bonanza, and AlphaGo, *The Econometrics Journal* 23, S1–S24.

Iskhakov, Fedor, John Rust, and Bertel Schjerning, 2020, Machine learning and structural econometrics: contrasts and synergies, *The Econometrics Journal* 23, S81–S124.

Israelov, Roni, and Bryan T Kelly, 2017, Forecasting the distribution of option returns, *Available at SSRN 3033242* .

Judd, Kenneth, 1996, Approximation, perturbation, and projection methods in economic analysis, in H. M. Amman, D. A. Kendrick, and J. Rust, eds., *Handbook of Computational Economics*, volume 1, first edition, chapter 12, 509–585 (Elsevier).

Judd, Kenneth L., Lilia Maliar, Serguei Maliar, and Rafael Valero, 2014, Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain, *Journal of Economic Dynamics and Control* 44, 92 – 123.

Kadan, Ohad, and Xiaoxiao Tang, 2020, A bound on expected stock returns, *The Review of Financial Studies* 33, 1565–1617.

Kaji, Tetsuya, Elena Manresa, and Guillaume Pouliot, 2020, An Adversarial Approach to Structural Estimation, *arXiv e-prints* arXiv:2007.06169.

Kase, Hanno, Leonardo Melosi, and Matthias Rottner, 2022, Estimating nonlinear heterogeneous agents models with neural networks, *Available at SSRN 4138711* .

Kelly, Bryan, and Hao Jiang, 2014, Tail risk and asset prices, *The Review of Financial Studies* 27, 2841–2871.

Krause, Andreas, and Carlos Guestrin, 2007, Nonmyopic active learning of gaussian processes: an exploration-exploitation approach, in *Proceedings of the 24th international conference on Machine learning*, 449–456.

Krueger, Dirk, and Felix Kubler, 2006, Pareto-improving social security reform when financial markets are incomplete!?, *American Economic Review* 96, 737–755.

Liaw, Andy, Matthew Wiener, et al., 2002, Classification and regression by randomforest, *R news* 2, 18–22.

Liu, Shuaiqiang, Anastasia Borovykh, Lech A Grzelak, and Cornelis W Oosterlee, 2019, A neural network-based framework for financial model calibration, *Journal of Mathematics in Industry* 9, 9.

Maliar, Lilia, and Serguei Maliar, 2014, Chapter 7 - numerical methods for large-scale dynamic economic models, in Karl Schmedders, and Kenneth L. Judd, eds., *Handbook of Computational Economics Vol. 3*, volume 3 of *Handbook of Computational Economics*, 325 – 477 (Elsevier).

Maliar, Lilia, Serguei Maliar, and Pablo Winant, 2021, Deep learning for solving dynamic economic models., *Journal of Monetary Economics* 122, 76–101.

Montanelli, Hadrien, and Qiang Du, 2019, New error bounds for deep relu networks using sparse grids, *SIAM Journal on Mathematics of Data Science* 1, 78–92.

Montanelli, Hadrien, Haizhao Yang, and Qiang Du, 2021, Deep relu networks overcome the curse of dimensionality for generalized bandlimited functions, *Journal of Computational Mathematics* 39, 801–815.

Nakkiran, Preetum, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever, 2021, Deep double descent: Where bigger models and more data hurt, *Journal of Statistical Mechanics: Theory and Experiment* 2021, 124003.

Norets, Andriy, 2012, Estimation of dynamic discrete choice models using artificial neural network approximations, *Econometric Reviews* 31, 84–106.

Pan, Jun, 2002, The jump-risk premia implicit in options: evidence from an integrated time-series study, *Journal of Financial Economics* 63, 3–50.

Park, Jooyoung, and Irwin W Sandberg, 1991, Universal approximation using radial-basis-function networks, *Neural computation* 3, 246–257.

Petersen, Philipp, and Felix Voigtlaender, 2018, Optimal approximation of piecewise smooth functions using deep relu neural networks, *Neural Networks* 108, 296–330.

Pflueger, Dirk, Benjamin Peherstorfer, and Hans-Joachim Bungartz, 2010, Spatially adaptive sparse grids for high-dimensional data-driven problems, *Journal of Complexity* 26, 508 – 522.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, 2007, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, third edition (Cambridge University Press, New York, NY, USA).

Ramachandran, Prajit, Barret Zoph, and Quoc V. Le, 2017, Swish: a self-gated activation function, *arXiv: Neural and Evolutionary Computing* .

Renner, Philipp, and Simon Scheidegger, 2018, Machine learning for dynamic incentive problems, *Available at SSRN 3282487* .

Schaefer, Stephen M., and Ilya A. Strebulaev, 2008, Structural models of credit risk are useful: Evidence from hedge ratios on corporate bonds, *Journal of Financial Economics* 90, 1 – 19.

Scheidegger, S., D. Mikushin, F. Kubler, and O. Schenk, 2018, Rethinking large-scale economic modeling for efficiency: Optimizations for gpu and xeon phi clusters, in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 610–619.

Scheidegger, Simon, and Ilias Bilionis, 2019, Machine learning for high-dimensional dynamic stochastic economies, *Journal of Computational Science* 33, 68 – 82.

Scheidegger, Simon, and Adrien Treccani, 2018, Pricing American Options under High-Dimensional Models with Recursive Adaptive Sparse Expectations*, *Journal of Financial Econometrics* nby024.

Stephenson, Cory, and Tyler Lee, 2021, When and how epochwise double descent happens, *arXiv preprint arXiv:2108.12006* .

Tripathy, Rohit, Ilias Bilionis, and Marcial Gonzalez, 2016, Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation, *Journal of Computational Physics* 321, 191–223.

Tripathy, Rohit K, and Ilias Bilionis, 2018a, Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification, *Journal of computational physics* 375, 565–588.

Tripathy, Rohit K., and Ilias Bilionis, 2018b, Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification, *Journal of Computational Physics* 375, 565 – 588.

Villa, Alessandro T, and Vytautas Valaitis, 2019, Machine learning projection methods for macro-finance models, *Economic Research Initiatives at Duke (ERID) Working Paper* .

Williams, Christopher KI, and Carl Edward Rasmussen, 2006, *Gaussian processes for machine learning*, volume 2 (MIT press Cambridge, MA).

Appendix

A Simulated results

Table A.1: This table presents the ranges for the surrogate model of the HM and Bates's training sample.

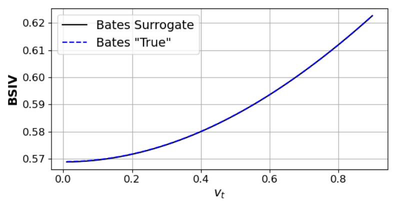
j	HM: $\underline{x}^{(j)}$	HM: $\bar{x}^{(j)}$	Bates: $\underline{x}^{(j)}$	Bates: $\bar{x}^{(j)}$
m	-9.00	5.000	-9.00	5.000
rf	0.00	0.075	0.00	0.075
dividend	0.00	0.050	0.00	0.050
v_t	0.01	0.900	0.01	0.900
T	1.00	365.000	1.00	365.000
κ	0.10	50.000	0.10	50.000
θ	0.01	0.900	0.01	0.900
σ	0.10	5.000	0.10	5.000
ρ	-	-	-1.00	-0.000
λ	-	-	0.00	4.000
ν_1	-	-	0.00	0.400
ν_2	-	-	0.00	0.400
p	-	-	0.00	1.000

B Parameters across time

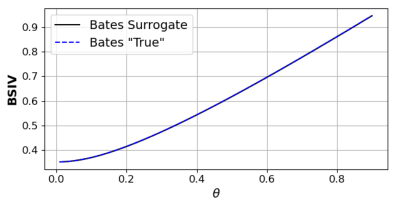
Figure A.4 shows, for the Bates model and HM, the time series of the hidden state v_t re-estimated at a daily frequency and smooth with a rolling average over the last 252 days. We can see that both time series are highly correlated across time. Furthermore, the volatility estimated with the HM is slightly higher than that estimated by the Bates.

Figures A.6 and A.5 in appendix B display the value of the parameters re-estimated daily for both models. We smooth each parameter across time with a rolling average over the last 252 days. We can see a strong variation of all parameters across time.

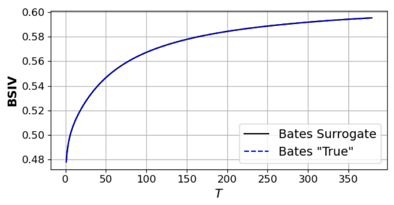
C Errors across strikes and maturities



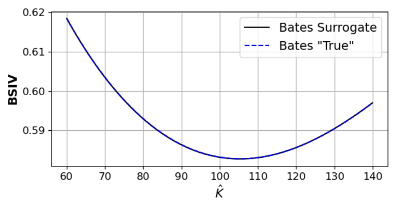
(a)



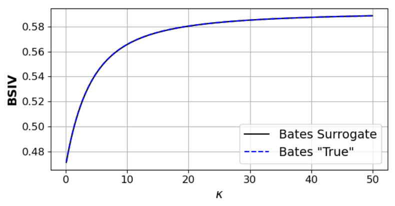
(b)



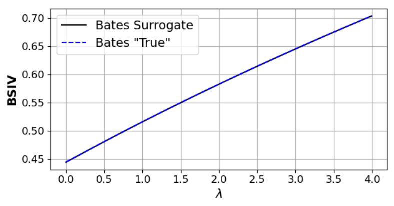
(c)



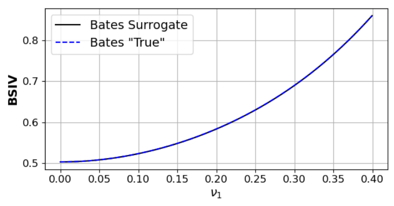
(d)



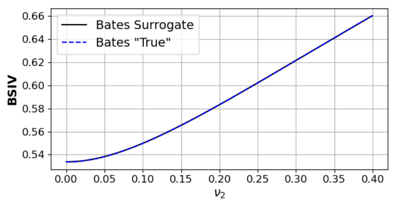
(e)



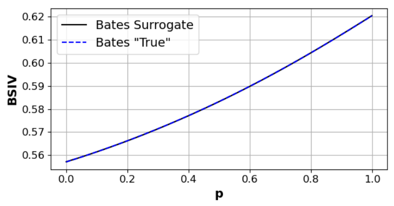
(f)



(g)



(h)



(i)

Figure A.1: The figures above show the complexity of the the Bates model's volatility surface as well as the quality of the surrogate's interpolations. To do so, the figures compare the BSIVs of the surrogate against the predictions from the "true" Bates. We populate the state space with points, where we keep all parameters and states fixed at the mid-range of possible values: $\hat{K} = 100.0, r = 0.0375, T = 190, \kappa = 25.05, \theta = 0.455, v_t = 0.455, \sigma = 2.55$. On each panel, we show the BSIV predicted by the Bates and its surrogate while varying one of the parameters or states in its admissible range (cf. table A.1).

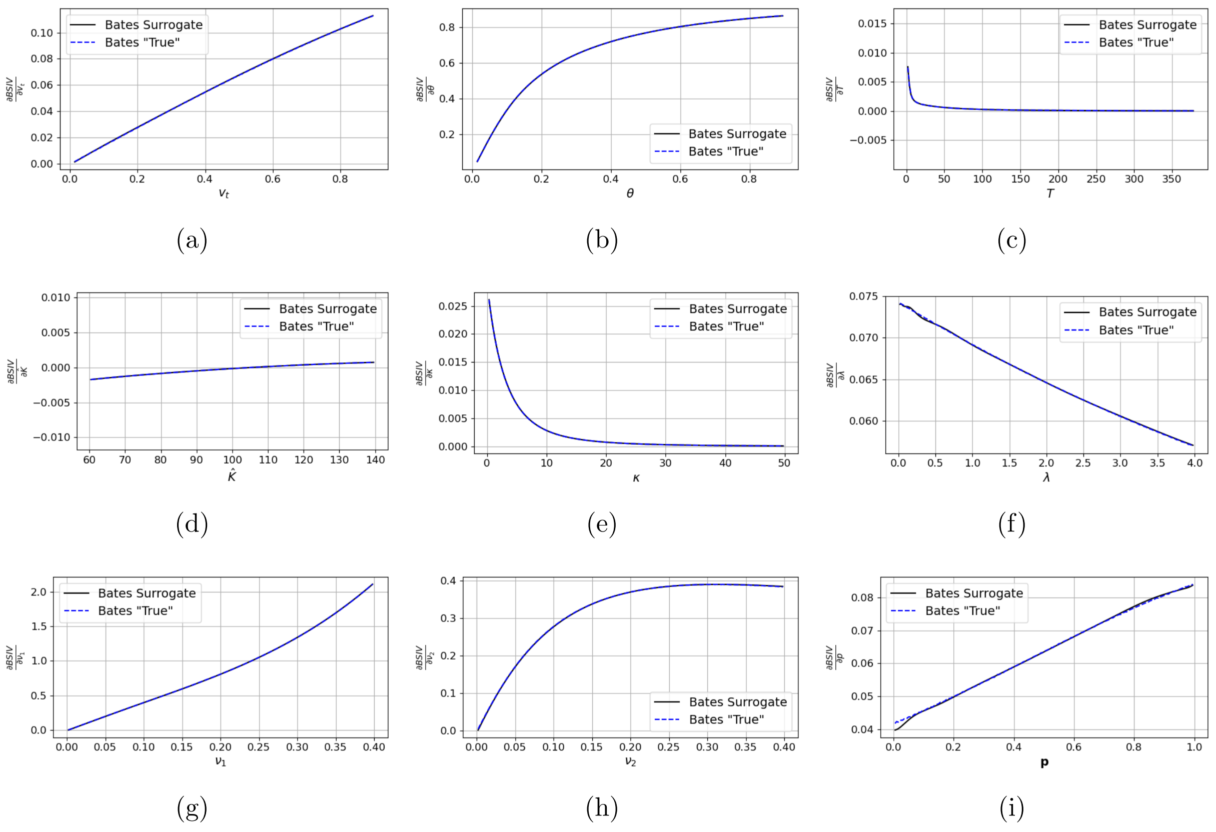
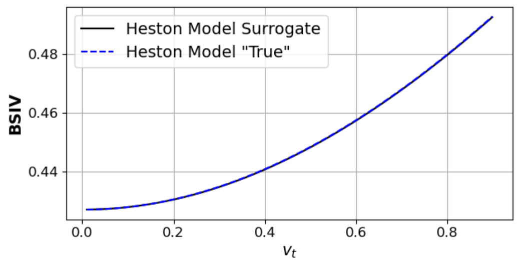
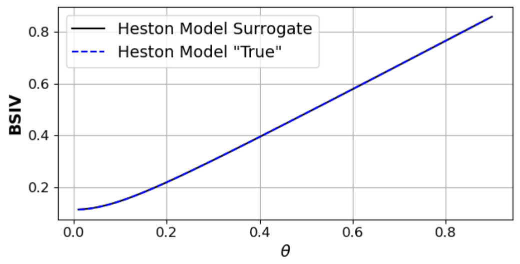


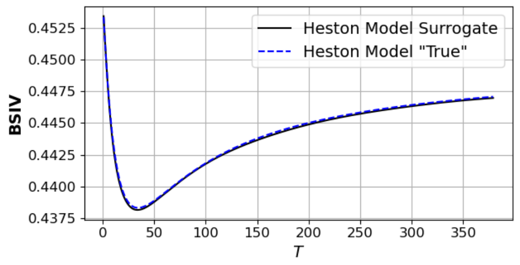
Figure A.2: The figures above show the complexity of the Bates models' volatility surface as well as the quality of the surrogate's interpolations. To do so, the figures compare the BSIVs of the surrogate against the predictions of the "true" Bates. We populate the state space with points, where we keep all parameters and states fixed at the mid-range of possible values: $\hat{K} = 100.0, r = 0.0375, T = 190, \kappa = 25.05, \theta = 0.455, v_t = 0.455, \sigma = 2.55$. On each panel, we show the BSIV predicted by the Bates and its surrogate while varying one of the parameters or states in its admissible range (cf. table A.1).



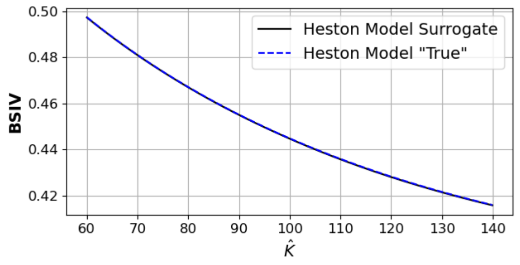
(a)



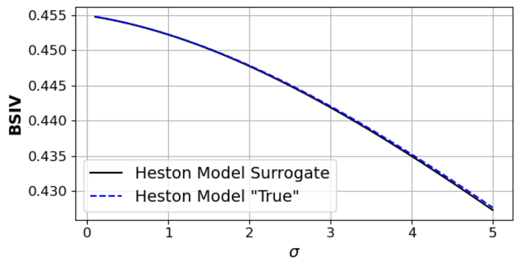
(b)



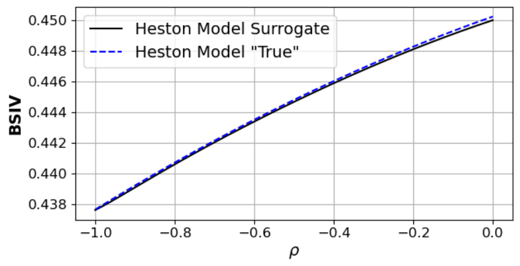
(c)



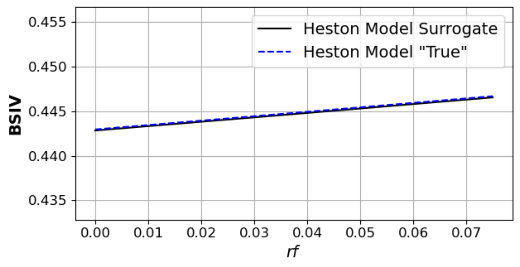
(d)



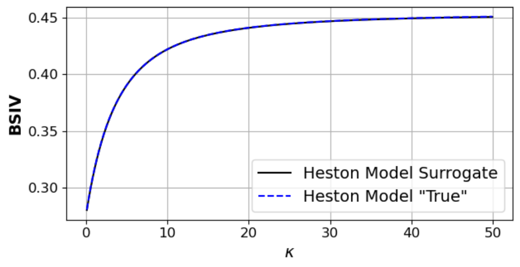
(e)



(f)

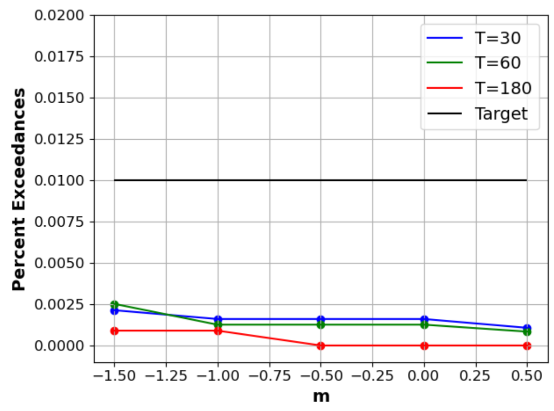


(g)

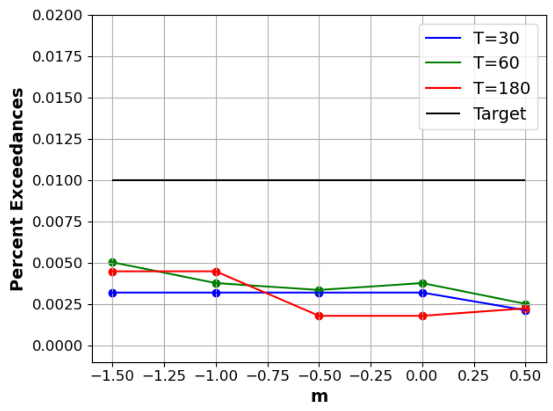


(h)

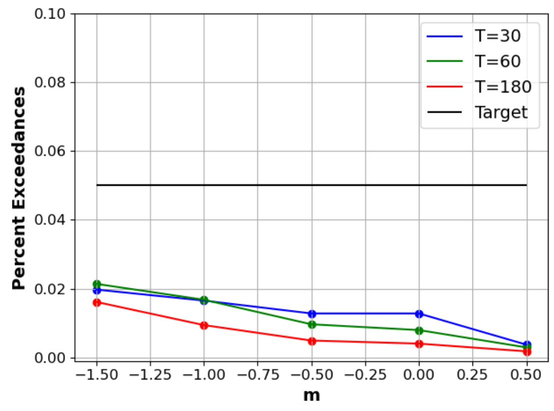
Figure A.2: The figures above compare the partial derivative of the BSIVs with regard to the models pseudo-state. We compare this quantities estimated with the surrogate and the the "true" Bates. We populate the state space with points, where we keep all parameters and states fixed at the mid-range of possible values: $\hat{K} = 100.0, rf = 0.0375, T = 190, \kappa = 25.05, \theta = 0.455, v_t = 0.455, \sigma = 2.55$. On each panel, we show the partial derivative predicted by the model and its surrogate while varying one of the parameters or states in its admissible range.



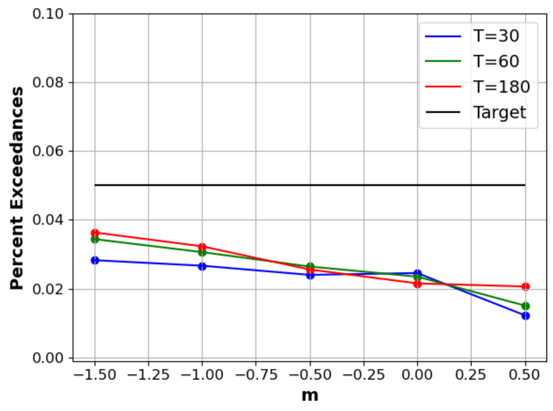
(i) HM, Target 1%



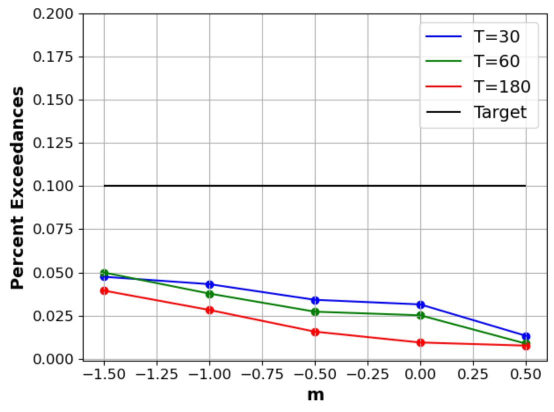
(j) Bates, Target 1%



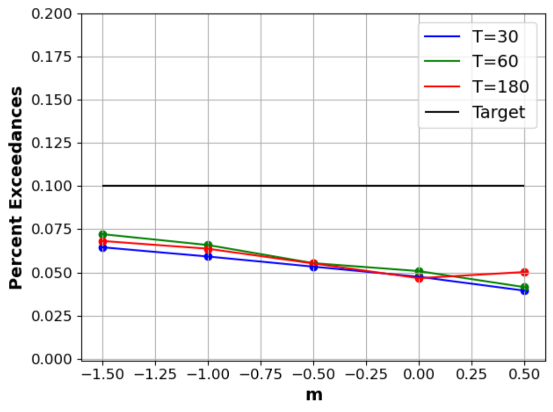
(k) HM, Target 5%



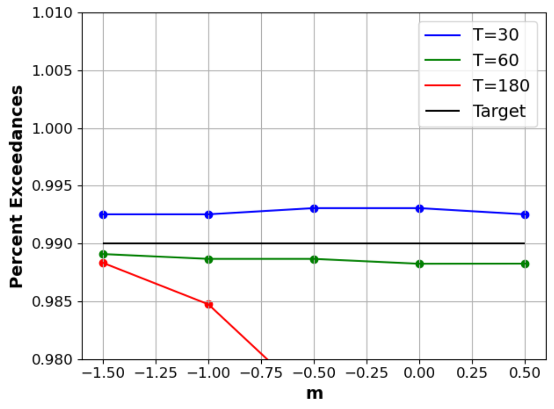
(l) Bates, Target 5%



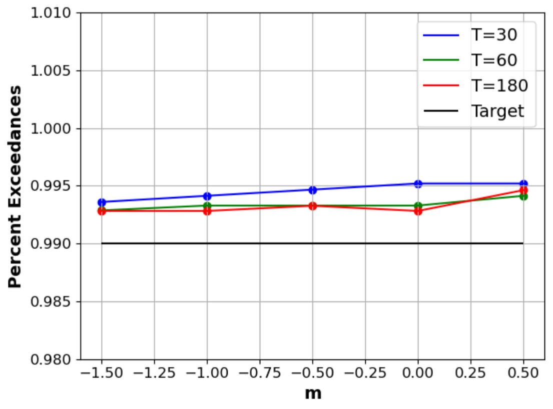
(m) HM, Target 10%



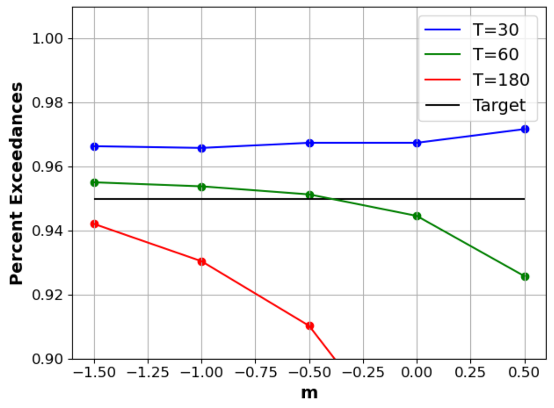
(n) Bates, Target 10%



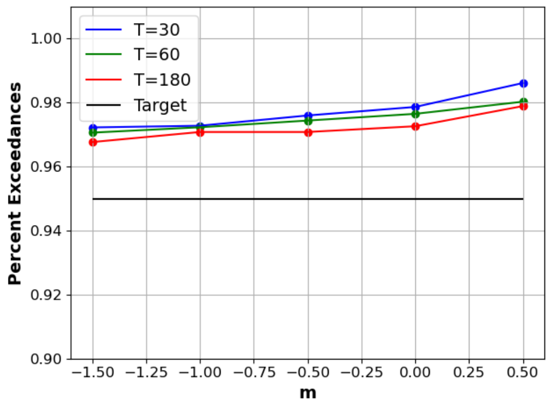
(o) HM, Target 99%



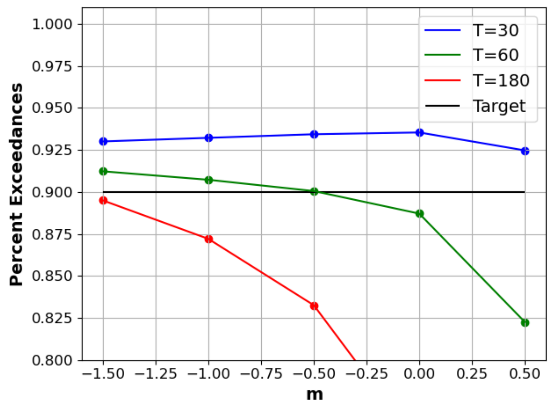
(p) Bates, Target 99%



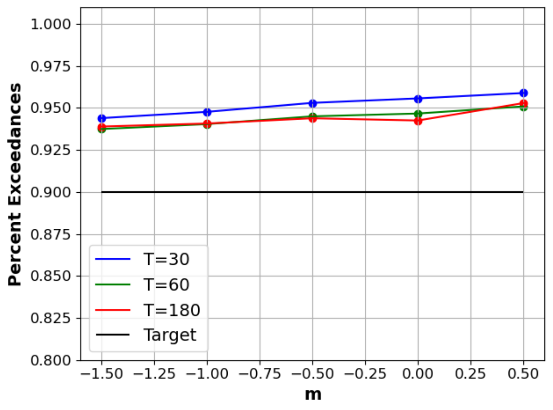
(q) HM, Target 95%



(r) Bates, Target 95%



(s) HM, Target 90%



(t) Bates, Target 90%

Figure A.3: The figures above show the performance of the HM (left panels) and Bates (right panels) to estimate the empirical distribution of option prices at time $t + 1$ for upper and lower quantiles. On the y-axis, we display the percent exceedance of both models as well as the target quantile. On the x-axis, we show the moneyness of the option brackets as defined by Eq. (18).

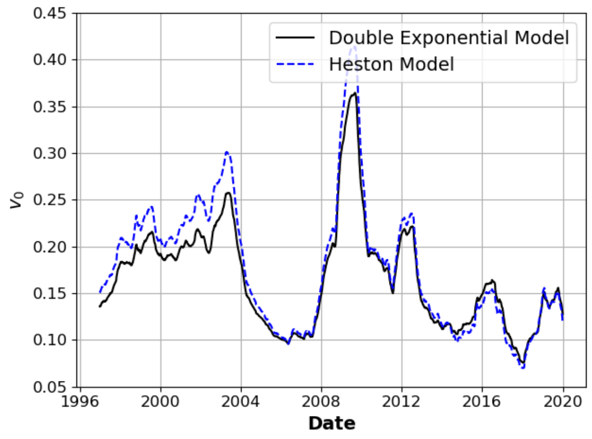
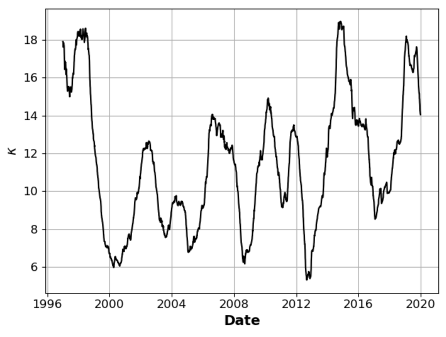
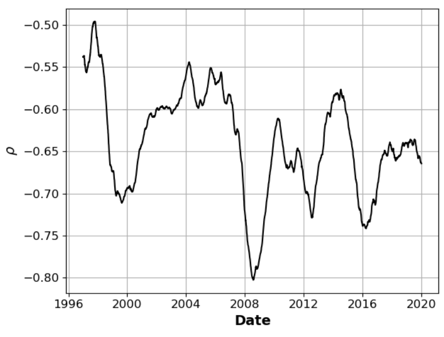


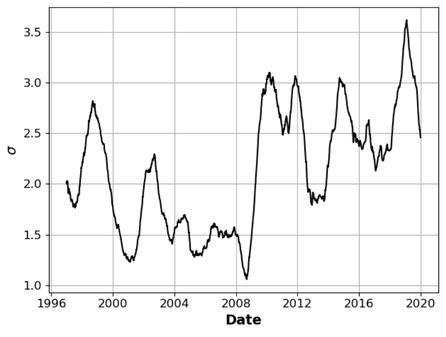
Figure A.4: The figure above depicts the value of the hidden volatility state v_t estimated every day along with each model's parameter. We smoothed the values with a rolling mean on the last 252 days.



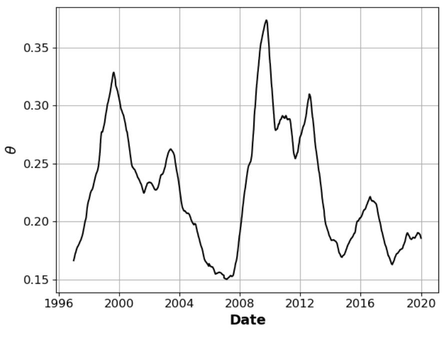
(a) κ



(b) ρ

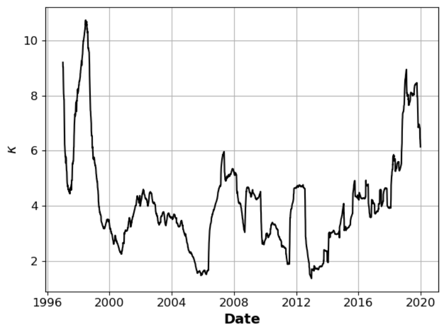


(c) σ

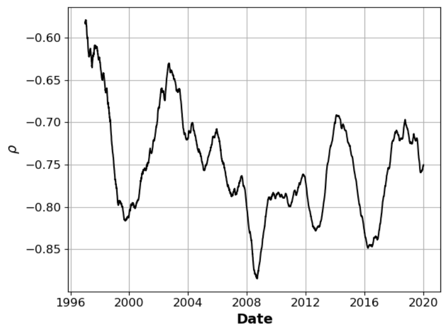


(d) θ

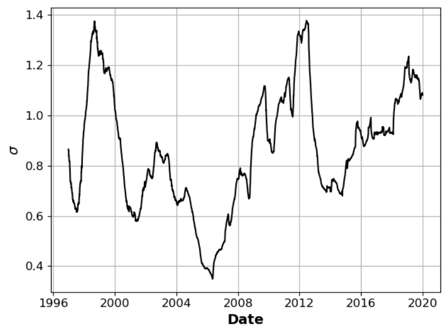
Figure A.5: The figures above show the daily estimation parameters of the HM. We smoothed the values with a rolling mean on the last 252 days.



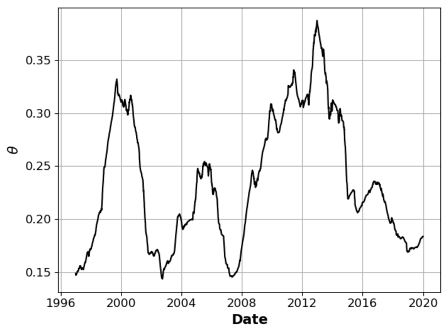
(a) κ



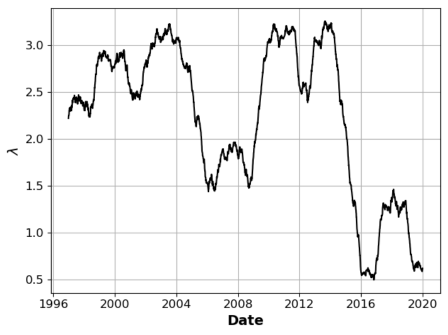
(b) ρ



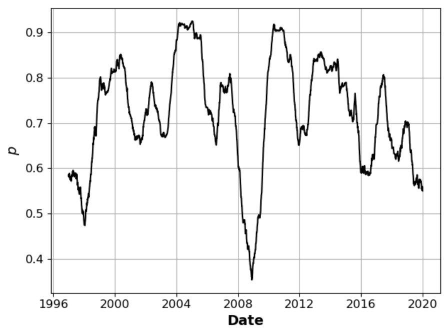
(c) σ



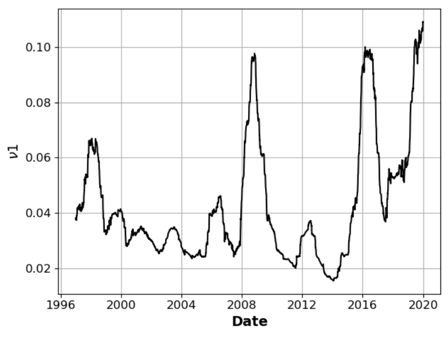
(d) θ



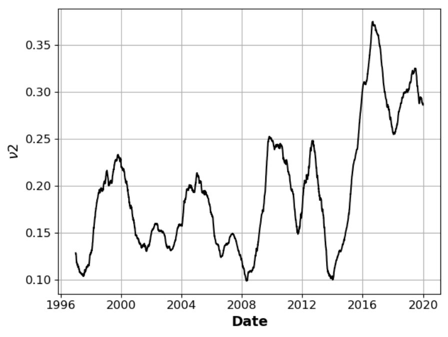
(e) λ



(f) p



(g) ν_u



(h) ν_d

Figure A.6: The figures above show the daily estimation parameters of the Bates. We smoothed the values with a rolling mean on the last 252 days.

D Additional analysis

Table A.2: This table expands on table 9 with additional percentiles of the replication error for each model.

	BSM	RF	HM	Bates
mean	0.19380	0.20133	0.15501	0.14175
std	0.42104	0.32173	0.39458	0.28051
min	0.00000	0.00000	0.00000	0.00000
5%	0.00423	0.00754	0.00601	0.00595
10%	0.01083	0.01878	0.01225	0.01220
15%	0.01747	0.02963	0.01881	0.01879
20%	0.02453	0.04081	0.02570	0.02569
25%	0.03220	0.05255	0.03305	0.03299
30%	0.04073	0.06476	0.04085	0.04073
35%	0.05014	0.07787	0.04924	0.04908
40%	0.06080	0.09247	0.05831	0.05811
45%	0.07288	0.10846	0.06836	0.06807
50%	0.08686	0.12512	0.07958	0.07914
55%	0.10295	0.14439	0.09227	0.09162
60%	0.12183	0.16658	0.10681	0.10580
65%	0.14332	0.19171	0.12358	0.12206
70%	0.17039	0.22086	0.14330	0.14120
75%	0.20360	0.25554	0.16752	0.16450
80%	0.24989	0.29937	0.19887	0.19474
85%	0.31197	0.35633	0.24335	0.23784
90%	0.41471	0.43864	0.31658	0.30624
95%	0.65096	0.59113	0.48294	0.44869
max	32.97002	30.43577	64.09164	29.19978