Learning Exotic Derivatives Without Calibration

A proof of concept

G.Amici[†], F.Brina[†], M.Mezzetti[†], A.Peroni[†]
M.Bianchetti[‡], P.Rossi[o]

[o]Prometeia. S.p.a. and University of Bologna

[†]University of Bologna

[‡]Intesa Sanpaolo and University of Bologna

June 10^{th}, 2021

Index

Acknowledgments

Computer time provided by:

https://www.e4company.com/

E4 Computer Engineering creates hardware and software solutions for High Performance Computing, Cloud Computing (Private and Hybrid), Containerization, High Performance Data Analytics, Artificial Intelligence, Deep Learning and Virtualization.

https://www.hpc.cineca.it/services/iscra

CINECA, the Italian most powerful HPC center, twice a year will directly award in excess of 100 millions core hours, to ensure an adequate supply to scientists and engineers for HPC-related research.

Introduction

1. Pricing in mathematical finance is building a complex function from a model following some constraint.
2. Models are just a rough description of markets, we need to 'calibrate'
3. The complexity of the calibration procedure depends on the complexity of the underlying model deemed capable to describe the market situation.
4. The calibrated model is the bridge between market data and prices (for prices that are not readily available as market data)
5. Models are a safeguard against producing "non arbitrage" prices

Introduction

Disclaimer

1. No magic application to improve on pricing models, just a more convenient way to work with models
2. We deem successful the approach if we can reproduce the ability of the model to price a given market.

NN approach to pricing

Let's call $\Pi(\alpha, G(\mathbf{g}))$ the function pricing a position G with a model described collectively by the parameters α, and the position described by the parameters $\mathbf{g}$.

example

For a vanilla option, $\mathbf{g}$ would be the pair maturity and strike.

A standard ML exercise would call for:

- generate, according to some random rule, a set of parameters α_n, $\mathbf{g}_n$ $1 \leq n \leq N$.
- for each $(\alpha_n \mathbf{g}_n)$ compute the function pricing the position G, with the model described the parameter set α_n:

$$\Pi_n := \Pi(\alpha_n, G(\mathbf{g}_n)).$$

NN approach to pricing

Iterating the procedure described above, we can build a large matrix as:

Regressors		Target
Model Parameters	Contract Parameters	
α_1	$\mathbf{g}_1$	Π_1
α_2	$\mathbf{g}_2$	Π_2
	...	
α_N	$\mathbf{g}_N$	Π_N

Table: Sample DB for NN training

and use it to train a neural network that, if all goes well, will learn the map

$$\phi_{NN} : \alpha, \mathbf{g} \rightarrow \Pi(\alpha, G).$$

Once trained, the network will perform as an efficient black box capable to price the position G.

NN approach to pricing

In order to use the trained network as a pricing engine, we must provide the correct set of parameters α, according to the following procedure:

1. look at market data and select a suitable set of liquid assets,
2. calibrate the model in order to select the best possible set of model parameters $\overline{\alpha}$,
3. feed the parameter set $\overline{\alpha}$ to ϕ_{NN} in order to price the position G.

The Market Based NN approach

- This strategy pays the very high cost of the calibration phase, unless we deal with simple models where model parameters can be read off directly from market data.
- For more complex models the above strategy is untenable and something must be done to circumvent the calibration phase.
- Since market data determine model parameters, the NN should be capable to learn the correct model starting from market data.

The Market Map

The calibration procedure can be described as the search for a map ϕ such that

$$\phi : \quad \text{market data } m(\mathbf{g}) \rightarrow \text{model parameters } \boldsymbol{\alpha}.$$

Once established (accepted) that the map ϕ exists, we do not really need to discover it, given that we can absorb it into the performance of the network.
If we knew such a map we could build the 'Market Map' $\mathcal{M}$ as

$$\mathcal{M} = \Pi \circ \phi : (\mathbf{g}, \quad \mathbf{m}(\mathbf{g})\,) = \Pi : (\mathbf{g}, \boldsymbol{\alpha}) \rightarrow \quad \text{prices}.$$

Rather that training the NN to learn the pricing map Π, we do train it to learn the map $\mathcal{M}$.

The Market Based NN approach: Training

1. generate a large set of parameters α_n, and market positions $\mathbf{g}_n,\ 1 \leq n \leq N$
2. for each parameter compute the price of the position $\Pi_n = \Pi(\alpha_n, G(\mathbf{g}_n))$
3. for each parameter compute a set of market observable $\mathbf{m}_n$ consistent with the chosen model and the parameter set α_n
4. use the pseudo market data $\mathbf{m}_n$ as regressors instead of the model parameters α_n when you train your NN ψ_G,

The Market Based NN approach: Training

Use the DB

Regressors		Target
$\mathbf{m}_1$	$\mathbf{g}_1$	Π_1
$\mathbf{m}_2$	$\mathbf{g}_2$	Π_2
	...	
$\mathbf{m}_N$	$\mathbf{g}_N$	Π_N

Table

to learn the map

$$\psi_{NN} : \mathbf{m}, \mathbf{g} \rightarrow \Pi(\alpha, G(\mathbf{g}))$$

The Market Based NN approach: Predict

1. given a position G read off the parameters $\mathbf{g}$,
2. look at market data and select the corresponding set of market data $\mathbf{m}$,
3. feed the market data $\mathbf{m}$, and the position parameters $\mathbf{g}$ to ψ_{NN} in order to price the position G.

Heston Model: European Options

For the Heston model we have the following model parameters

$$\boldsymbol{\alpha} = \{\overline{\nu}, \theta, \kappa, \eta, \rho\}$$

Recipe for vanilla options

1. generate random values for $\mathbf{g}_n = \{T_n, \hat{K}_n\}$ and $\boldsymbol{\alpha}_n = \{\overline{\nu}_n, \theta_n, \kappa_n, \eta_n, \rho_n\}$,
2. for each set $\boldsymbol{\alpha}_n, \mathbf{g}_n$ use the model to compute the prices Π_n
3. select a set of strike $\overline{K}_1, \ldots, \overline{K}_J$ as representative of the smile
4. for each $\overline{K}_j$ use the model to compute the model price:

$$\Pi_j = \mathbb{E}[\,(\,M(\boldsymbol{\alpha}_n, T_n) - \overline{K}_j\,)^+\,],$$

5. invert the J prices Π_j to compute the implied volatility smile

$$\sigma_n(T_n, \overline{K}_j) \quad 1 \leq j \leq J$$

European Options: Volatility Smile

Train the NN with the data set:

Regressors			Target
T_1	$\hat{K}_1$	σ_1	Π_1
T_2	$\hat{K}_2$	σ_2	Π_2
	$\vdots$	$\vdots$	
T_N	$\hat{K}_N$	σ_N	Π_N

Table: Sample input data set for the Heston Model.

Empirically a good choice to identify the calibration map ϕ is obtained by selecting, for each T, the points:

$$\overline{\mathbf{K}} = \{.900, .925, .950, .975, 1.000, 1.025, 1.050, 1.075, 1.100\}.$$

European Options: Numerical Results

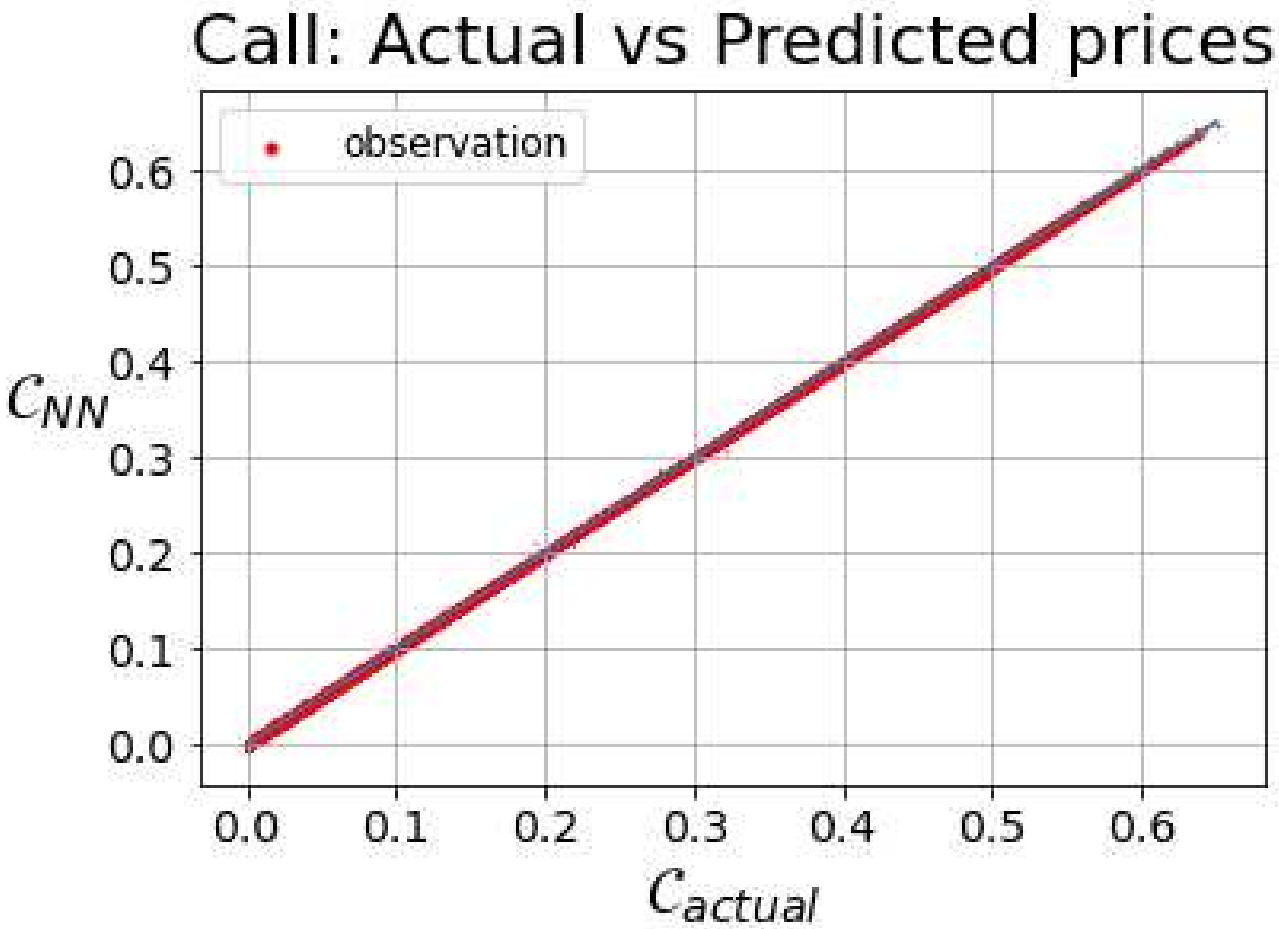


Figure: Scatter plot for the European call test set of 1 million entries. The NN predictions are compared with the actual prices.

European Options: Numerical Results

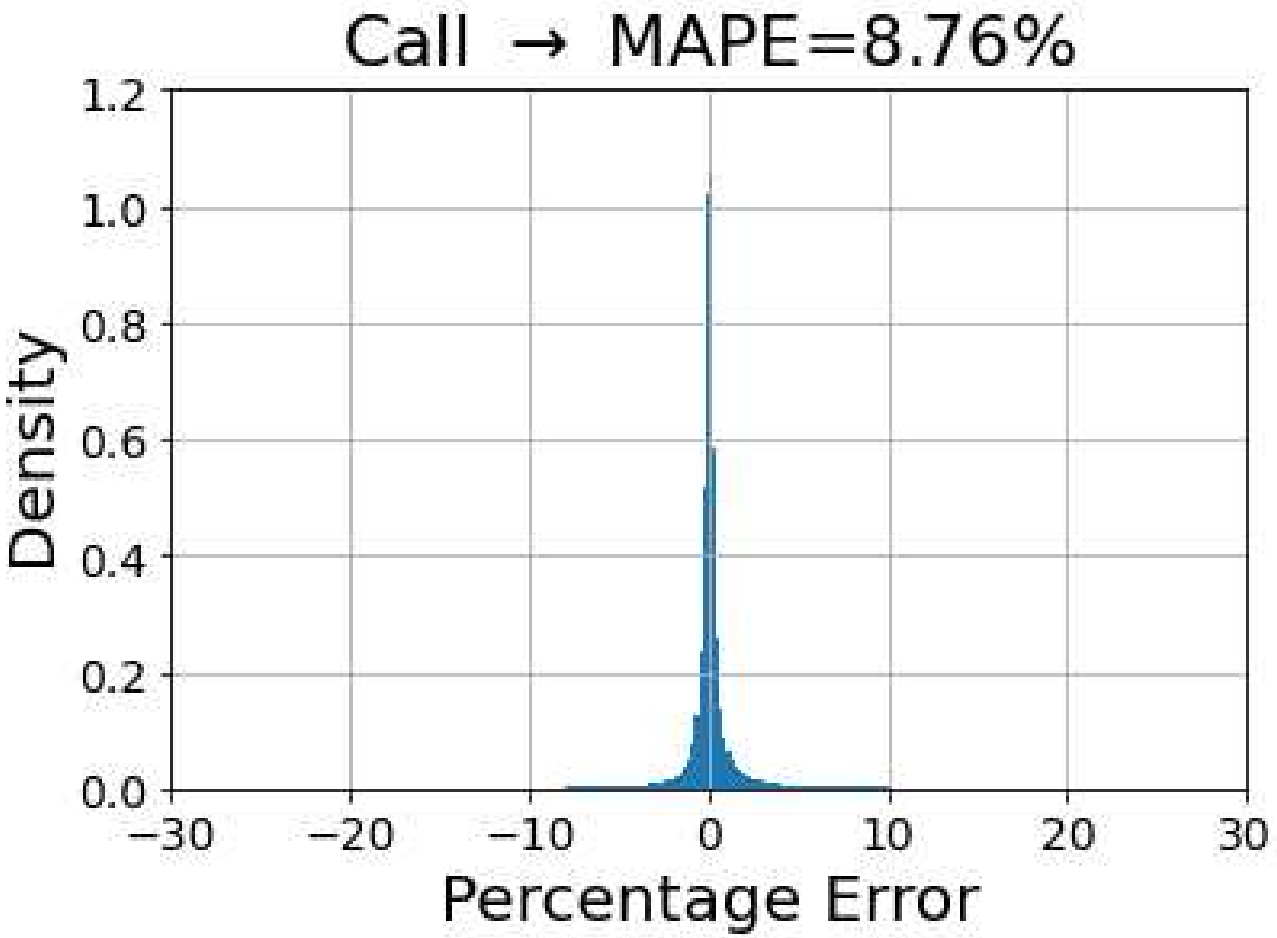


Figure: Percentage errors distribution for the NN predictions on the European call test set of 1 million entries. The title shows the MAPE (mean absolute percentage error).

European Options: Market test

1. Market test refer to the use of the trained NN when trying to replicate prices of S&P 500 of May 19^{th}, 2017.
2. The procedure used to feed the network with regressors, is the one described in these notes.
3. An interpolation was used to extract the smile at the appropriate maturity and strikes.

European Options: Market test

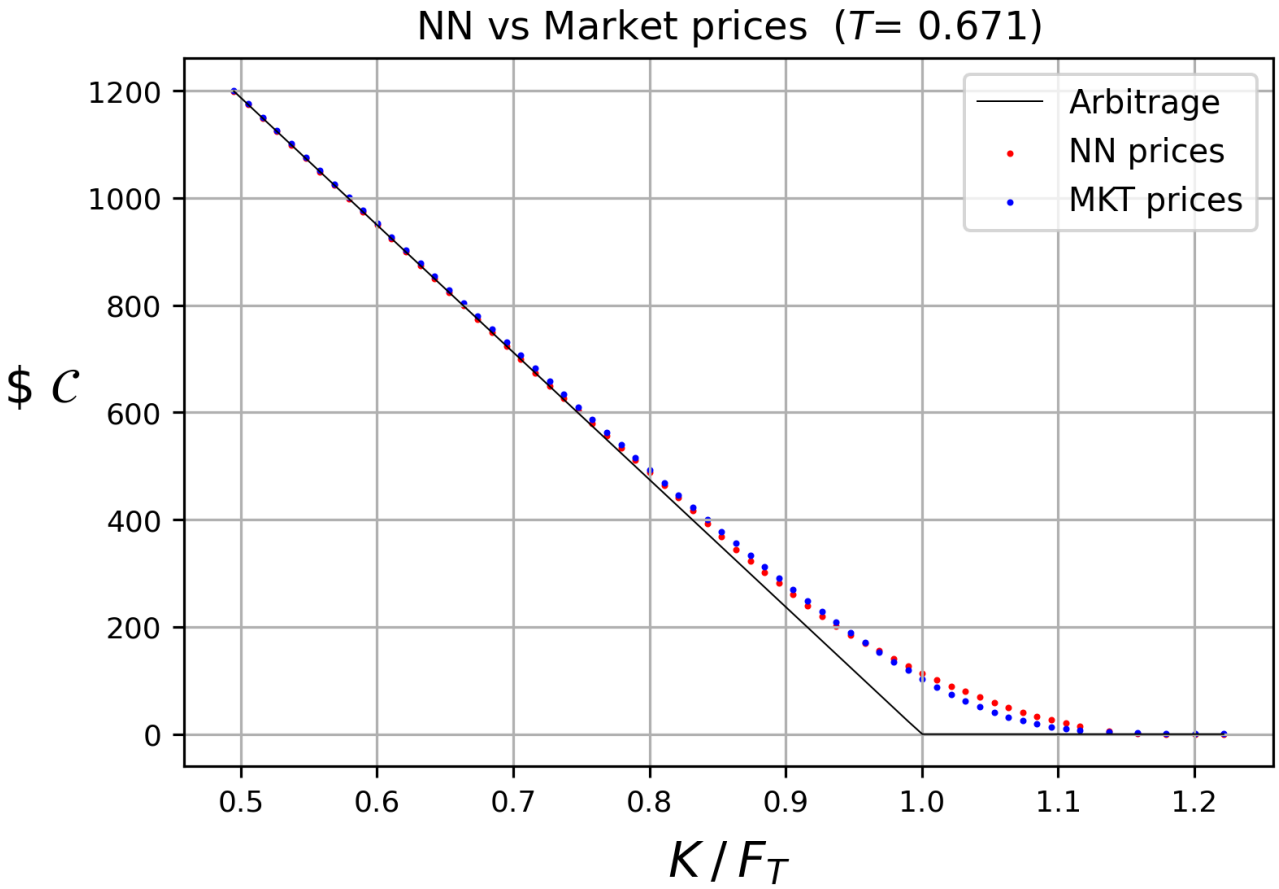


Figure: Comparison between the NN predictions and market prices quoted for maturity $T = 0.671$ years. The prices are plotted as a function of moneyness. The no-arbitrage threshold (intrinsic value) is also displayed.

European Options: Market test

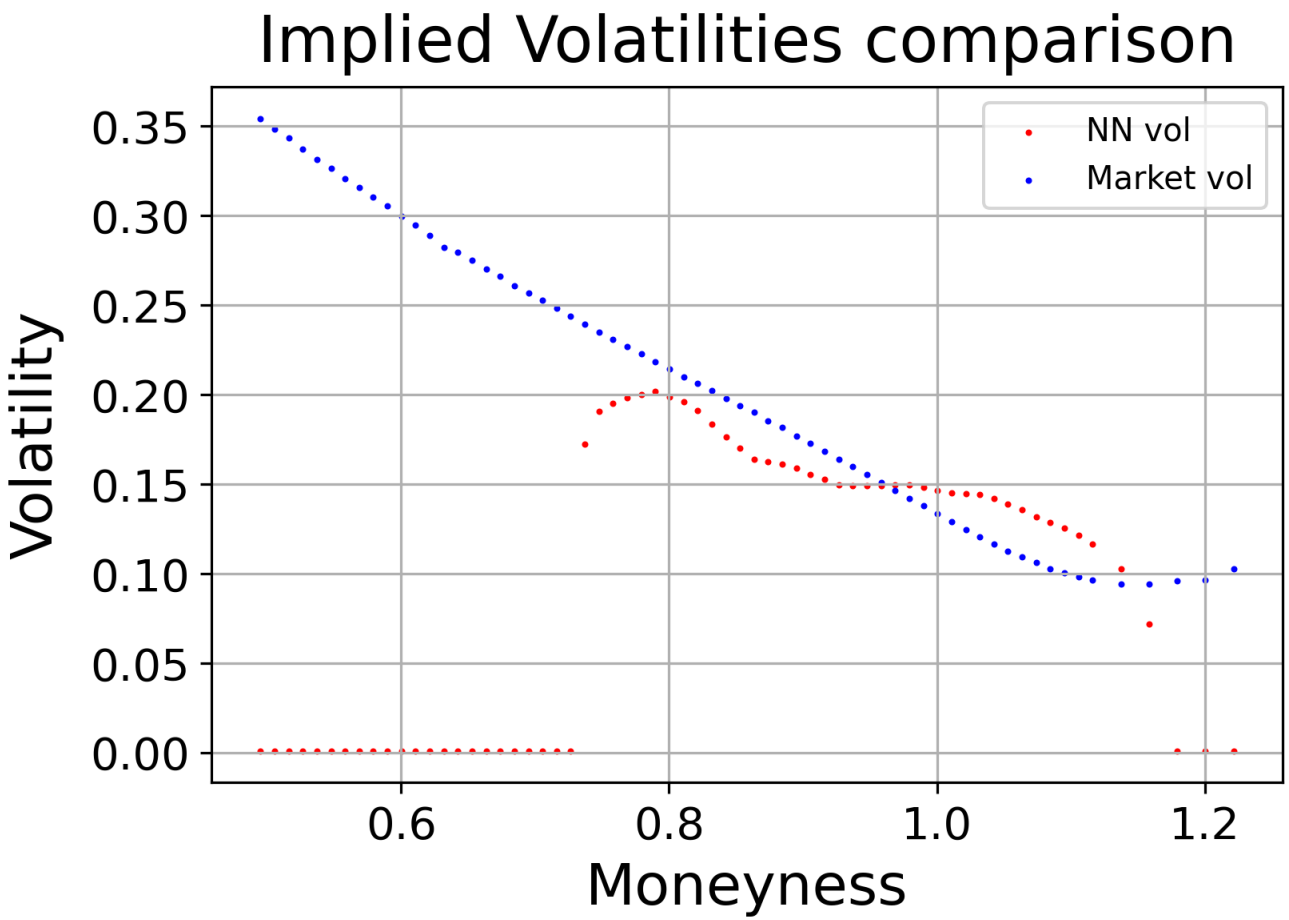


Figure: Comparison between the BS implied volatilities in NN prices and those implied in market prices quoted for maturity $T = 0.671$ years. Null implied volatilities correspond to NN prices too close to the intrinsic value.

European Options: Market test

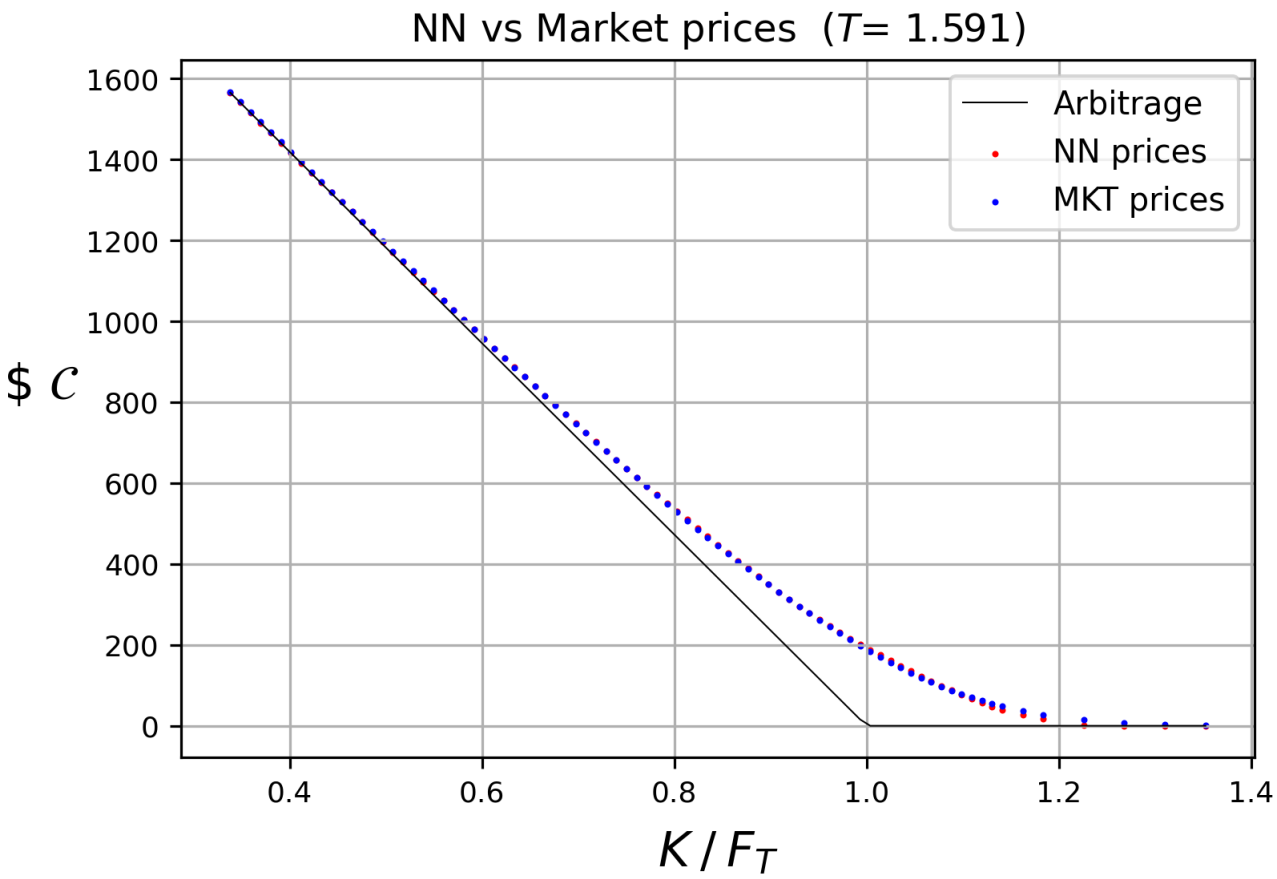


Figure: As before, for $T = 1.591$ years.

European Options: Market test

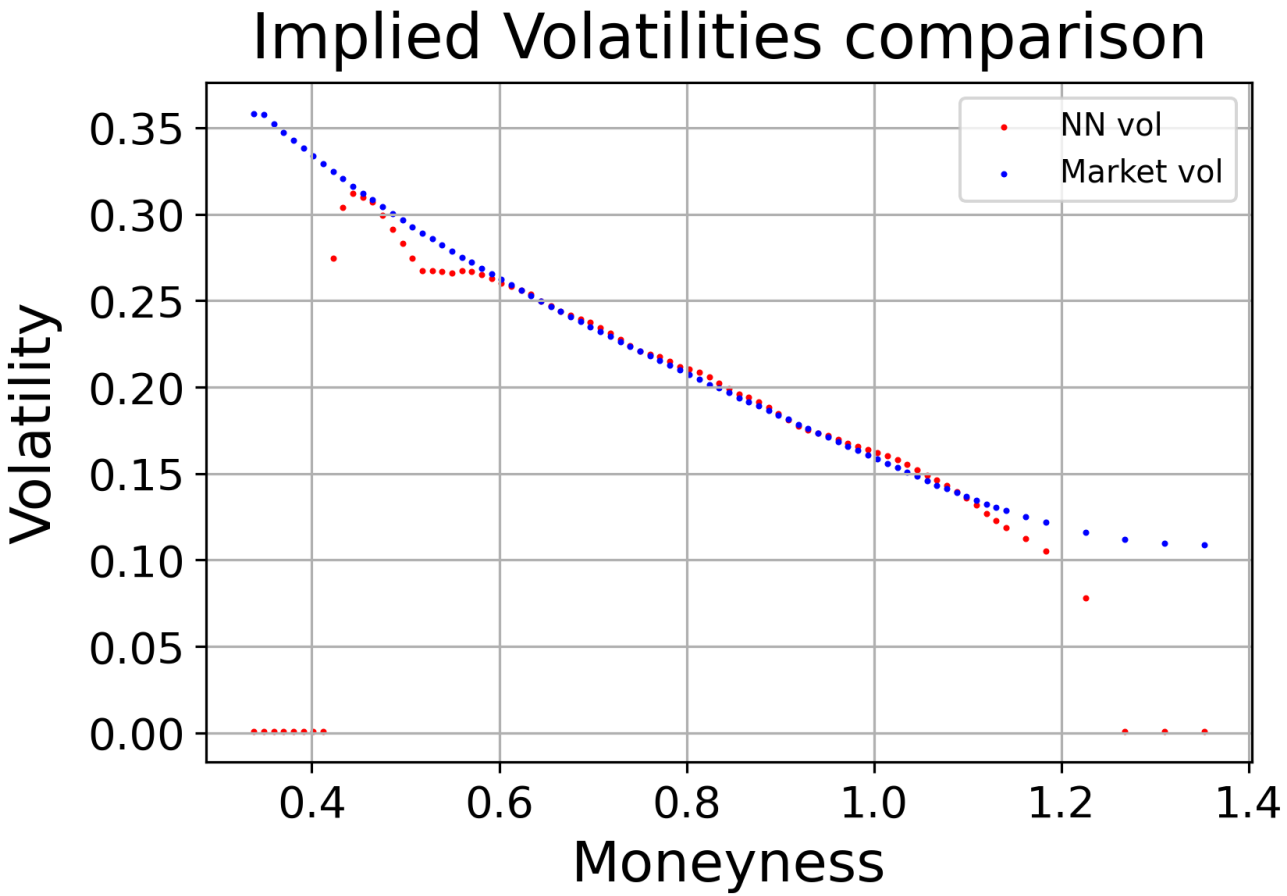


Figure: As before, for $T = 1.591$ years.

Knock Out Options

$$\text{Let} \quad S_{min}(T) := \min_{0 \le t \le T} S(t)$$

$$\text{The payoff is defined as:} \quad [\, S(T) - K \,]^{+} \mathbf{1}_{\{S_{min}(T) > B\}}$$

For Knock Out options we have two extra parameters with respect to the vanilla option: the barrier level and the interest rate.

Regressors					Target
r_1	B_1	T_1	$\hat{K}_1$	σ_1	Π_1
r_2	B_2	T_2	$\hat{K}_2$	σ_2	Π_2
		...			
r_N	B_N	T_N	$\hat{K}_N$	σ_N	Π_N

Table: Sample input data set for the Heston Model

where $\hat{K} = \frac{K}{S_0}$, $q = 0$, and $B_1, \ldots, B_N$ are barrier levels.

Knock Out Options: Results

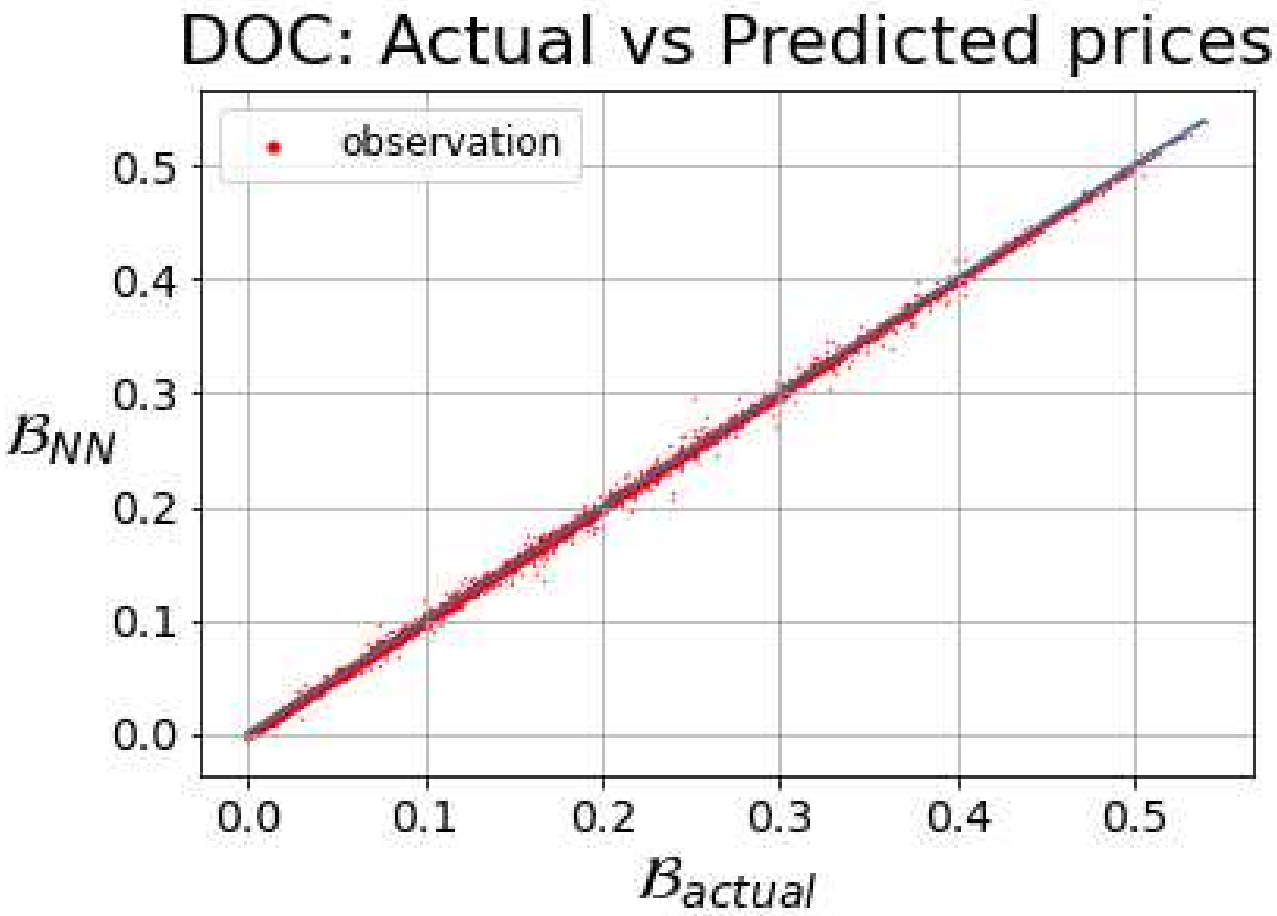


Figure: Scatter plot for the DOC (down-out-call) test set (10,000 entries). NN prices against FD prices.

Knock Out Options: Results

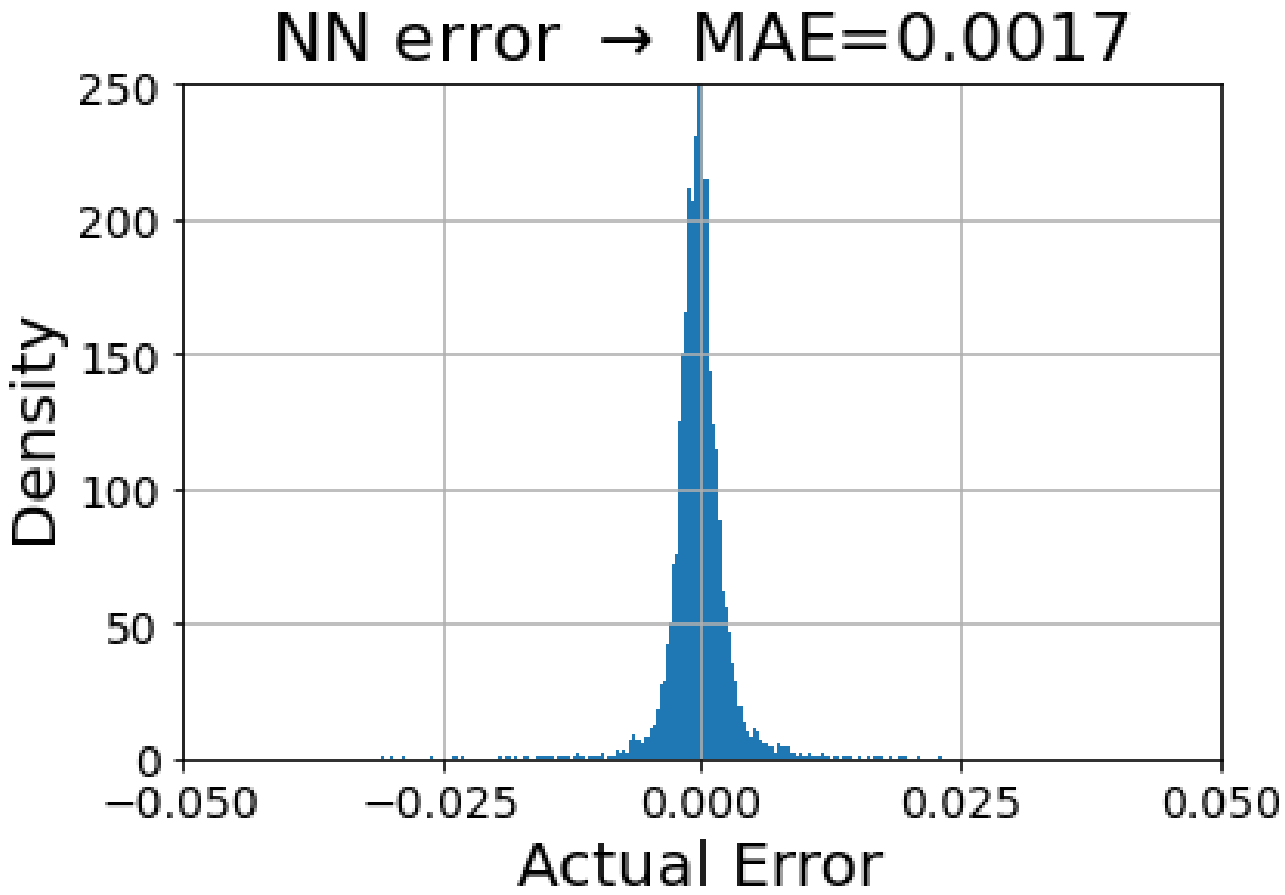


Figure: Errors distribution for the DOC (down-out-call) test set (10,000 entries). The title shows the MAE (mean absolute error).

Knock Out Options: Results

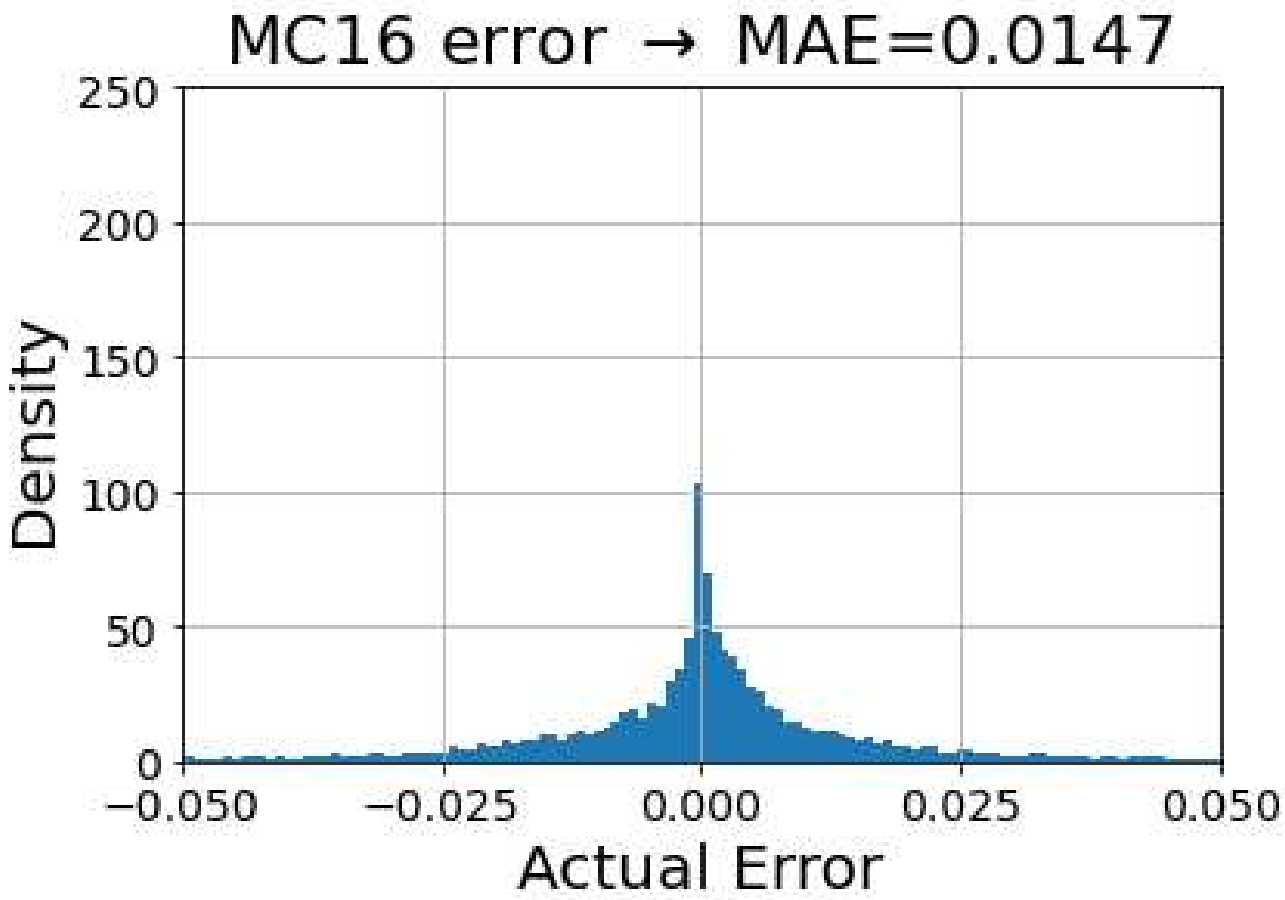


Figure: Errors distribution for the DOC (down-out-call) test set (10,000 entries) of Monte Carlo prices computed with $2^{16} = 65536$ MC scenarios for underlying stock price. The title shows the MAE (mean absolute error).

Knock Out Options: Results

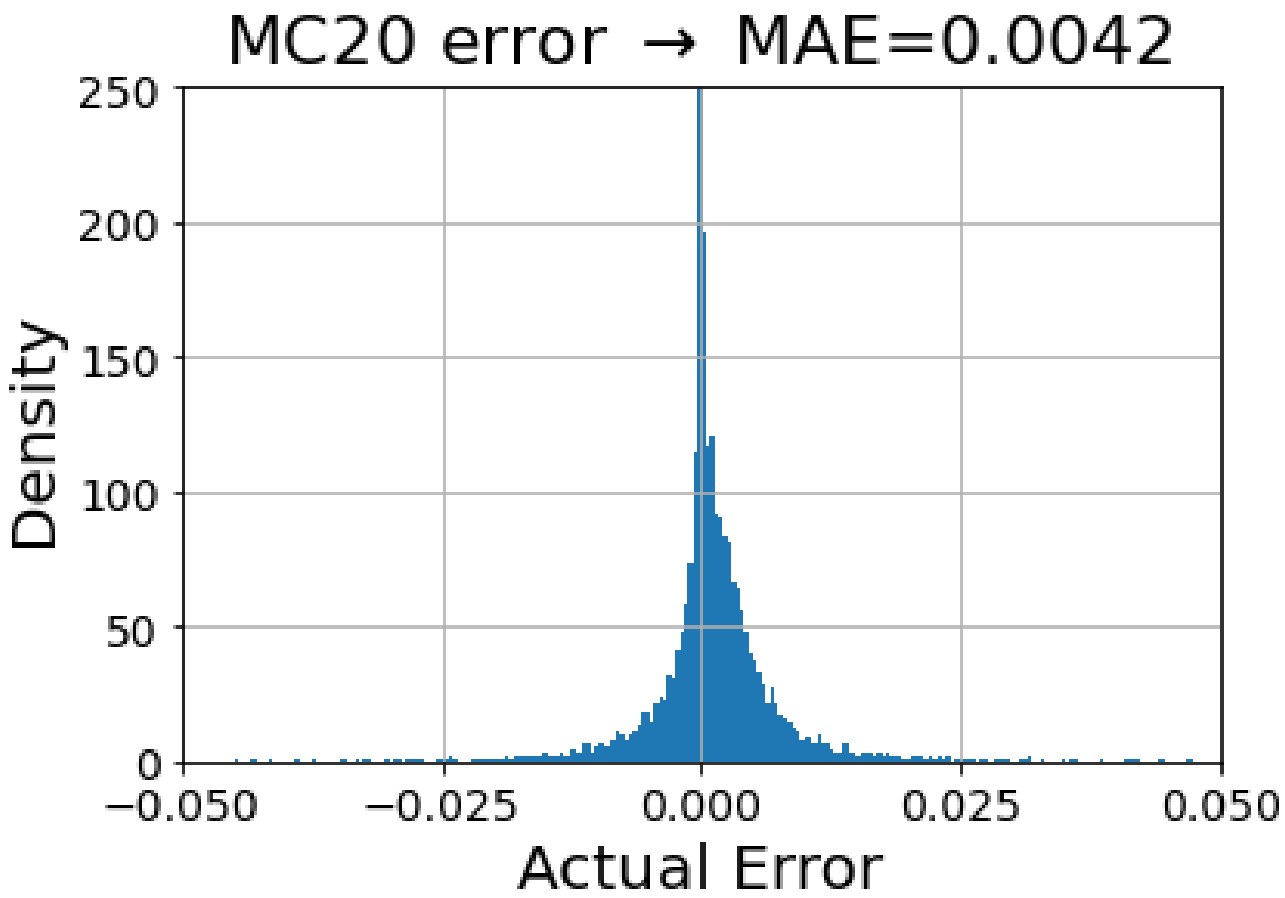


Figure: Errors distribution for the DOC (down-out-call) test set (10,000 entries) of Monte Carlo prices computed with $2^{20} = 1048576$ MC scenarios for underlying stock price. The title shows the MAE (mean absolute error).

Double Knock Out Options

For Double Knock Out options we have two extra parameters with respect to the vanilla option: the low and the high barriers.

Regressors				Target	
B_1^l	B_1^h	T_1	$\hat{K}_1$	σ_1	Π_1
B_2^l	B_2^h	T_2	$\hat{K}_2$	σ_2	Π_2
		...			
B_N^l	B_N^h	T_N	$\hat{K}_N$	σ_N	Π_N

Table: Sample input data set for the Heston Model

- $\hat{K} = \frac{K}{S_0}$
- $B_1^l, \ldots, B_N^l$ are low barriers
- $B_1^h, \ldots, B_N^h$ are high barriers.

Double Knock Out Options: Results

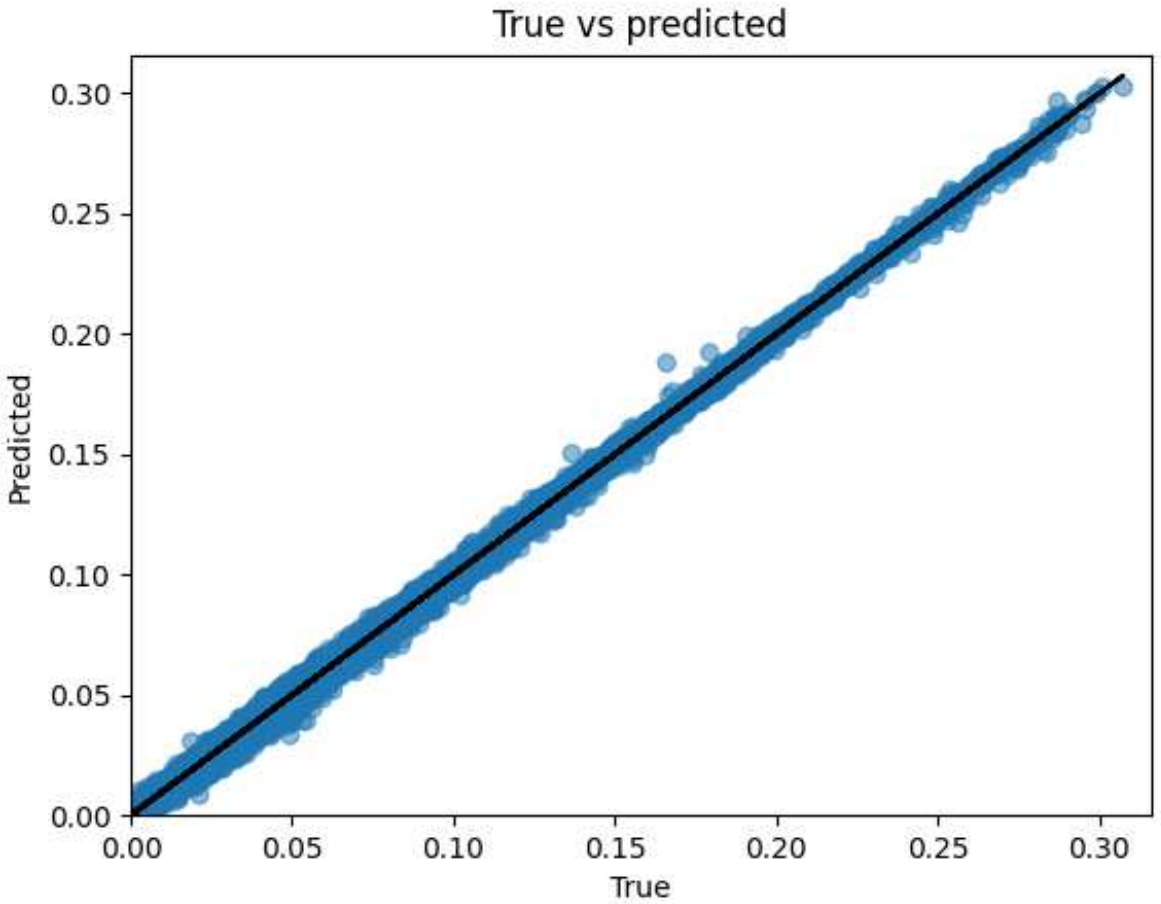


Figure: Scatter plot for the Double Knock Out test set. NN prices against FD prices.

The Hull+White Model

The Hull+White model is quite convenient as a playground given that

1. we can test out ideas without having to resort to expensive MC simulation to produce the 'correct result' (model result).
2. Within the H+W model we have closed form solution both for Caps and Swaptions.

The dynamic, in the bank account numeraire, of the H+W model , is given by the SDE

$$dr(t) = (\theta(t) - \gamma r(t))dt + \sigma dW_t, r(0) = r_o$$

The deterministic function $\theta(t)$ is chosen to match todays observed curve

Model Generation

We have calibrated the NS parameters on a real Euribor 6 months curve,

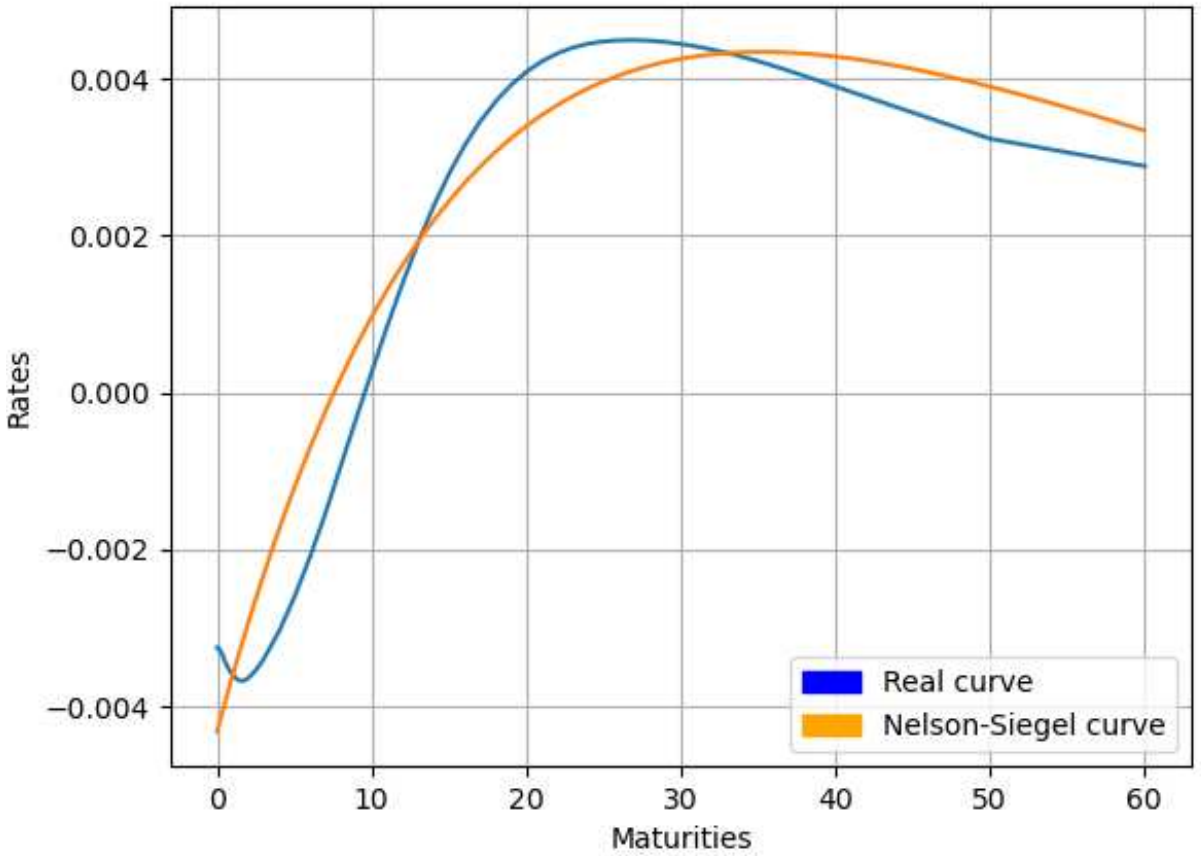


Figure: Nelson-Siegel curves vs original Euribor 6 month curve

Model Generation

. . . next we have sampled NS parameters from a normal distribution centered around the calibrated parameters.

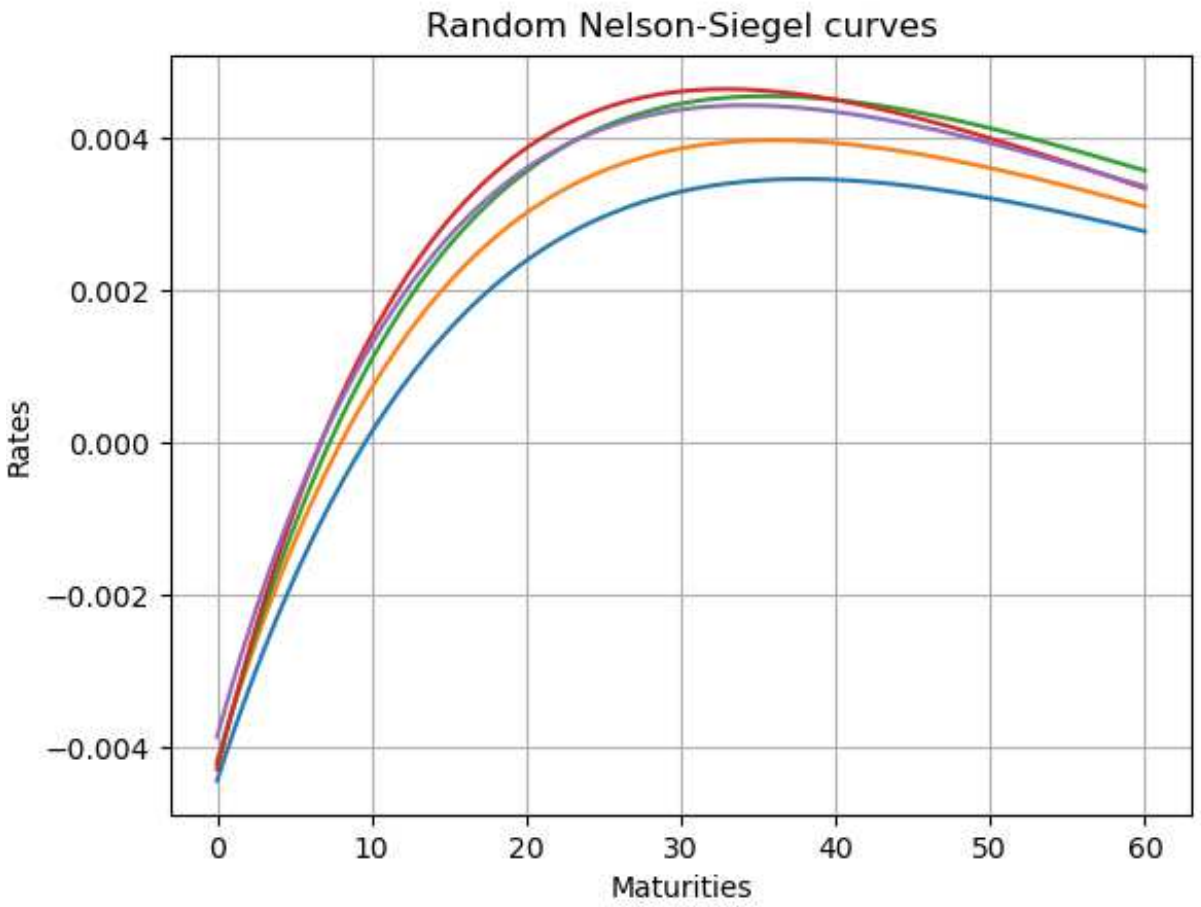


Figure: A sample of random curves generated by Nelson-Siegel parameters

Regressors and target. Summary

In the language adopted so far we have

$$\boldsymbol{\alpha} = \{\gamma, \sigma, \ \beta_0, \beta_1, \beta_2, \tau\}$$

- Building realistic discount curves is extremely important, given that this curve controls almost all of the regressors and the target price.
- The NS parameters will be replaced, as regressors, by suitably selected interest rates
- model parameters are replaced by the volatility smile

Contract parameters: Maturity and Tenor

Contract parameters $\mathbf{g}$ are:

- Maturity T_n: U ([0, 30]), we have sampled uniformly distributed maturities, inside the range [0, 30];
- Tenor N_{np}: for each maturity we have trained the NN over quoted tenors:

$$1y, 2y, 3y, 4y, 5y, 6y, 7y, 8y, 9y, 10y, 15y, 20y, 25y, 30y$$

- Strike κ_n: U ([ATM - 400 bps, ATM + 400 bps]),

Contract parameters: Strikes

To generate random strikes, we have set up a procedure that goes like this: for each row n of the training set, given:

$$\{\text{IR\_curve}_n, \gamma_n, \sigma_n, \quad T_n, \quad N_{np}\}$$

compute the Swap Rate $S_p(t_n)$ corresponding to the time schedule:

$$t_m, t_m + \delta t, \ldots, t_m + p\delta t$$

given $S_p(t_n)$, sample uniformly distributed strikes within the range:

$$S_p(t_n) \pm 400 \text{ bps}$$

Swaptions Curve's representative Rates

As market data, needed to replace the NS paramters, we selected some of the most liquid rates in the market.

- Spot and forward Euribor rates:

$$R_{\text{Spot}}(0, 3m) \qquad R_{\text{fwd}}(3m, 6m) \qquad R_{\text{fwd}}(6m, 1y)$$

- Spot Swap Rates for 6-months payment schedule from 1 to 30 years:

$$S_{2p}(t_0), \quad p = 2, 5, 10, 15, 20, 25, 30, 40, 50, 60$$

Market Parameters

The dynamical parameters of the model are replaced by the volatility smile:

- **Shift**=5%; necessary for working with negative rates with log-normal dynamics;
- **Strikes**: the smile section has been computed on these set of strikes, expressed as difference in basis point with respect to the ATM:

$$\text{Strikes} = \left\{ \begin{array}{ccc} \text{ATM} - 150\text{bps} & \text{ATM} - 100\text{bps} & \text{ATM} - 50\text{bps} \\ \text{ATM} - 25\text{bps} & \text{ATM} & \text{ATM} + 25\text{bps} \\ \text{ATM} + 50\text{bps} & \text{ATM} + 100\text{bps} & \text{ATM} + 150\text{bps} \end{array} \right\}$$

Network Topology

- 6 Hidden Layers, Dense structure: all the nodes of layer K, are connected to nodes of layer K $\pm$ 1;
- 600 nodes per layer;

Numerical results

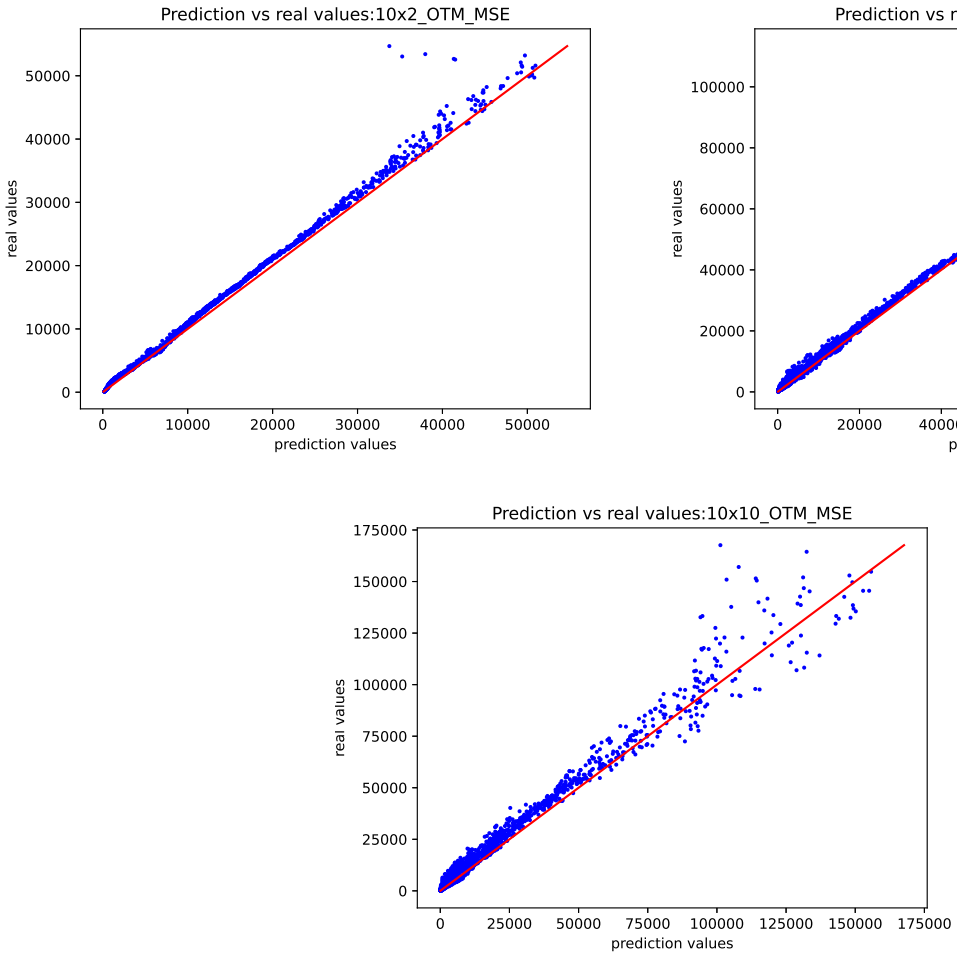


Figure: Swaption: scatter plot on the test set for OTM payer 10 X 2, 10 X 5, and 10 X 10 Swaptions. Strike is ATM + 150bps.

Numerical results

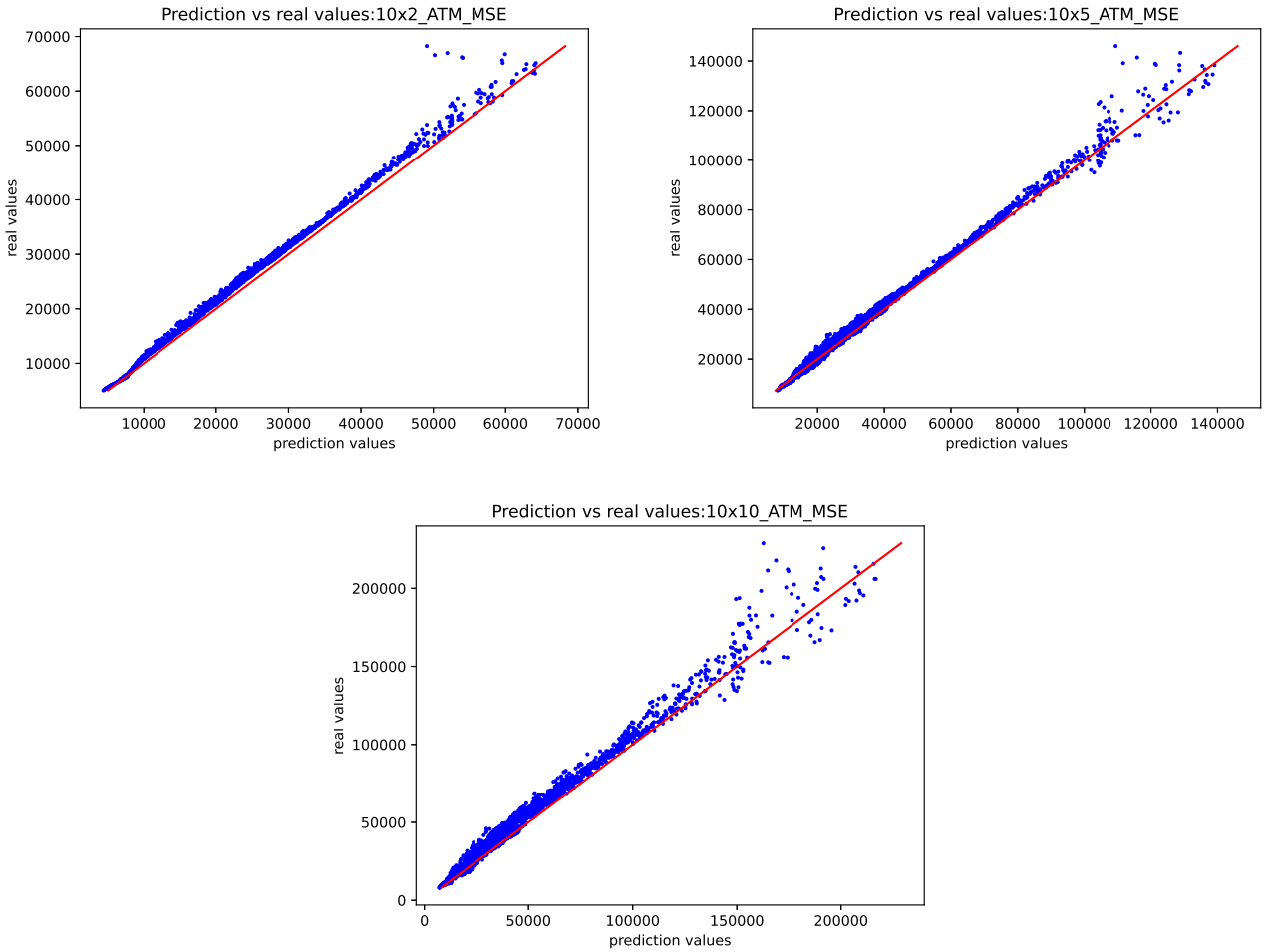


Figure: Swaption: scatter plot on the test set for the 10 X 2, 10 X 5, and 10 X 10 Swaptions. Strike is ATM

Numerical results

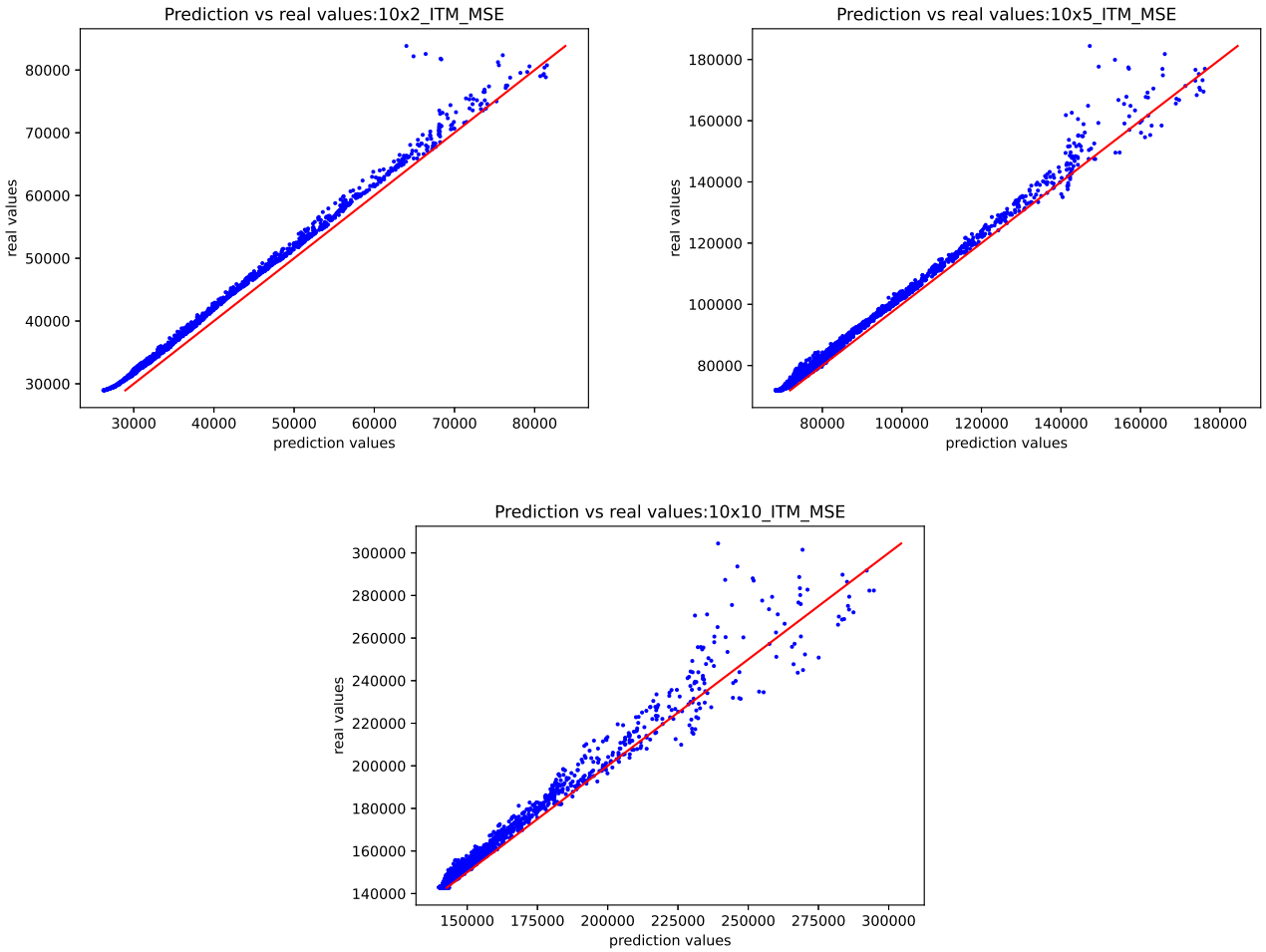
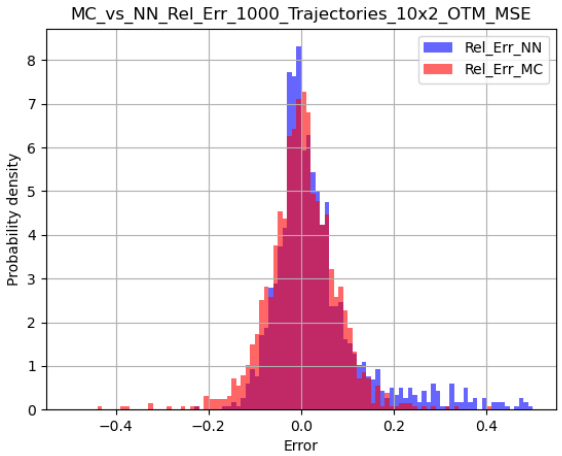
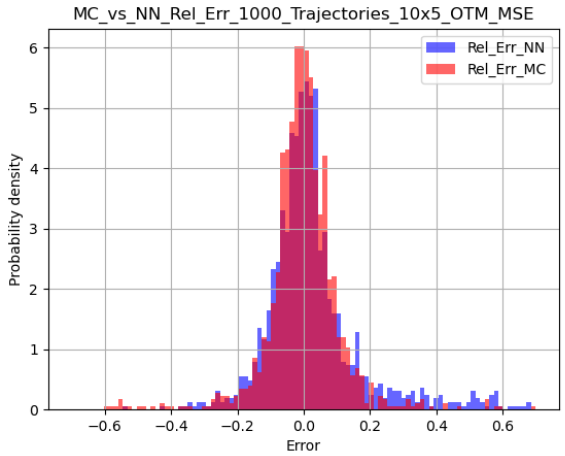


Figure: Swaption: scatter plot on the test set for 10 X 2, 10 X 5, and 10 X 10 Swaptions. Strike is ATM - 150 bps

Numerical results. MC vs NN





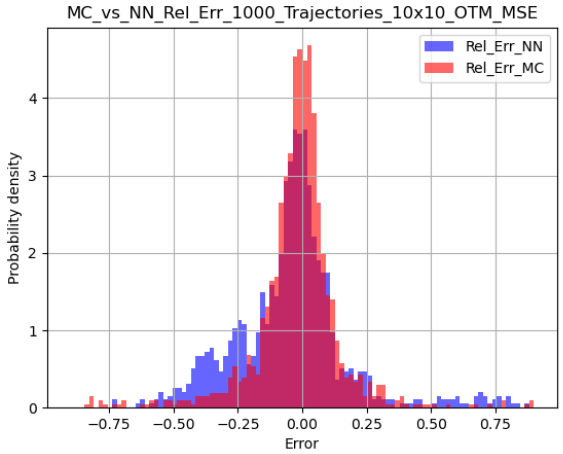


Figure: Swaption: comparison of the error distribution for MC and NN for different OTM Swaptions with 10 years maturity

Conclusions

From the results presented so far we may summarize the following conclusions.

1. We established a pricing framework where stochastic dynamics are used to generate large, consistent, and arbitrage-free sets including both market data and derivatives prices, which, in turns, are used to train NNs to price the same instruments.
2. We tested different dynamics (Black-Scholes, Heston, Hull-White), and we showed that our NNs, once properly trained, can price both plain vanillas and some exotics with good precision and absorbing negligible computational resources.
3. Our framework is both flexible with respect to the introduction of other and more complex stochastic dynamics and exotic derivatives, and robust with respect to the tuning of the NNs' hyper-parameters. It allows, in principle, to price any derivative, overcoming both the scarcity of observable market prices and the computational bottlenecks due to model calibration and numerical pricing algorithms, thus constituting a promising alternative for efficient risk management of portfolios of financial instruments.

Future Work

Our approach can be extended in a number of directions.

1. The Heston model provides volatility surfaces not much skewed, and actually quite flat in many cases; a better choice to investigate could be the stochastic volatility model with jumps suggested by (**?**). Analogously, for interest rates, one could explore better dynamics, e.g. the two-factors Hull-White model (see e.g. **?**), or the Generalized Forward Market Model recently proposed by (**?**) and (**?**).
2. Data sets could be extended both with more data, using wider/finer ranges of model parameters' values, and more payoffs, e.g. including digital, path dependent, and early exercise features.

Future Work (cont'd)

3. Greeks could be introduced in the framework both using automatic adjoint differentiation (AAD) as suggested by (**?**), which requires greeks as regressor of the NN, or using polynomial interpolation as suggested by (**?**), which does not require greeks as regressors.

4. Stress test: we could use our NN, without any further training, to compute prices on stressed market scenarios (provided that the stressed scenario is within the training domain).