Your Neural Network is Your Net Worth

Valuation and Hedging of Interest Rate Derivatives with Automatic Adjoint Differentiation and Differential Machine Learning

Nicklas Kenno Hansen

MSc. Mathematics-Economics
University of Copenhagen

12th of January 2024

Table of Contents

1. Scope and Concepts
2. Market Description
3. Monte Carlo Engine
4. Automatic Differentiation
5. Differential Machine Learning
   - Differential Regression
   - Differential Neural Network
6. General Multi-Factor Stochastic Volatility Model by Trolle & Schwartz
7. Conclusion and Further Research

The Scope in Brief

- Develop a flexible and scalable Monte Carlo (MC) engine instrumented with Automatic Adjoint Differentiation (AAD) capable of performing valuation and risk calculations for (virtually) arbitrary products and models.
- Utilize the engine to train Differential Machine Learning (DiffML) models "live" with.

The Concepts in Brief:

- Monte Carlo provides pathwise samples.
- AAD gives *analytical derivatives* in constant time and is useful for estimating hedge-coefficients (Greeks).
- DiffML trains faster and more accurate than classical ML through the additional differential labels which act as regularization without bias.

Linear Products & Rates

Zero Coupon Bond $P(t,T) = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\right] = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-\int_t^T r(s)ds}\right]$

Simple Forward Rate $F(t,T,T+\delta) = \frac{1}{\delta}\left(\frac{P(t,T)}{P(t,T+\delta)} - 1\right)$

Simple Spot Rate $F(T,T+\delta) = F(T;T,T+\delta) = \frac{1}{\delta}\left(\frac{1}{P(t,T+\delta)} - 1\right)$

Swap Rate $S(t,T_0,T_n) = \frac{P(t,T_0)-P(t,T_n)}{\delta\sum_{i=1}^n P(t,T_i)} = K$

Payer Swap $\Pi_p(t) = N\left[P(t,T_0) - P(t,T_n) - K\delta\sum_{i=1}^n P(t,T_i)\right]$

Interest Rate Derivatives

European Caplet $Cpl(t;T,T+\delta) = \frac{1}{\tilde{K}}\mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\left(\tilde{K} - P(T,T+\delta)\right)^+\right]$

European Swaption $V_p^{EU}(t) = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\left(\Pi_p(T)\right)^+\right]$

Barrier Swaption $V_p^{Barr}(t) = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}(\Pi_p(T))^+ \mathbb{1}\left\{\Pi_p(t) < B, \forall t \in \{\tau\}\right\}\right]$

Bermudan Swaption $V_p^{Ber}(t) = \sup_{\tau^* \in \{\tau\}} \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(\tau^*)}(\Pi_p(\tau^*))^+\right]$

Monte Carlo Engine

We consider a **product** defined as a collection of **cashflows** contingent on the market state at predetermined *event dates*

$$\{CF(t_i),\ 0 \leq t_i \leq T\}\,.$$

Risk Neutral Pricing Theorem

The price of a product is the risk-neutral expected value of its numeraire discounted cashflows,

$$V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\sum_{t\leq t_i\leq T} \frac{B(t)}{B(t_i)} CF(t_i;\boldsymbol{x}_{t_i})\right] = \mathbb{E}_t^{\mathbb{Q}}\left[g(\boldsymbol{x})\right] = \int_A g(\boldsymbol{x}(\omega))\, d\mathbb{Q}(\omega),\ \forall A\in\mathcal{F}_t.$$

Monte Carlo Estimator

The Monte Carlo Estimator for $N \in \mathbb{N}$ i.i.d. samples (scenarios) is the given be the average of the sampled realisations

$$V = \int_\Omega g(\boldsymbol{x}(\omega))d\mathbb{Q}(\omega) \approx \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}_i) =: \hat{V}, \qquad \lim_{N\to\infty} \hat{V} = V.$$

The **model** samples the state variables X and maps these into market observables $\boldsymbol{x}$ on the products event dates (timeline, TL)

$$\zeta : \{X_t, 0 \leq t \leq T\} \rightarrow \{\boldsymbol{x}_t, t \in TL\}.$$

In practice this is done by simulating SDEs

$$dX(t) = \mu(t, X(t))dt + \sigma(t, X(t))dW(t), \quad 0 \leq t \leq T.$$

The most straight forward schema is the explicit *Euler–Maruyama method*

$$X^i(t_{k+1}) = X^i(t_k) + \mu^i(t_k, X_{t_k})\Delta t_k + \sum_{j=1}^{m} \sigma^{ij}(t_k, X_{t_k})\sqrt{\Delta t_k} Z_k^j, \quad k = 0, ..., M-1.$$

Segregated Responsibilities

- The **product** is responsible for determining *when* and *what* is sampled in each scenario generated.
- The **model** is in turn responsible for *how* the state and market variables are generated on the product's event dates in each scenario from the sample space. I.e. determines the probability distributions.

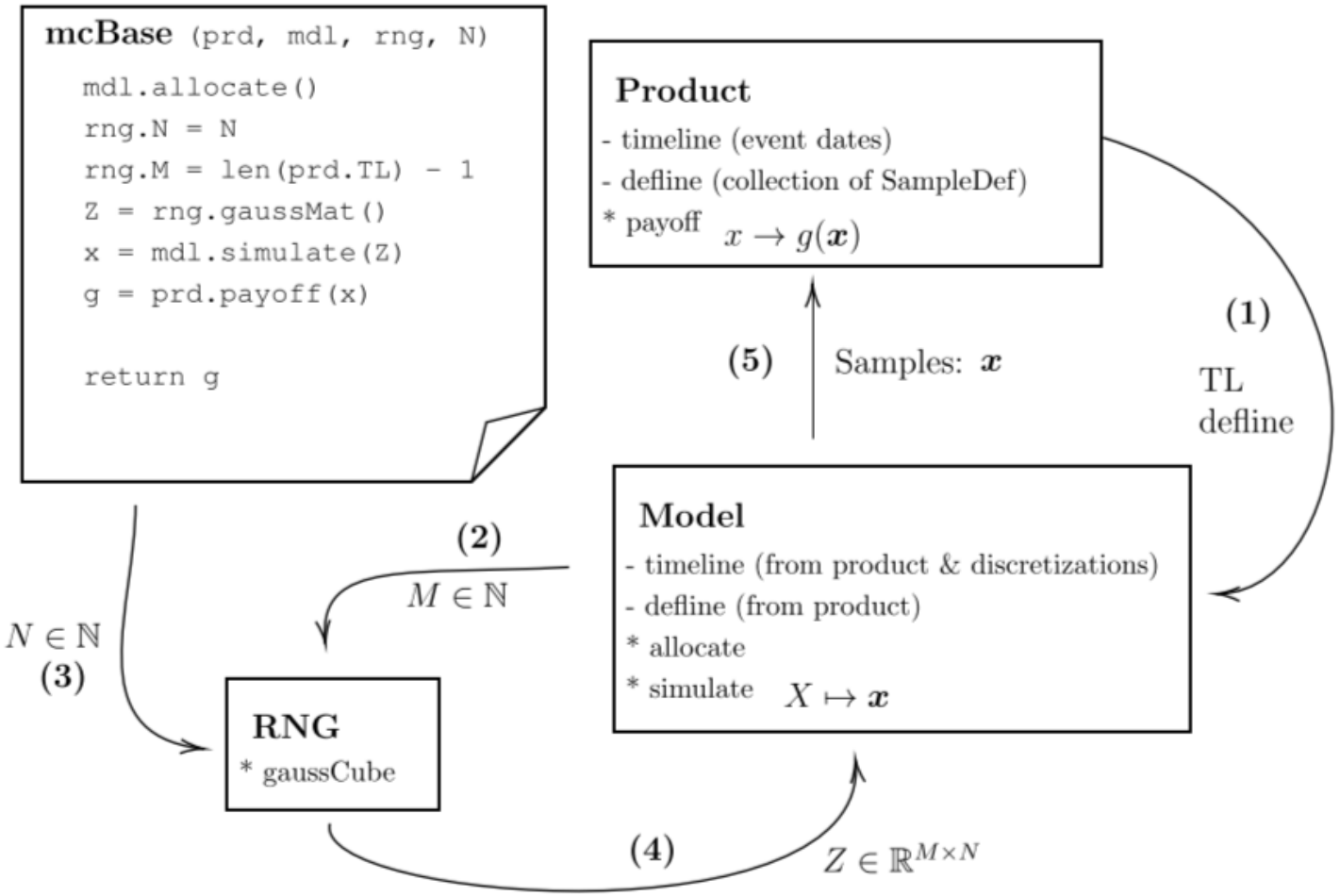


Figure: Overview of the Monte Carlo engine and the basic Monte Carlo algorithm `mcBase`. The arrows represents data flow. A dash (-) indicates an attribute of the object class whereas a star (*) represents callable methods.

Callable products such as Bermudan Swaptions are tricky as their value depend on the exercise strategy. This is proxied by LSM (OLS regression).

Risk Neutral Pricing Theorem for Callable Products

The price of a callable product is given by

$$V_t = \sup_{t \leq \tau \leq T} \mathbb{E}_t^{\mathbb{Q}} \left[g(\boldsymbol{x}_\tau) \right] = \sup_{t \leq \tau \leq T} \mathbb{E}_t^{\mathbb{Q}} \left[\sum_{t \leq t_i \leq \tau} \frac{B(t)}{B(t_i)} CF(t_i, \boldsymbol{x}_{t_i}) \right],$$

where the option holder exercises at the optimal stopping time τ^* according to an optimal exercise strategy.

Valuation split into 2 Distinct Simulations:

- The *pre-simulations* which is a set of $n \in \mathbb{N}$ scenarios used to fit / train the regression. Essentially giving us a proxy or estimator for the optimal exercise strategy.
- The *main simulations* which is a set of $N \in \mathbb{N}$ scenarios which uses the trained regressor to give the estimated optimal exercise times. Then the N simulations are used similar to the previous subsections to evaluate the Monte Carlo estimator.

Automatic Differentiation

The main ideas for calculating sensitivities:

- **Automatic Differentiation (AD)** is *analytical differentiation* of a function through its calculation graph.
- AD works efficiently together with a *pathwise* Monte Carlo implementation.
- Differentiation of the Monte Carlo estimator wrt. $\boldsymbol{a}$ gives the Jacobian

$$\bar{\boldsymbol{a}} := \frac{\partial}{\partial \boldsymbol{a}} \hat{V} \approx \frac{\partial}{\partial \boldsymbol{a}} \left(\frac{1}{N} \sum_{i=1}^{N} g(\boldsymbol{x}^i) \right) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{\partial}{\partial \boldsymbol{a}} \left(g(\boldsymbol{x}^i) \right)}_{\text{Pathwise differentiation}} .$$

2 main approaches to AD through operation overload:

- Forward: Accumulation by using dual numbers ($a + b\varepsilon,\ \varepsilon^2 = 0,\ \varepsilon \neq 0$).
- Backward: **Automatic Adjoint Differentiation (AAD)** stores mathematical operations from the *Directed Acyclic Graph* (DAG) on *tapes* (memory) and enables *check pointing* for improved performance. (Back-propagation!)

Black's formula: Caplet with expiry (fixing) T and accrual period δ

$$Cpl(0,T,T+\delta) = \delta P(0,T)\left(F(0,T,T+\delta)\Phi(d_1) - K\Phi(d_2)\right)$$

$$d_{1,2} = \frac{\log\left(\frac{F(0,T,T+\delta)}{K}\right) \pm \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}$$

Calculate "Vega" with AD (repeated use of the chain-rule):

$$\bar{\sigma} := \frac{\partial C}{\partial std}\frac{\partial std}{\partial \sigma} := \frac{\partial std}{\partial \sigma}\overline{std}.$$

$$\begin{aligned}
\overline{std} &= -\frac{d}{std}\bar{d} + \frac{1}{2}\bar{d}_1 - \frac{1}{2}\,\bar{\mathrm{d}}_2 = -\frac{d}{std}\left(\bar{d}_1 + \,\bar{\mathrm{d}}_2\right) + \frac{1}{2}\bar{d}_1 - \frac{1}{2}\,\bar{\mathrm{d}}_2 \\
&= \left(\frac{1}{2} - \frac{d}{std}\right)\bar{d}_1 - \left(\frac{1}{2} + \frac{d}{std}\right)\bar{d}_2 \\
&= \left(\frac{1}{2} - \frac{d}{std}\right)\varphi(d_1)\overline{nd_1} - \left(\frac{1}{2} + \frac{d}{std}\right)\varphi(d_2)\overline{nd_2} \\
&= \left(\frac{1}{2} - \frac{d}{std}\right)\varphi(d_1)\delta P(0,T+\delta)F\bar{v} + \left(\frac{1}{2} + \frac{d}{std}\right)\varphi(d_2)\delta P(0,T+\delta)K\bar{v}.
\end{aligned}$$

Black's formula for caplets - Calculation Graph

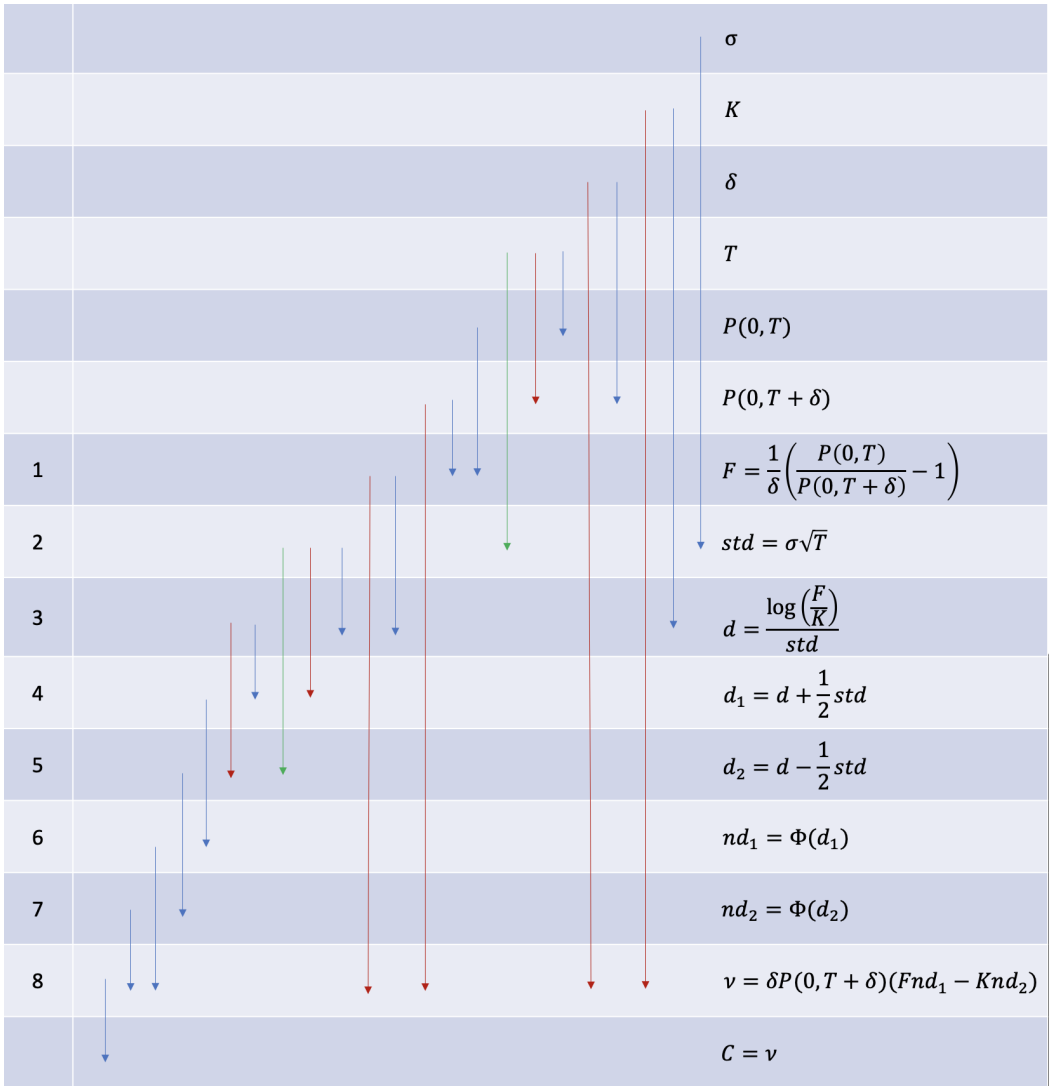


Figure: Visualization of using AD on Black's formula for caplets. The colored arrows refers to how many dependencies each element has.

Differential Machine Learning

Differential Machine Learning - Objectives

The objective is to find a closed-form deterministic function, h, which replicates the true unknown pricing function (assuming Markovianity)

$$V_t = \mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T) \mid \mathcal{F}_t\right] = \mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T) \mid \boldsymbol{x}_t\right] = h(\boldsymbol{x}_t).$$

It's derivatives should also replicate sensitives wrt. arguments $\boldsymbol{a}$, i.e.

$$\frac{\partial V_t}{\partial \boldsymbol{a}} = \frac{\partial}{\partial \boldsymbol{a}}\mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T) \mid \boldsymbol{x}_t\right] = \mathbb{E}^{\mathbb{Q}}\left[\frac{\partial g(\boldsymbol{x}_T)}{\partial \boldsymbol{a}} \mid \boldsymbol{x}_t\right] = \frac{\partial h(\boldsymbol{x}_t)}{\partial \boldsymbol{a}}.$$

The sensitivities can be with respect to both

- Market observables ($\boldsymbol{a} = \boldsymbol{x}$), which are used for hedge-coefficients, or
- Model parameters ($\boldsymbol{a} = \mathbf{P}$), which are often used for calibration.

Main Idea:

1. Simulate pathwise discounted payoffs using Monte Carlo Engine.
2. Use Automatic Adjoint Differentiation to get pathwise sensitivities.
3. Fit an estimator $\hat{h}$ (exists per *Universal Approximation Theorem).*

Performing Monte Carlo simulation with $N \in \mathbb{N}$ scenarios generates our dataset for training / fitting the estimator. Each row is of the form

$$\mathbf{X}^{(i)} \in \mathbb{R}^n, \qquad \boldsymbol{y}^{(i)} \in \mathbb{R}, \qquad \mathbf{Z}^{(i)} = \frac{\partial \boldsymbol{y}^{(i)}}{\partial \mathbf{X}^{(i)}} \in \mathbb{R}^n.$$

If we want hedge coefficients, we need often need sensitivities wrt. market variables ($\boldsymbol{a} = \boldsymbol{x}$). Monte Carlo instrumented with AAD generally give us sensitivities wrt. model parameters, therefore we use a trick through the chain-rule:

Sensitivities wrt. market variables

For each sampled scenario in the Monte Carlo simulation instrumented with AAD, we can calculate the pathwise differential wrt. model parameters

$$\frac{\partial \boldsymbol{y}^{(i)}}{\partial \mathbf{P}} = \frac{\partial \boldsymbol{y}^{(i)}(\mathbf{P})}{\partial \mathbf{X}^{(i)}(\mathbf{P})}^{\top} \frac{\partial \mathbf{X}^{(i)}(\mathbf{P})}{\partial \mathbf{P}}.$$

Rearranging by transposing and multiplying with the pseudo-inverse, we obtain sensitivites wrt. market varialbes

$$\mathbf{Z}^{(i)} \equiv \frac{\partial \boldsymbol{y}^{(i)}(\mathbf{P})}{\partial \mathbf{X}^{(i)}(\mathbf{P})}^{\top} = \left(\frac{\partial \mathbf{X}^{(i)}(\mathbf{P})}{\partial \mathbf{P}}^{\top} \right)^{-1} \frac{\partial \boldsymbol{y}^{(i)}}{\partial \mathbf{P}}.$$

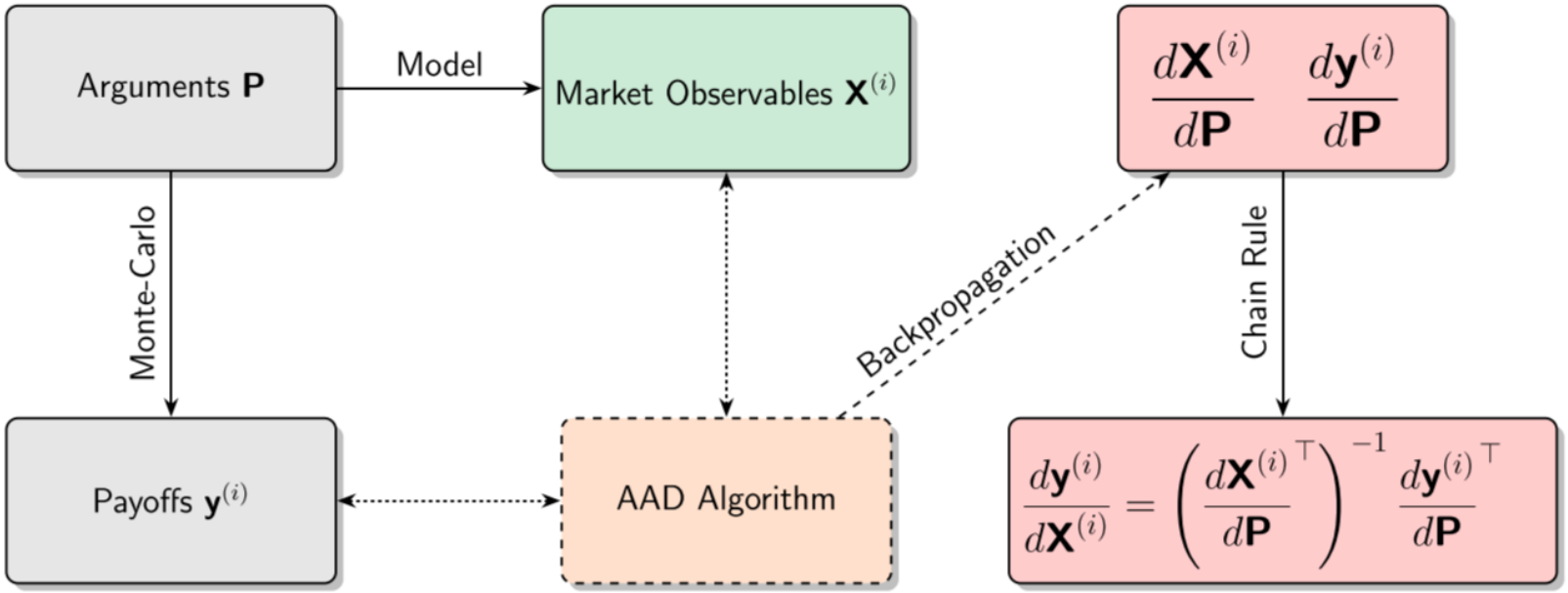


Figure: Overview of training data generation using Monte Carlo instrumented with AAD. Dashed lines indicate calculations carried out by the AAD algorithm.

Differential Regression

Differential Regression imposes *unbiased regularization* of the OLS-regressor by fitting both output ($\boldsymbol{y}$, payoffs) and differential ($\mathbf{Z}$, sensitivities) labels.

Select a set of $B_r \in \mathbb{N}$ differentiable basis functions

$$\psi := \psi(\mathbf{X}) \equiv [\psi_1(\mathbf{X}), \ldots, \psi_{B_r}(\mathbf{X})] \in \mathbb{R}^{N \times B_r}.$$

Objective is to minimize the adjusted MSE

$$\min_{\beta} \left\{ \underbrace{\frac{1}{N} \|\boldsymbol{y} - \psi\beta\|^2}_{\text{Classical MSE}} + \sum_j \frac{1}{N} \alpha_j \|\mathbf{Z}_j - D\psi_j \beta\|^2 \right\}$$

where $D\psi_j := \partial\psi(\mathbf{X})_j / \partial \mathbf{X}_j$ denotes the matrix of derivatives of the basis functions ψ wrt. the j'th covariate $\mathbf{X}_j$ and predictions are made by $\boldsymbol{y} \approx \psi(\mathbf{X})\hat{\beta}$ and $\mathbf{Z}_j \approx D\psi_j \hat{\beta}$ for prices and sensitivities, respectively.

The Adjusted Normal Equation for Differential Regression

$$\hat{\beta} = \left(\psi^\top \psi + \sum_j \alpha_j D\psi_j^\top D\psi_j \right)^{-1} \left(\psi^\top \boldsymbol{y} + \sum_j \alpha_j D\psi_j^\top \mathbf{Z}_j \right).$$

In the Vasicek model the instantaneous short rate dynamics under the risk-neutral measure $\mathbb{Q}$ is given by

$$dr_t = a(b - r_t)dt + \sigma dW_t^{\mathbb{Q}}.$$

An major advantage of its simplicity is the capability of exact simulation - also under the terminal (forward) measure $\mathbb{Q}^T$.

The short rate is normally distributed

$$r_t|r_0 \sim \mathcal{N}\left(r_0 e^{-at} + b\left(1 - e^{-at}\right) \; ; \; \frac{\sigma^2}{2a}\left(1 - e^{-2at}\right)\right).$$

The model has an (exponential) affine termstructure (ATS)

$$\begin{aligned}
P(t,T) &= \exp\left\{-A(t,T) - B(t,T)r(t)\right\}, \\
B(t,T) &= \frac{1 - e^{-a(T-t)}}{a}, \\
A(t,T) &= \left(b - \frac{\sigma^2}{2a^2}\right)\left[(T-t) - B(t,T)\right] + \frac{\sigma^2}{4a}B^2(t,T).
\end{aligned}$$

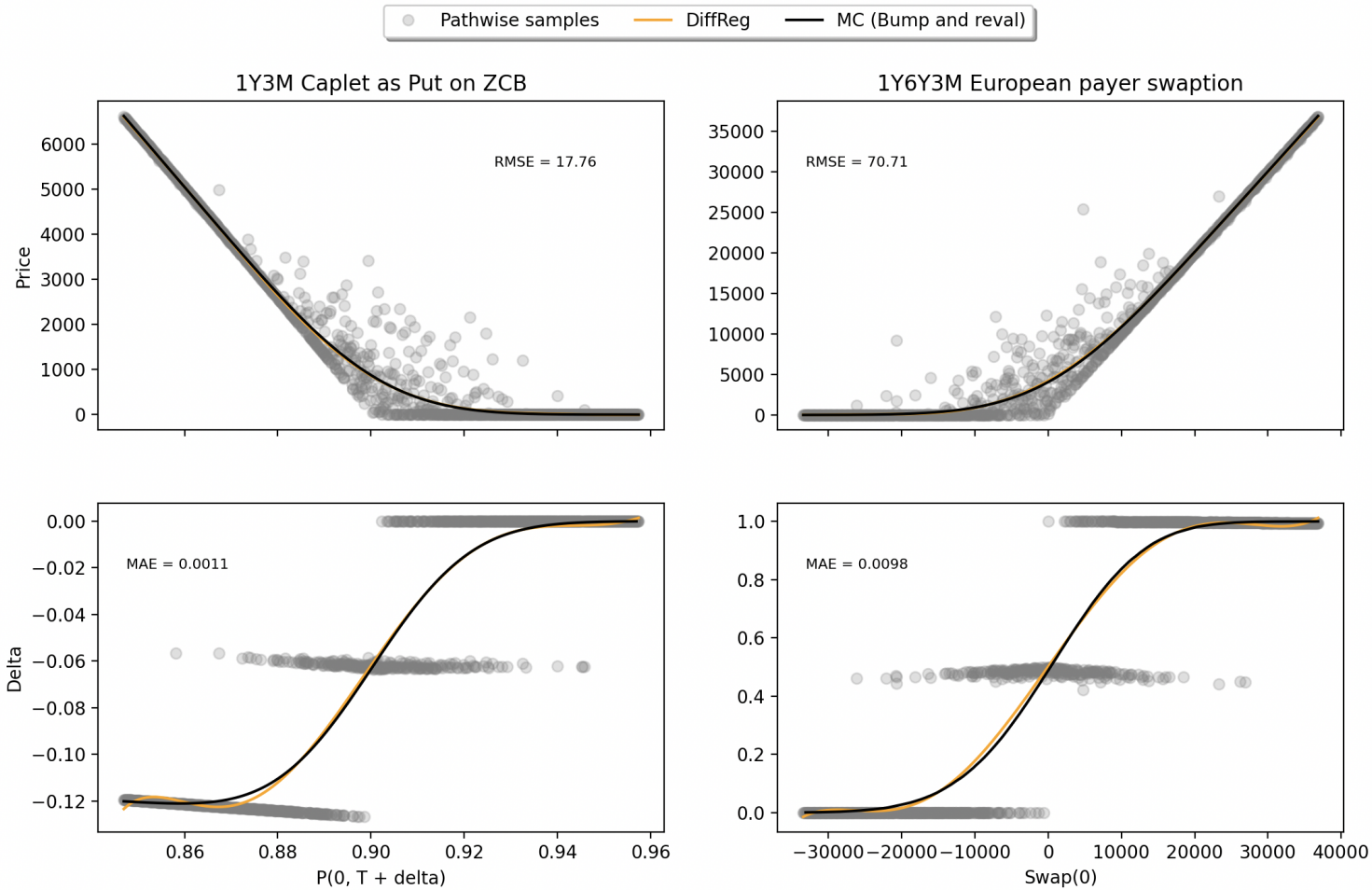


Figure: Price (upper panel) and delta (lower panel) estimation for a 1Y3M caplet on zero-coupon bond (left panel) and an 1Y6Y3M European swaption (right panel), by differential regression using polynomial degree $= 7$ and regularization $\alpha = 1$. The model is trained on 1,024 AV training samples, with strike/swap rate $K = 8.92\%$ and notional $N = 1,000,000$.

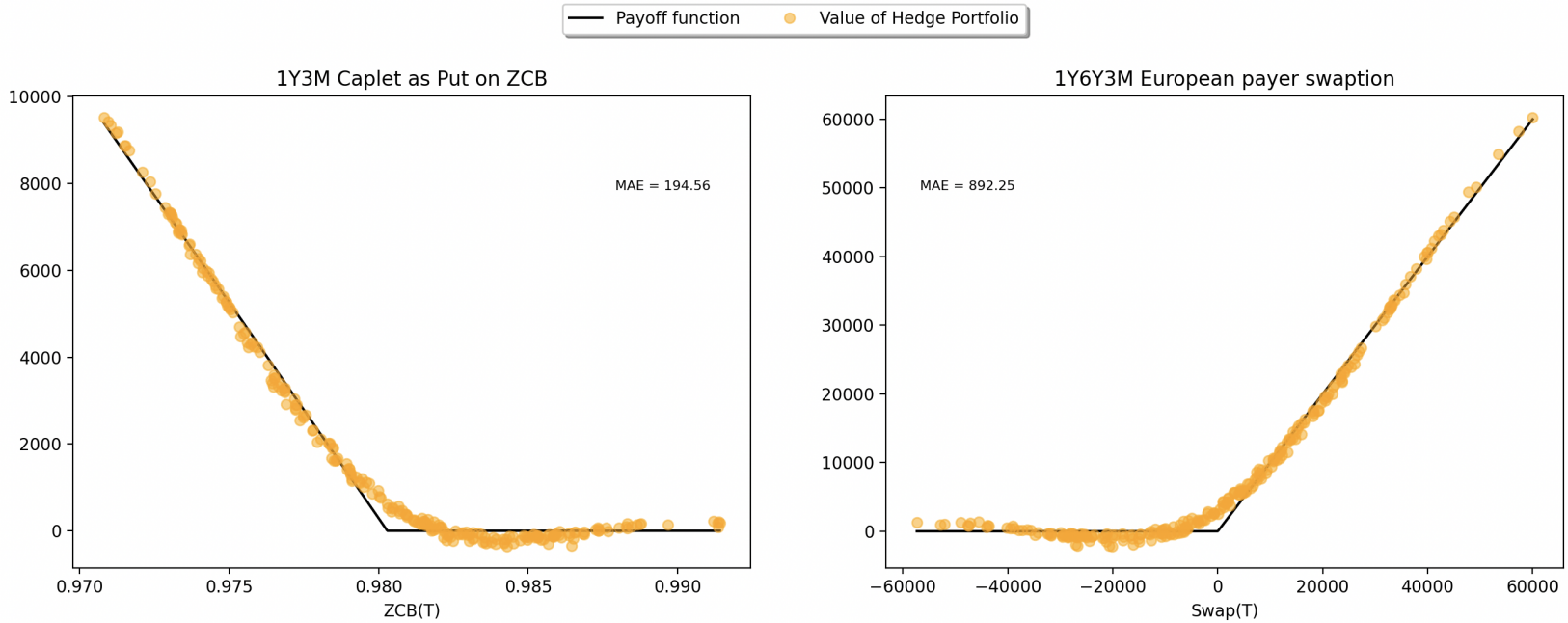


Figure: Hedge error of delta hedging a 1Y3M caplet as put on a zero-coupon bond with strike rate $K = 8.71\%$ and a 1Y6Y3M European payer swaption with swap rate $K = 7.88\%$ using differential regression, both with notional $N = 1,000,000$. The model is in both cases trained on 1,024 AV samples with using 7 degree polynomial and regularization $\alpha = 1$ and the portfolio is re-balanced 250 times through the options' lifespan of 1 year.

Differential Regression - Delta Estimates for Different Expires

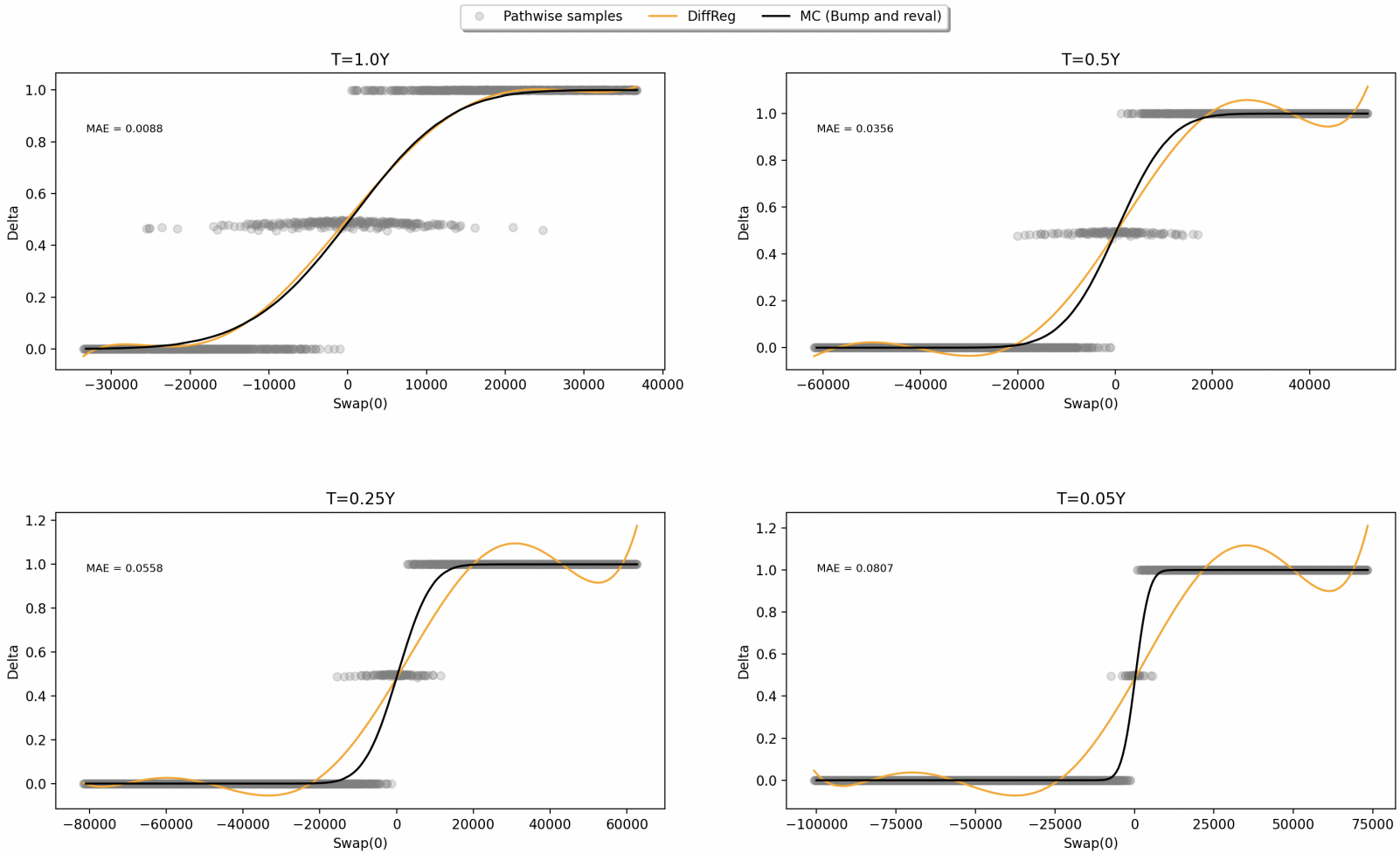


Figure: Delta estimation for European payer swaption until expiry, by 7'th order differential regression. The model is trained on 1,024 AV training samples, with strike/swap rate $K = 8.982\%$ and notional $N = 1,000,000$.

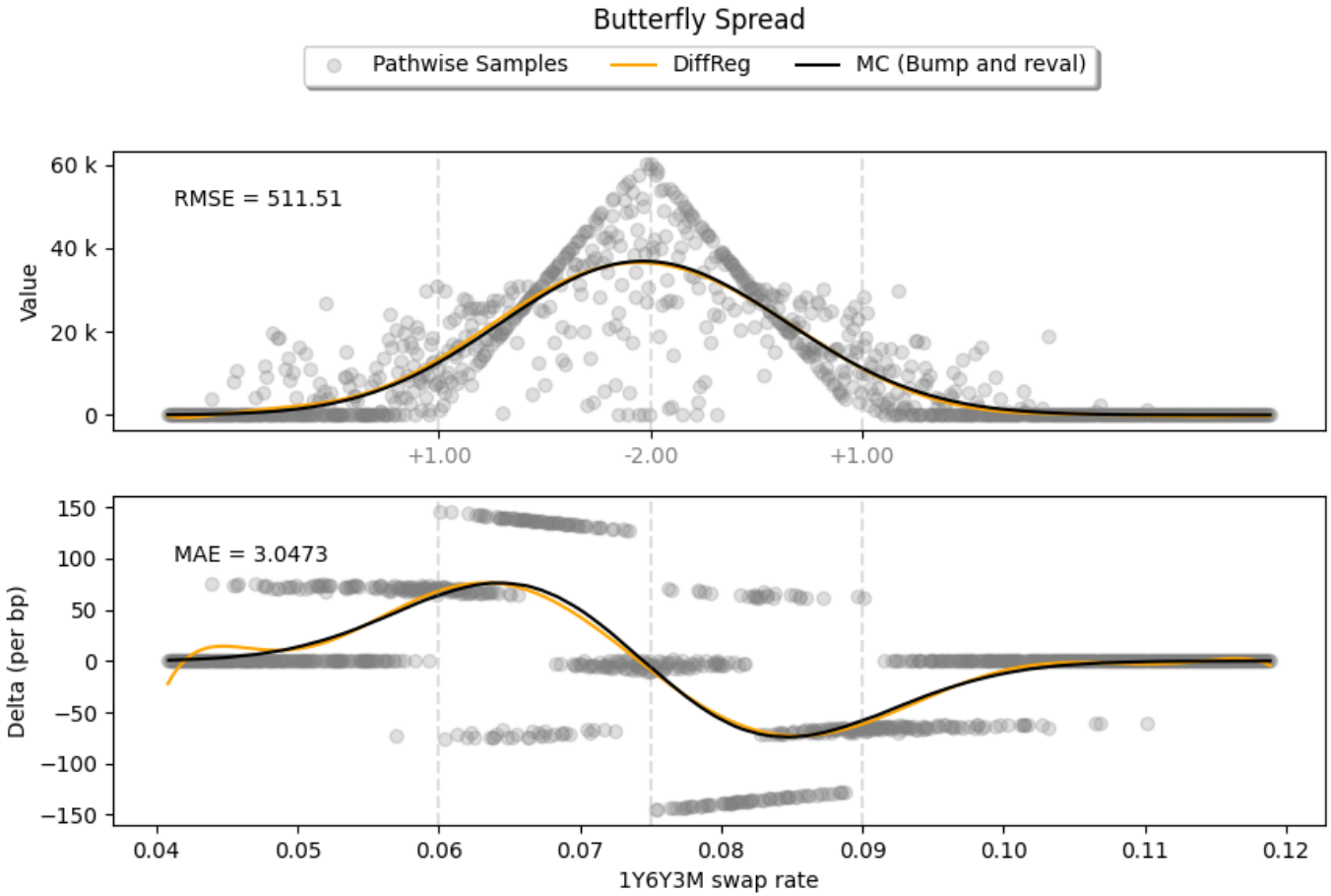


Figure: Butterfly spread value and sensitivity functions over different swap rates. The underlying swaptions all expiry at $1Y$, and the underlying swaps fixes 4 times per year ($\delta = 0.25$) from $1Y, \ldots, 6Y$, each with a notional of $1,000,000$. The fix rates are 6.0%, 7.5%, and 9.0% The differential regressor is trained on $1,024$ samples with AV reduction. The differential regularization is set to $\alpha = 1.0$, and the polynomial has 9 degrees.

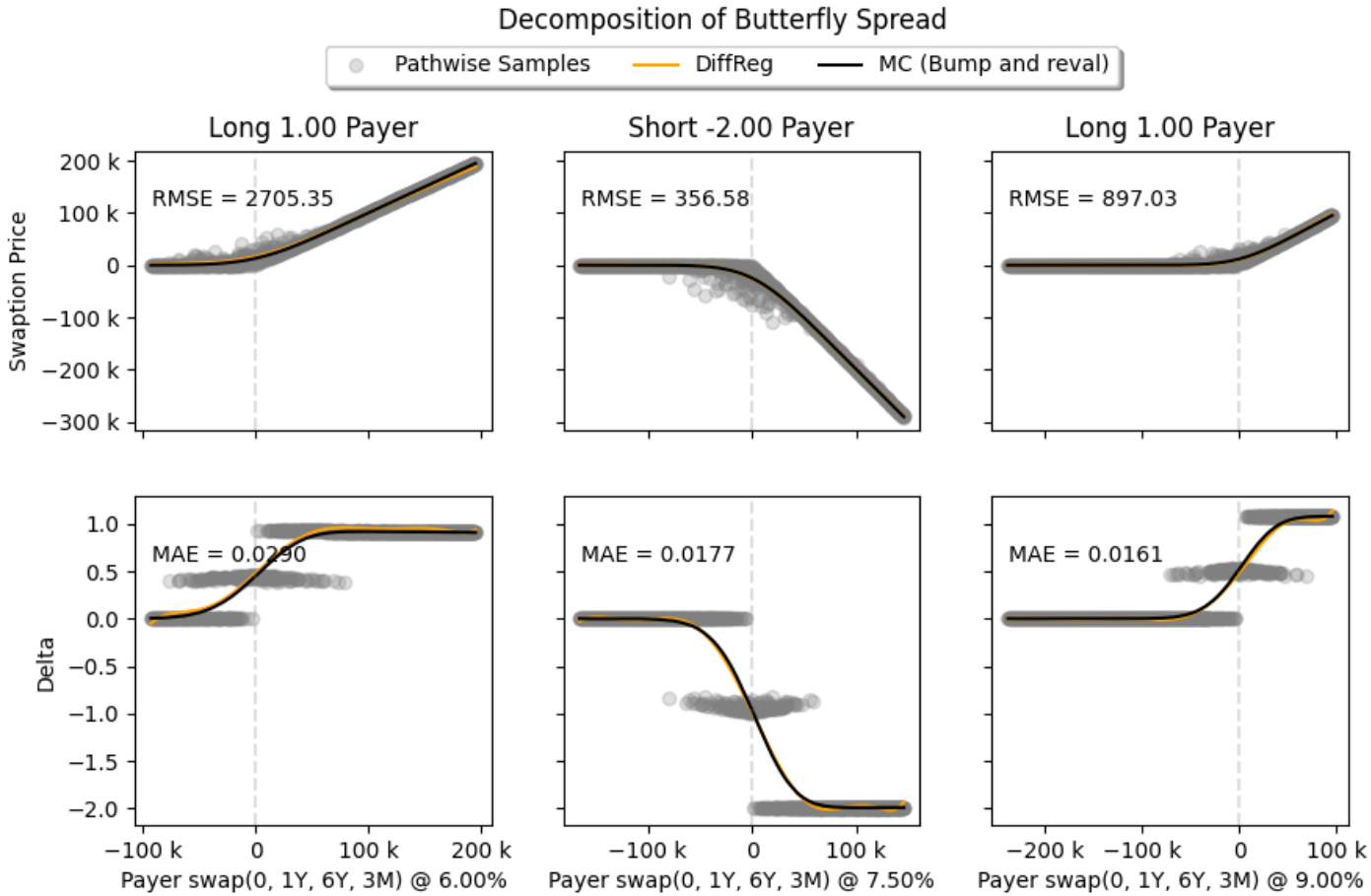


Figure: Decomposition of Butterfly spread. Settings are the same as in the figure above.

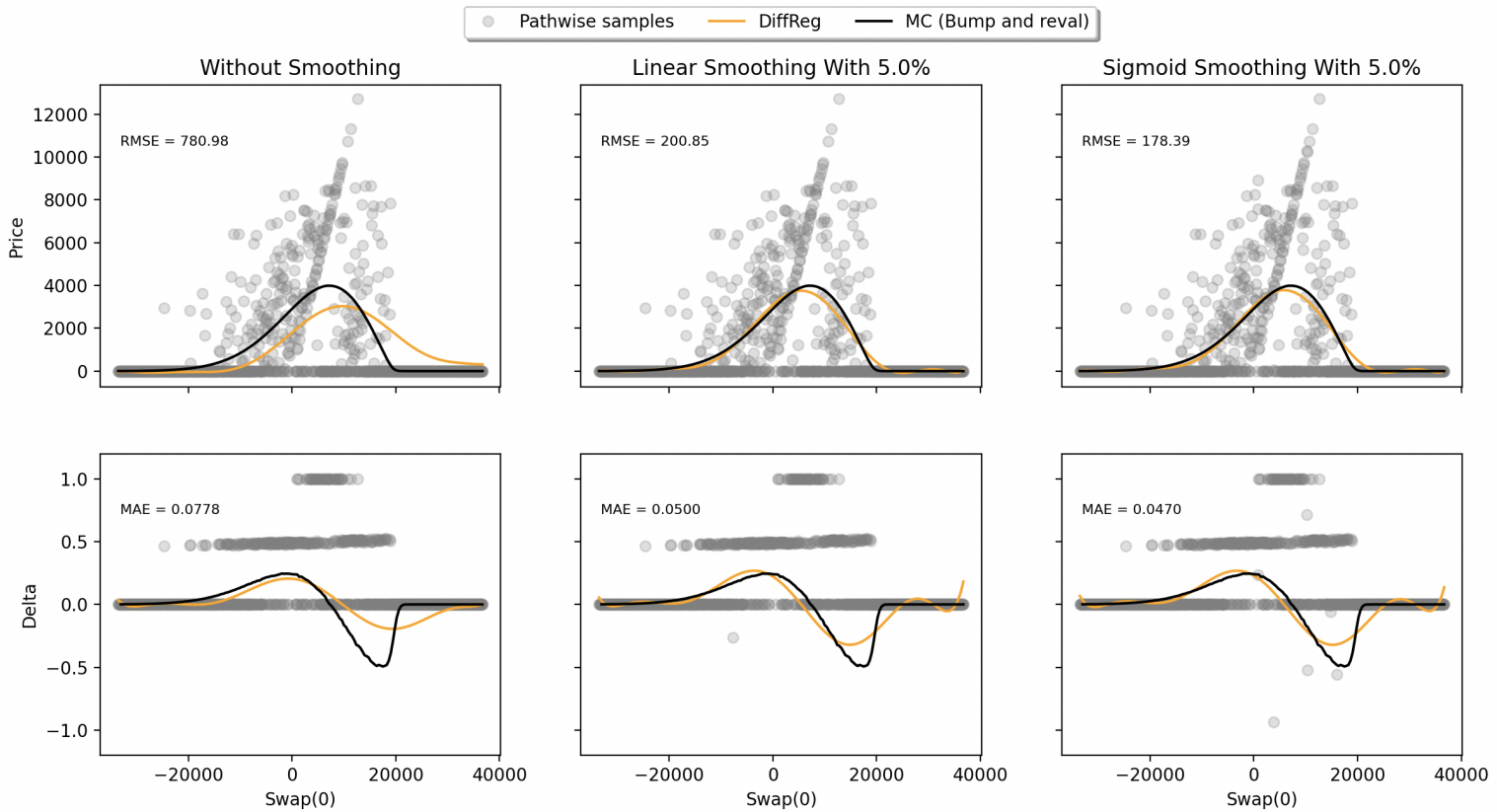


Figure: Delta and price estimation of a 1Y3M European up-and-out barrier swaption using differential regression trained on 1,024 training samples with 9 degrees, interactions, and $\alpha = 1.0$ for regularization. The barrier is $H = 20,000$ with smoothing interval $\pm 5\%$ of H. The swaption is monitored every 2.5 day, with a notional of 1,000,000 and strike rate 8.71%. The MC price and delta are estimated without any smoothing.

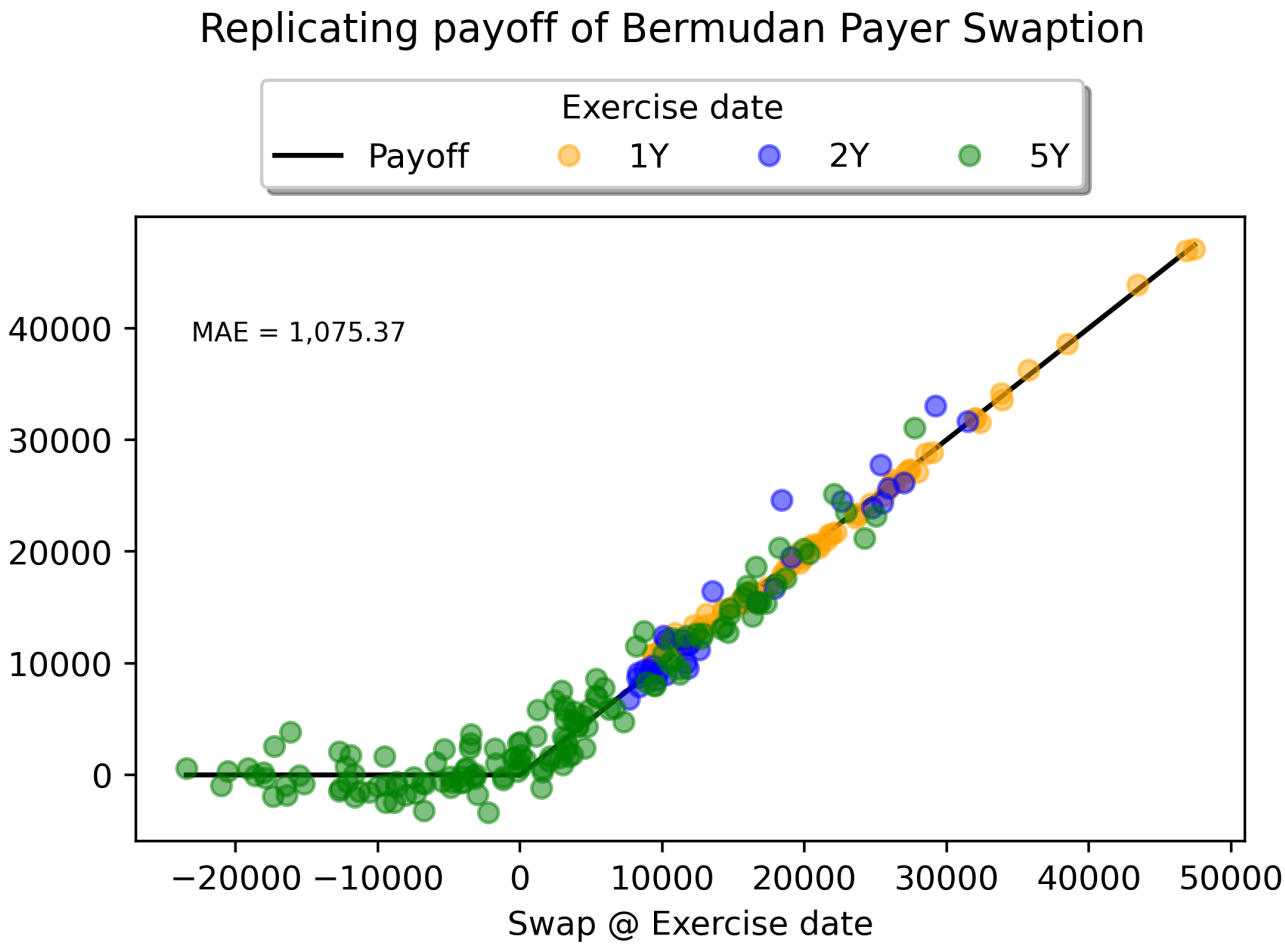


Figure: Replication of payoff function for a co-terminal Bermudan Payer Swaption using differential regression trained on 1,024 training samples with 9 degrees, interactions, and $\alpha = 1.0$ for regularization. Exercise dates are 1Y, 2Y, 5Y, maturity of the underlying payer swaps is 10Y, the strike rate is 7.03%, and the notional is 1,000,000. The hedge frequency is 100 times per year. The exercise condition is proxied using LSM with $n = 5,000$ pre-simulations.

Differential Neural Network

We express a (feed-foraward) Neural Network as function which can be expressed a directed computational graph with $L \in \mathbb{N}$ layers

$$\boldsymbol{y} = G_L(\boldsymbol{X}) + \varepsilon.$$

Each layer consists of

- A weight matrix $w_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$,
- A bias vector $b_\ell \in \mathbb{R}^{n_\ell}$,
- An activation function $g_\ell : \mathbb{R} \to \mathbb{R}$.

The transition from the ℓ'th layer to the $\ell + 1$'th is characterized by the transformation

$$\nu_{\ell+1} = g_\ell(\nu_\ell)w_\ell + b_\ell \in \mathbb{R}^{n_\ell}.$$

Their composition defines the Neural Network graph

$$G_L(x) = \nu_L \circ \nu_{L-1} \circ \cdots \circ \nu_0.$$

The *Universal Approximation Theorem* guarantees the existent of a perfect estimator in the form of a Neural Network (in the limit).

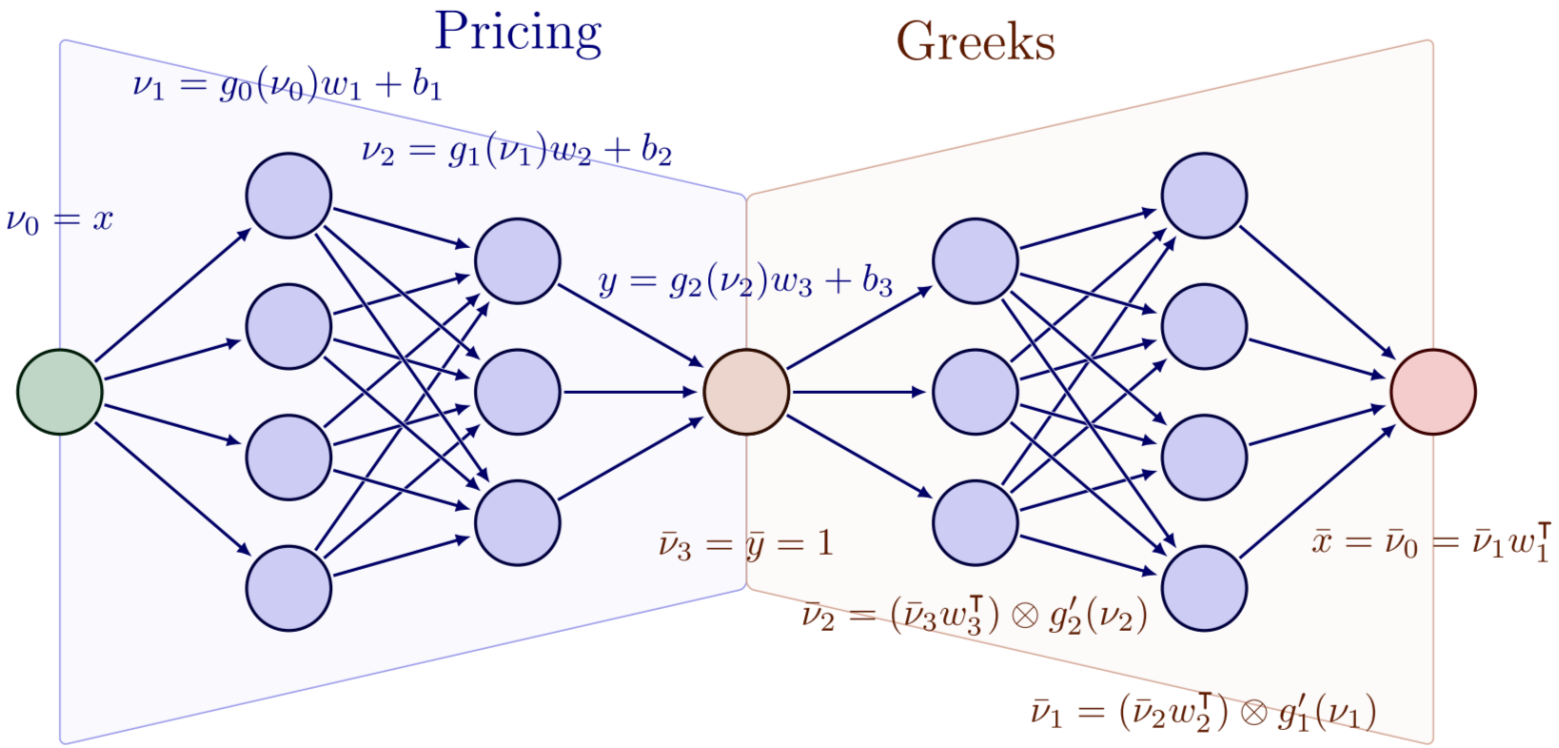


Figure: Illustration of Twin Network

- Training a (Twin) Neural Network is done through **Back-propagation** (AAD) and **Stochastic Gradient Decent**.
- Data is **standardized** before training.
- The **Objective function** for determining the weights and biases are

$$\begin{aligned}\{w_\ell, b_\ell\}_{\ell=1,\dots,L} &= \arg\min \mathcal{L}\left(\{w_\ell, b_\ell\}_{\ell=1,\dots,L}\right) \\ &= a\mathcal{L}_{\text{val}} + b\mathcal{L}_{\text{diff}} \\ &= aMSE + b\frac{\sum_j \lambda_j^2 \overline{MSE}_j}{n}\end{aligned}$$

- We use the *SoftPlus* and *Sigmoid* as activation functions

$$\text{SoftPlus}(x) = \log(1+e^x), \quad \text{SoftPlus}'(x) = \text{Sigmoid}(x) = (1+e^{-x})^{-1}.$$

Differential Neural Network - Delta Estimates for Different Expires

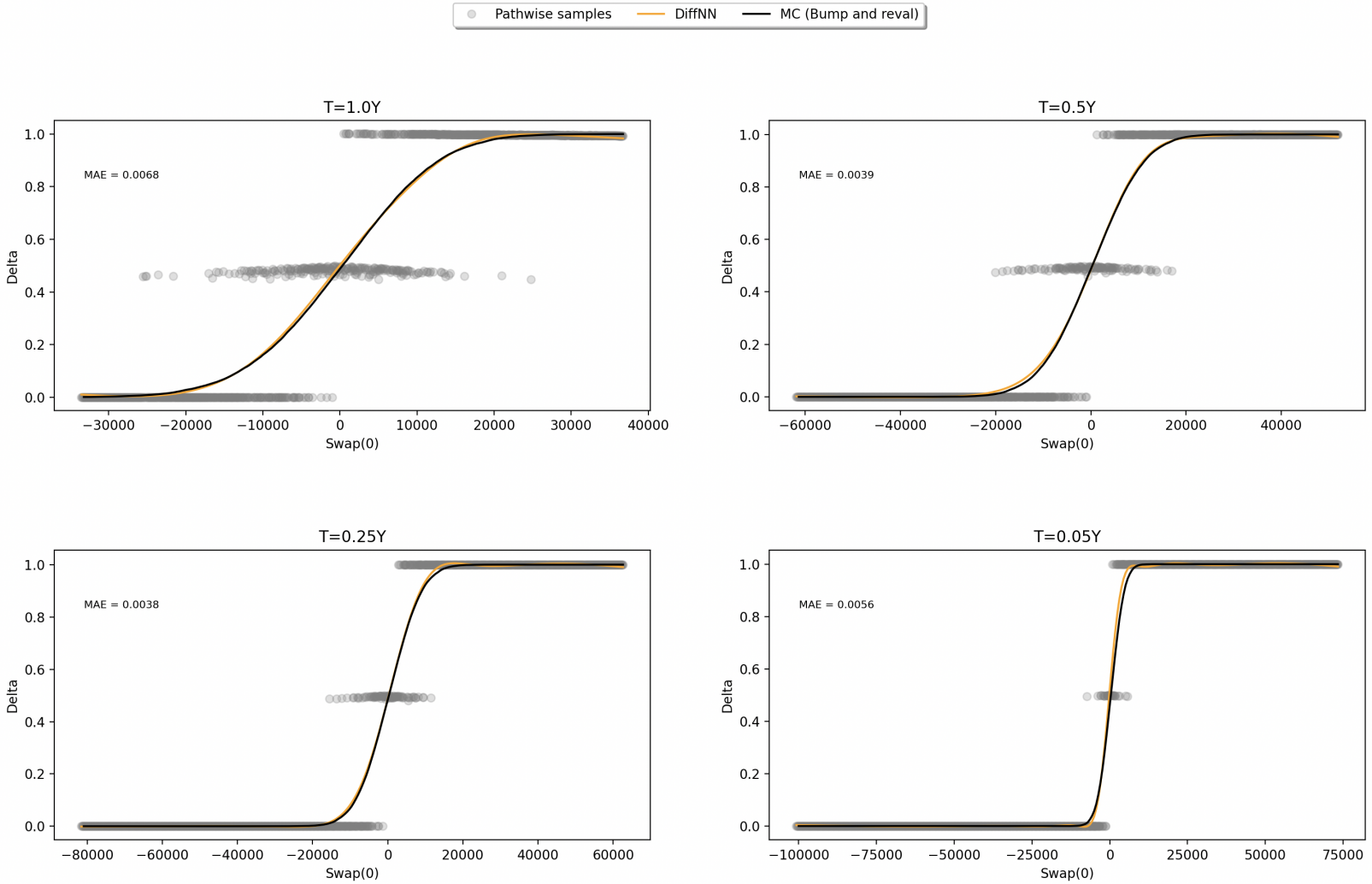


Figure: Delta estimation for a 1Y6Y3M European payer swaption until expiry by differential Neural Network. The model is trained on 4096 AV training samples, with swap strike rate $K = 8.71\%$ and notional $N = 1,000,000$. NN specifications; epochs $= 500$, layers $= 4$ with hidden units $= 20$ in each.

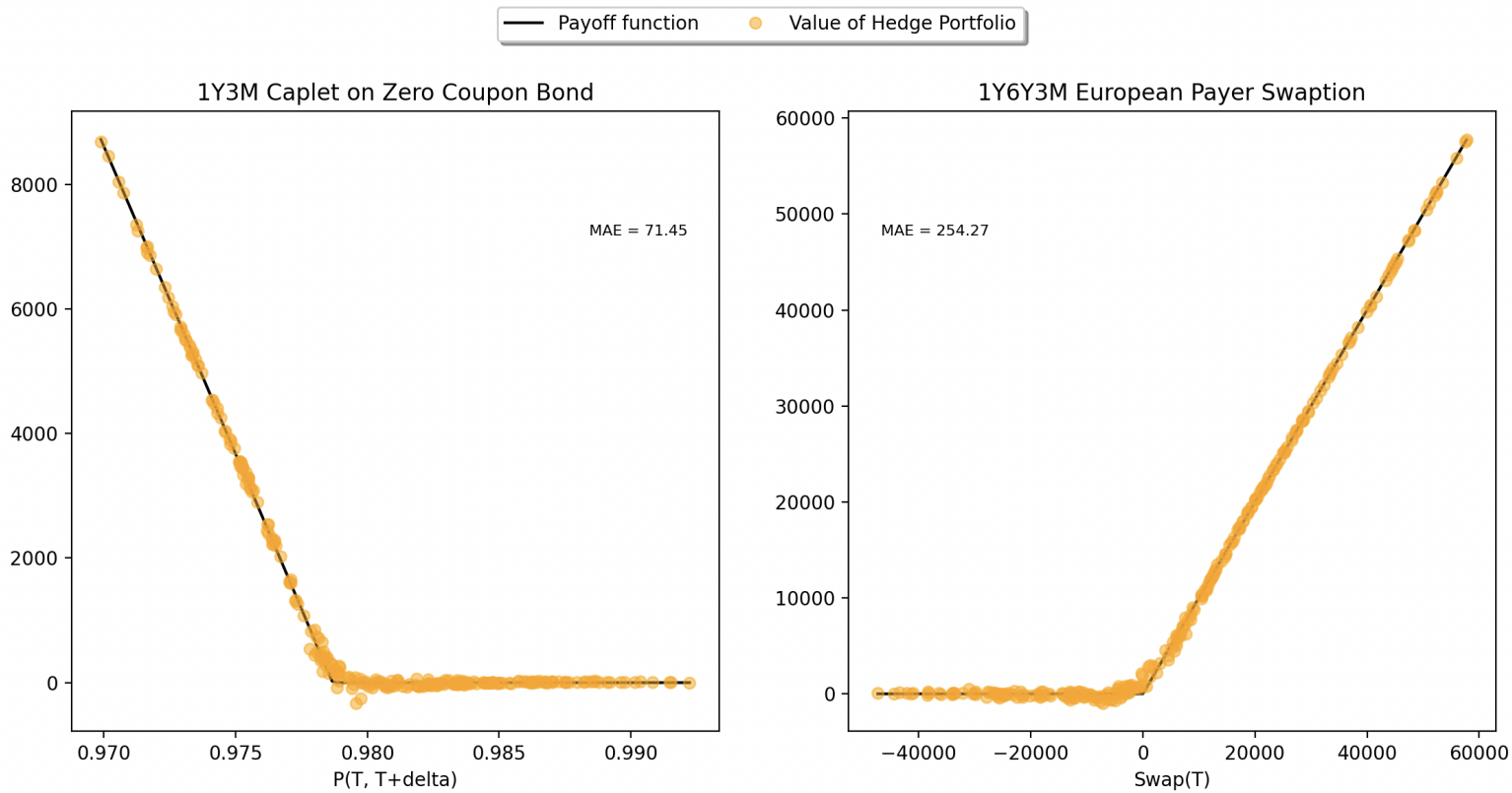


Figure: Delta hedge every 2.5 half day of a 1Y3M Caplet on a ZCB and a 1Y6Y3M European payer swaption by differential Neural Network. The model is trained on 4096 AV training samples, with swap strike rate $K = 8.71\%$ and notional $N = 1,000,000$. NN specifications; epochs $= 250$, layers $= 4$ with hidden units $= 20$ in each.

Takeaway:

- Differential Regression for speed.
- Differential Neural Network for accuracy.

Product	**Diff. Reg**	**Diff. NN**
Caplet	178.28	61.02
Eur Swaption	347.29	240.41
Ber Swaption	2222.68	1624.51

Table: Accuracy comparison of standard deviation of hedge error.

Product	**Diff. Reg**	**Diff. NN**
Caplet	1.429	1310.269
Eur Swaption	1.598	1327.414
Ber Swaption	1.792	1349.301

Table: Time complexity comparison of price estimation in milliseconds.

General Multi-Factor Stochastic Volatility Model
Trolle & Schwartz (2006)

Features of the General Multi-Factor Stochastic Volatility Model by Trolle and Schwartz (2006):

- Markovian specification within the HJM-framework with $N \times 2$ state variables describing the entire term structure through the instantaneous forward curve, $f(t, T)$. Hench, also arbitrage-free.
- Semi-analytical pricing of options on ZCB (hence also caps / floors).
- Accurate approximation of options on coupon bonds (hence swaptions).
- Well-suited for valuation of exotic interest rate derivatives by MC.
- "Extended" Affine Term Structure specification for market price of risk - allows for econometric estimation using Kalman filter and forecasting.
- Fits market data well and highlights relative price difference between caps / floors and swaptions.

The forward curve dynamics show *unspanned stochastic volatility* and follow

$$df(t,T) = \mu_f(t,T)dt + \sum_{i=1}^{N} \sigma_{f,i}(t,T)\sqrt{\nu_i(t)}dW_i^{\mathbb{Q}}(t),$$

$$d\nu_i(t) = \kappa_i(\theta_i - \nu_i(t))dt + \sigma_i\sqrt{\nu_i(t)}\left(\rho_i dW_i^{\mathbb{Q}}(t) + \sqrt{1-\rho_i^2}dZ_i^{\mathbb{Q}}\right).$$

HJM Specification of Forward Rate Volatility (Arbitrage-free & Markovian)

The model is an instance of the HJM framework and is therefore arbitrage-free as the drift can be specified on the form

$$\mu_f(t,T) = \sum_{i=1}^{N} \nu_i(t)\sigma_{f,i}(t,T)\int_t^T \sigma_{f,i}(t,u)du.$$

The forward rate volatility is specified in spreadable form (Markovian) by

$$\sigma_{f,i}(t,T) := (\alpha_{0,i} + \alpha_{1,i}(T-t))e^{-\gamma_i(T-t)}.$$

In our appendix, we provide the details for the derivation of the deterministic formula for the forward curve as a function of the state variables

$$f(t,T) = f(0,T) + \sum_{i=1}^{N} \mathcal{B}_{x_i}(T-t)x_i(t) + \sum_{i=1}^{N}\sum_{j=1}^{6} \mathcal{B}_{\phi_{j,i}}(T-t)\phi_{j,i}(t),$$

where we define the following deterministic functions

$$\mathcal{B}_{x_i}(\tau) = (\alpha_{0,i} + \alpha_{1,i}\tau)e^{-\gamma_i\tau},$$

$$\mathcal{B}_{\phi_{1,i}}(\tau) = \alpha_{1,i}e^{-\gamma_i\tau},$$

$$\mathcal{B}_{\phi_{2,i}}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(\alpha_{0,i} + \alpha_{1,i}(T-t)),$$

$$\mathcal{B}_{\phi_{3,i}}(\tau) = -\left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right) + \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma} + 2\alpha_{0,i}\right)(T-t) + \frac{\alpha_{1,i}^2}{\gamma}(T-t)^2\right)e^{-2\gamma_i(T-t)},$$

$$\mathcal{B}_{\phi_{4,i}}(\tau) = \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)e^{-\gamma(T-t)},$$

$$\mathcal{B}_{\phi_{5,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma_i} + 2\alpha_{0,i} + 2\alpha_{1,i}(T-t)\right)e^{-2\gamma_i(T-t)},$$

$$\mathcal{B}_{\phi_{6,i}}(\tau) = -\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}.$$

The actual simulation of the model is carried out through the $N \times 8$ state and auxiliary variables

$$
\begin{aligned}
x_i(t) &= \int_0^t \sqrt{\nu_i(s)} e^{-\gamma_i(t-s)} dW_i^{\mathbb{Q}}(s), \\
\phi_{1,i}(t) &= \int_0^t \sqrt{\nu_i(s)} (t-s) e^{-\gamma_i(t-s)} dW_i^{\mathbb{Q}}(s), \\
\phi_{2,i}(t) &= \int_0^t \nu_i(s) e^{-\gamma_i(t-s)} ds, \\
\phi_{3,i}(t) &= \int_0^t \nu_i(s) e^{-2\gamma_i(t-s)} ds, \\
\phi_{4,i}(t) &= \int_0^t \nu_i(s) (t-s) e^{-\gamma_i(t-s)} ds, \\
\phi_{5,i}(t) &= \int_0^t \nu_i(s) (t-s) e^{-2\gamma_i(t-s)} ds, \\
\phi_{6,i}(t) &= \int_0^t \nu_i(s) (t-s)^2 e^{-2\gamma_i(t-s)} ds,
\end{aligned}
$$

where the $\nu_i(t)$ works in the background.

In our appendix, we provide details for the derivation of the bond reconstruction formula

$$P(t,T) = \exp\left\{-\int_t^T f(t,u)du\right\}$$
$$= \frac{P(0,t)}{P(0,T)} \exp\left\{\sum_{i=1}^{N} B_{x_i}(T-t)x_i(t) + \sum_{i=1}^{N}\sum_{j=1}^{6} B_{\phi_{j,i}}(T-t)\phi_{j,i}(t)\right\}$$

where we have new set of deterministic functions

$$B_{x_i}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau} - 1\right) + \tau e^{-\gamma_i\tau}\right),$$

$$B_{\phi_{1,i}}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(e^{-\gamma_i\tau} - 1\right),$$

$$B_{\phi_{2,i}}(\tau) = \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau} - 1\right) + \tau e^{-\gamma_i\tau}\right),$$

$$B_{\phi_{3,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{2\gamma_i^2} + \frac{\alpha_{0,i}}{\gamma_i} + \frac{\alpha_{0,i}^2}{2\alpha_{1,i}}\right)\left(e^{-2\gamma_i\tau} - 1\right) + \left(\frac{\alpha_{1,i}}{\gamma_i} + \alpha_{0,i}\right)\tau e^{-2\gamma_i\tau} + \frac{\alpha_{1,i}}{2}\tau^2 e^{-2\gamma_i\tau}\right),$$

$$B_{\phi_{4,i}}(\tau) = \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau} - 1\right),$$

$$B_{\phi_{5,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{\gamma_i} + \alpha_{0,i}\right)\left(e^{-2\gamma_i\tau} - 1\right) + \alpha_{1,i}\tau e^{-2\gamma_i\tau}\right),$$

$$B_{\phi_{6,i}}(\tau) = -\frac{1}{2}\left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(e^{-2\gamma_i\tau} - 1\right).$$

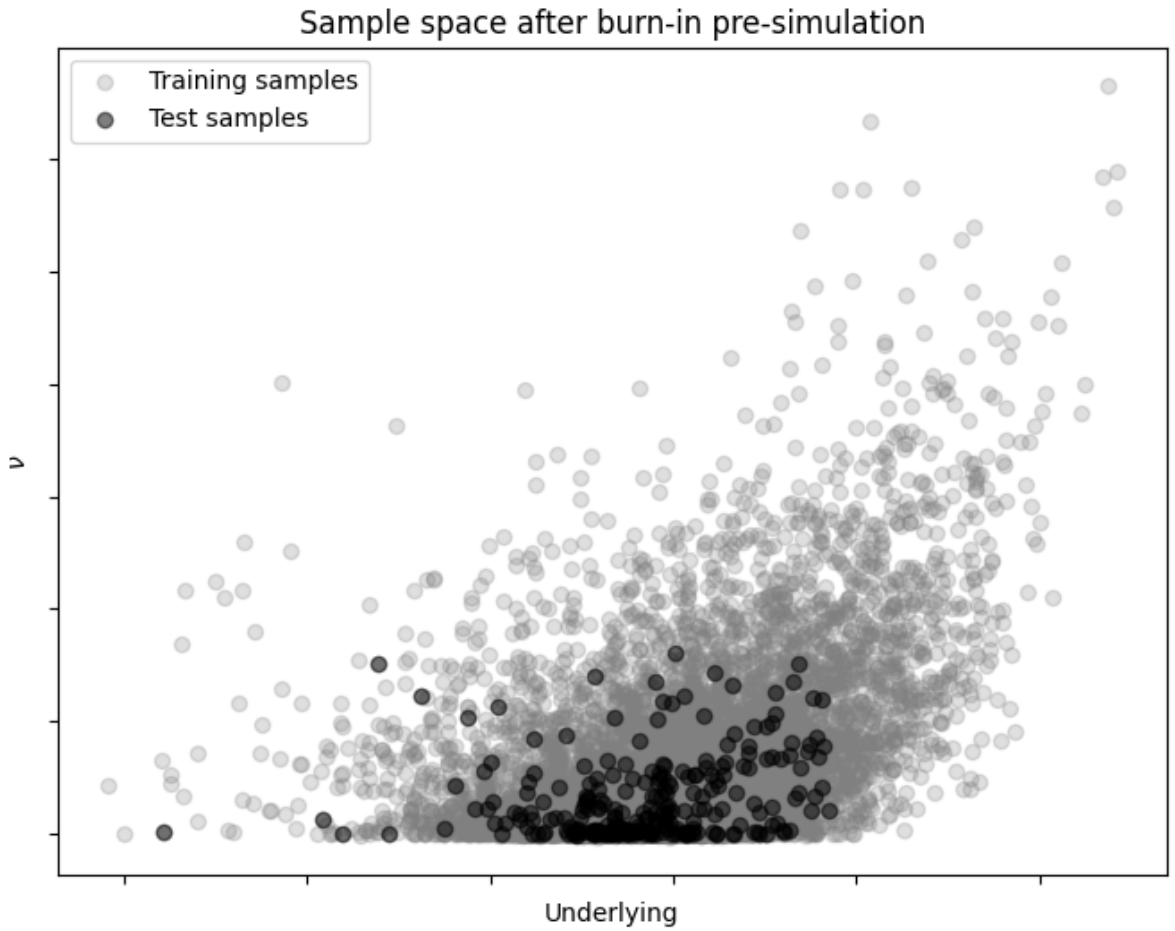


Figure: Sample space after burn-in pre-simulation from time t_{-1} to t_0.

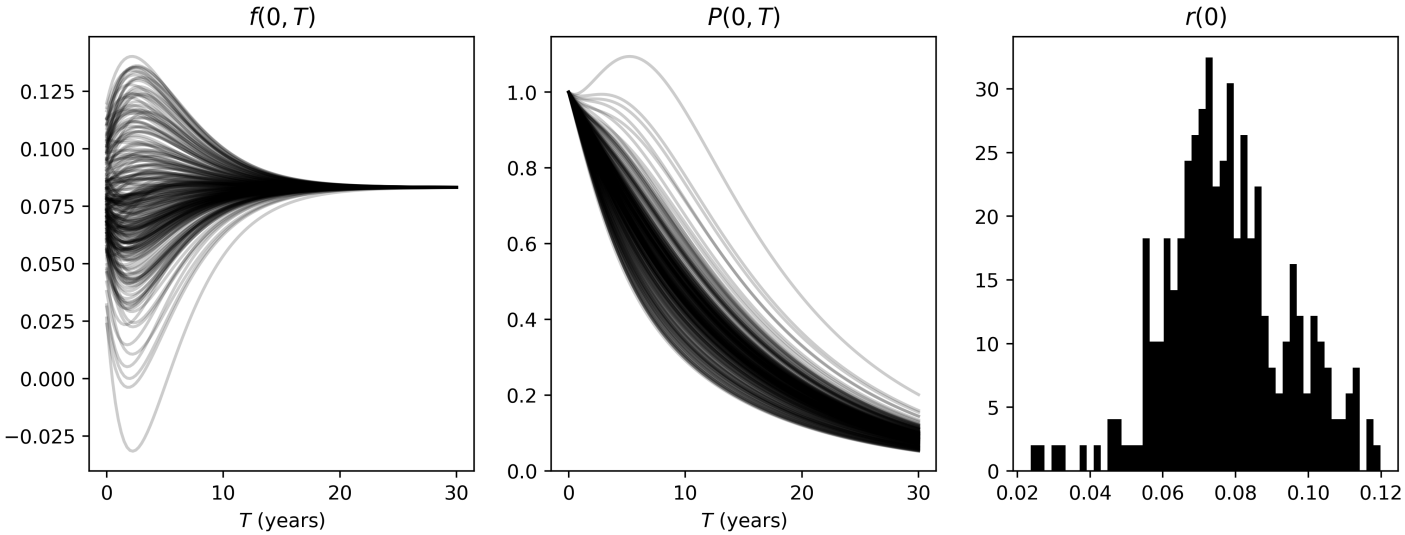


Figure: Summary of states for test cases after burn-in pre-simulations. Left: Instantaneous forward curves. Mid: Zero coupon bond curves. Right: Density of instantaneous risk free short rate.

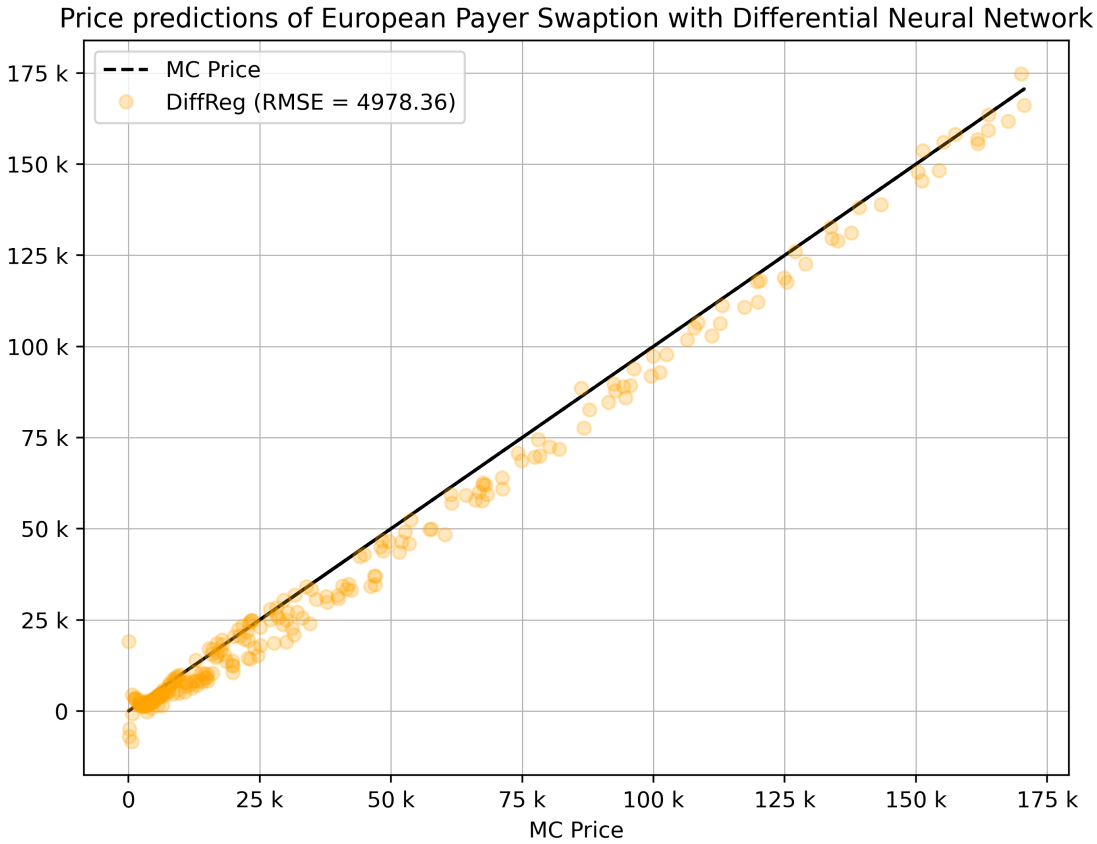


Figure: Comparison of price predictions for a European Payer Swaption using Differential Regression and Monte Carlo price estimates from $50,000$ paths. The exercise date is $1Y$. The underlying payer swap has a notional of $N = 1,000,000$, a fixing rate at 8.41 pct., quarterly fixings, $\delta = 0.25$, with the first fixing date being at the exercise date and the last being 5 years later. The Differential Regressor is trained on 1,024 training samples with 5 degrees, interactions, and using $\alpha = 1.0$ for regularization.

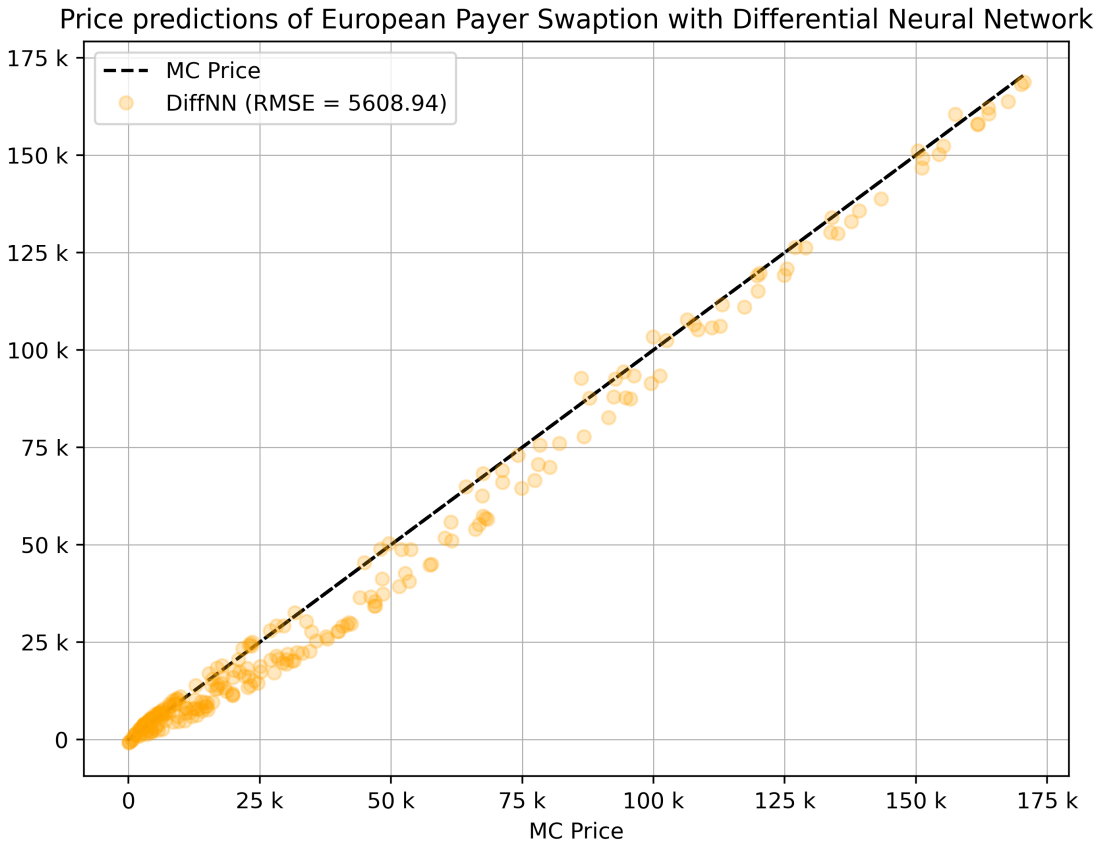
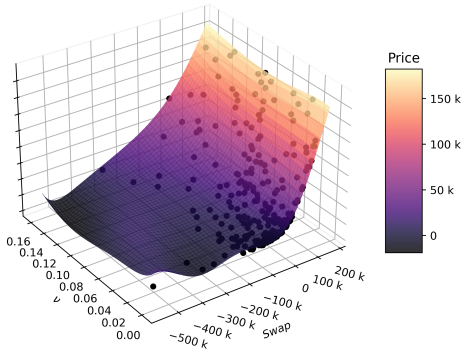
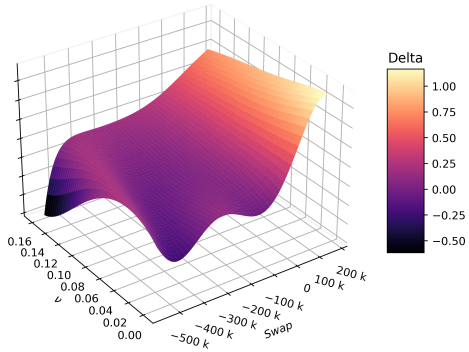


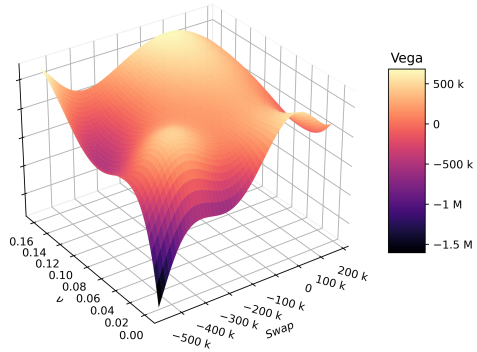
Figure: Comparison of price predictions for a European Payer Swaption using Differential Neural Network and Monte Carlo price estimates from $50,000$ paths. The exercise date is $1Y$. The underlying payer swap has a notional of $N = 1,000,000$, a fixing rate at 8.41 pct., quarterly fixings, $\delta = 0.25$, with the first fixing date being at the exercise date and the last being 5 years later. The Differential Neural Network has 4 hidden layers, each with 20 nodes, is trained on 8,192 training samples with 250 epochs. The activation functions used are softplus and sigmoid. For regularization we use $\lambda = 1.0$.



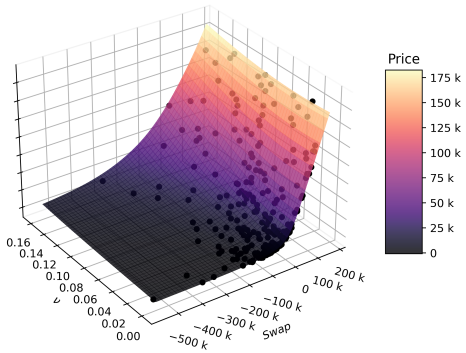
Price DiffReg



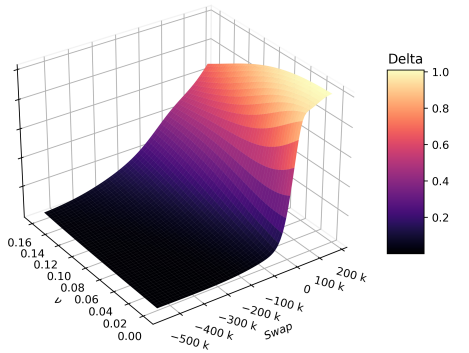
Delta DiffReg



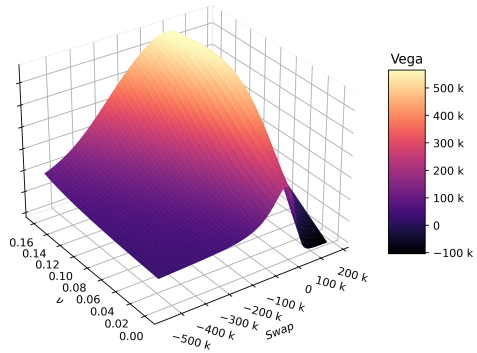
Vega DiffReg



Price DiffNN



Delta DiffNN



Vega DiffNN

Conclusion and Further Research

Conclusion: We have successfully implemented a flexible and scalable Monte Carlo engine instrumented with Automatic Adjoint Differentiation for the purpose of training Differential Machine Learning models "live" to value and hedge interest rate derivatives.

Our findings show that the choice of the specific Differential Machine Learning model is a trade-off:

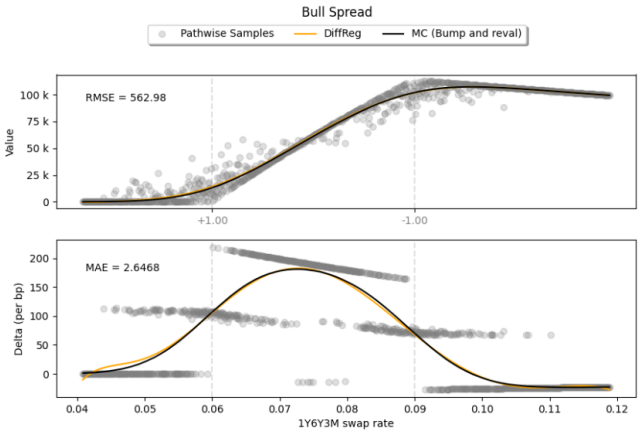
- Differential Regression for speed.
- Differential Neural Network for accuracy.

Further Research:

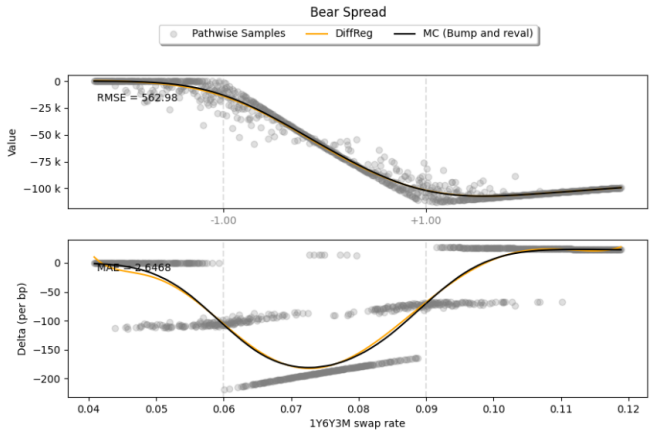
- Implicit Function Theorem for calibration.
- Differential PCA for dimension reduction.
- Advanced Asymptotic Control.

This slide is intentionally left blank

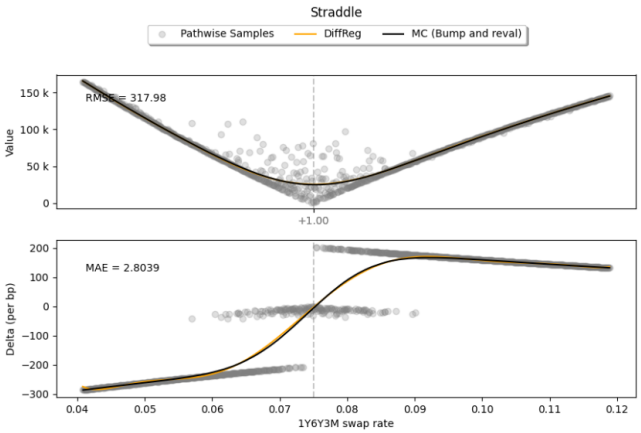
Appendix



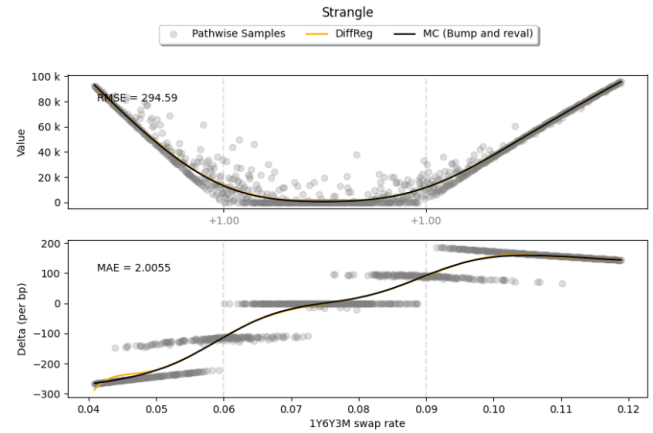
(a) Bull Spread (long 1 payer swaption @ 6.0%. and short 1 payer swaption @ 9.0%.)



(b) Bear Spread (short 1 payer swaption @ 6.0%. and long 1 payer swaption @ 9.0%)



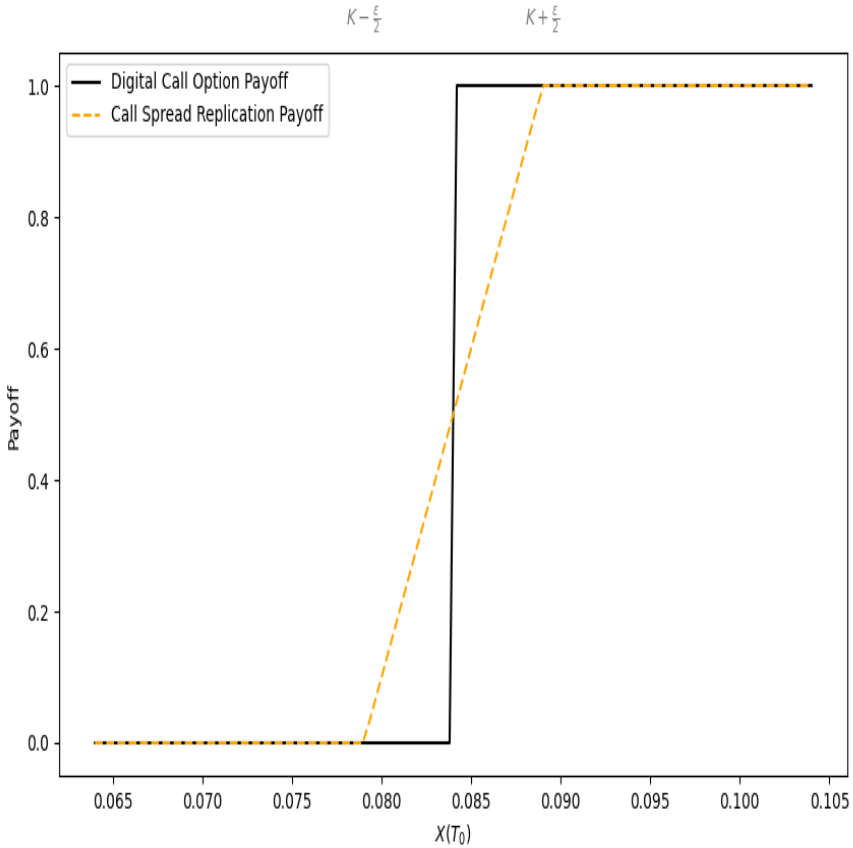
(c) Straddle (long 1 payer swaption @ 7.5% and long 1 receiver swaption @ 7.5%)

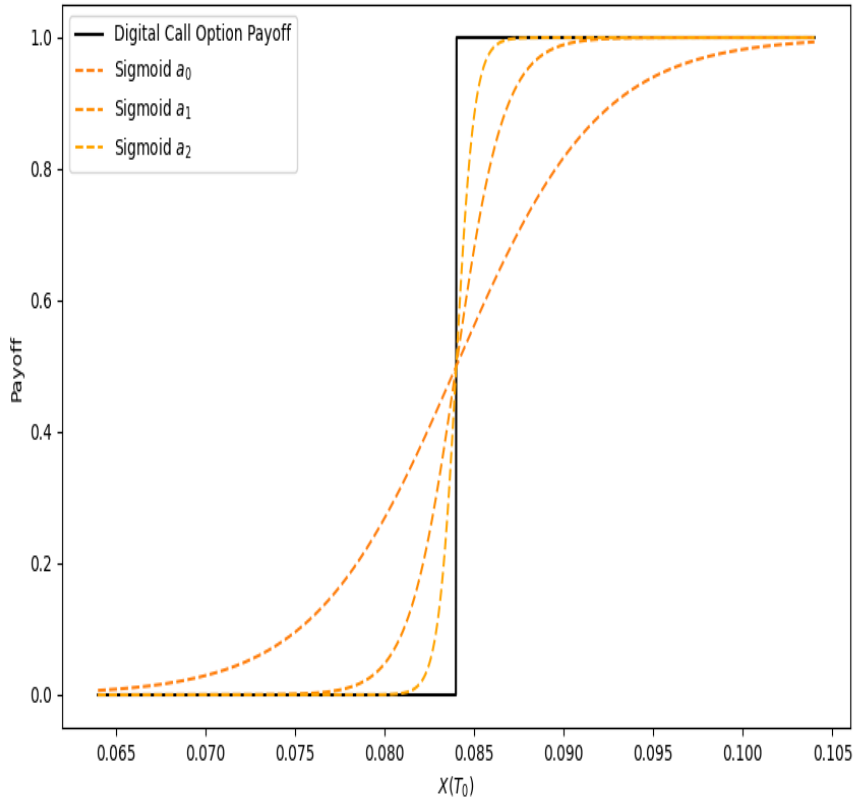


(d) Strangle (long 1 payer swaption @ 6.0% and long 1 receiver swaption @ 9.0%.)

Figure 18: Common option strategies as a function of the underlying swap rate. Settings are as in figure 16.

Appendix B.1: Smoothing Digital





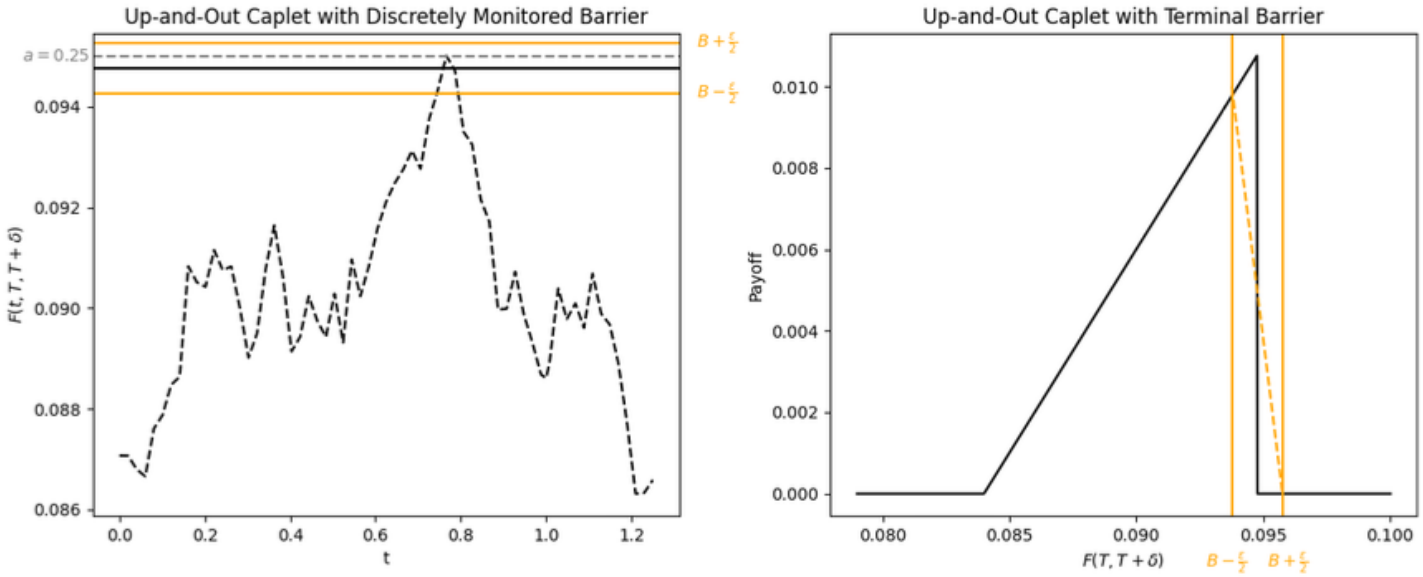


Figure 5: Smoothing techniques for Barrier options.

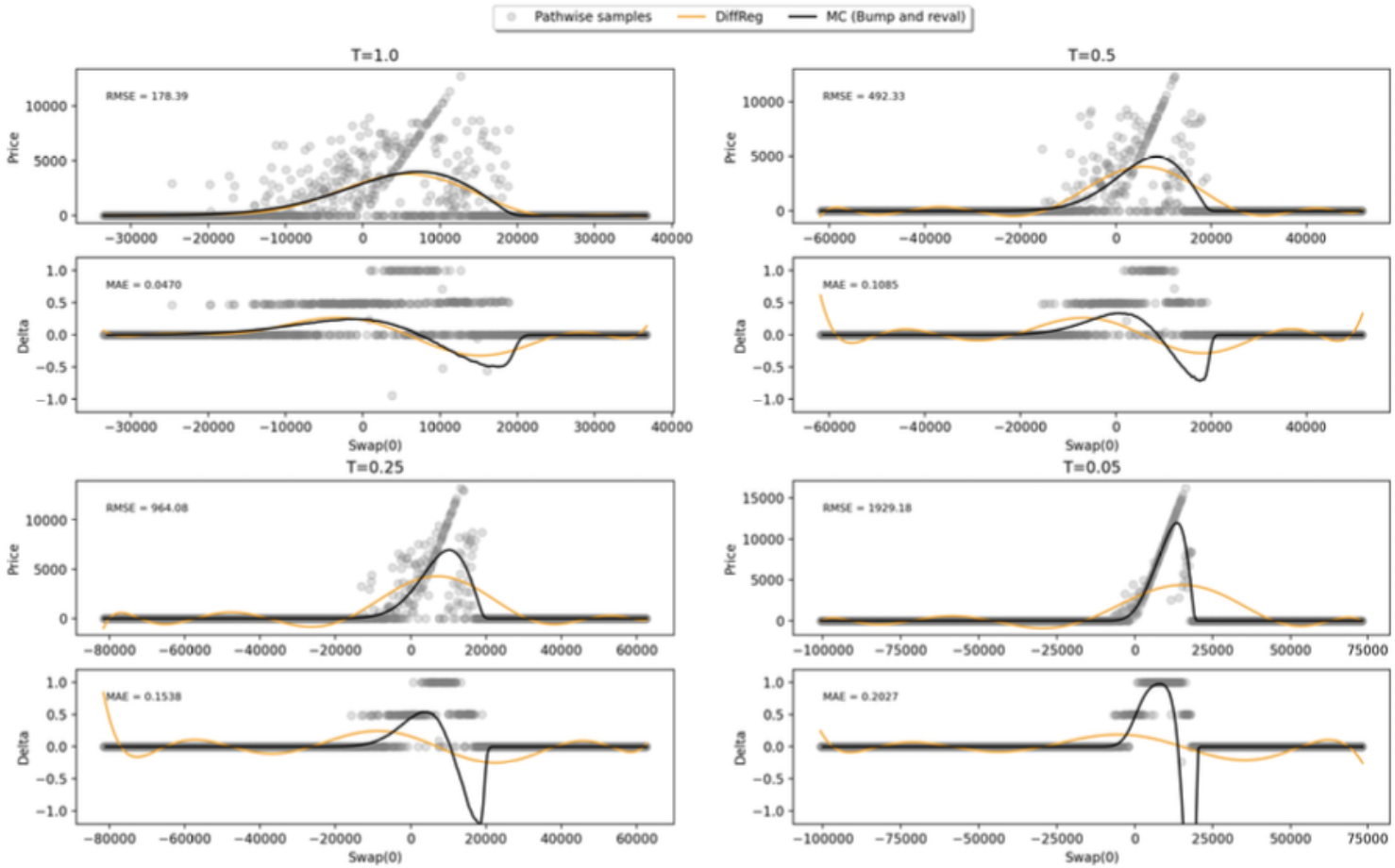


Figure 14: Delta and price estimation for varying maturities of European up-and-out barrier swaptions using differential regression trained on 1,024 training samples with 9 degrees, interactions, and $\alpha = 1.0$ for regularization. The barrier level is $B = 20,000$ with smoothing interval $\pm 5\%$ of H using sigmoid function. The swaption is monitored every 2.5 day, with a notional of 1,000,000 and strike rate 8.71%. The MC price and delta are estimated without any smoothing.

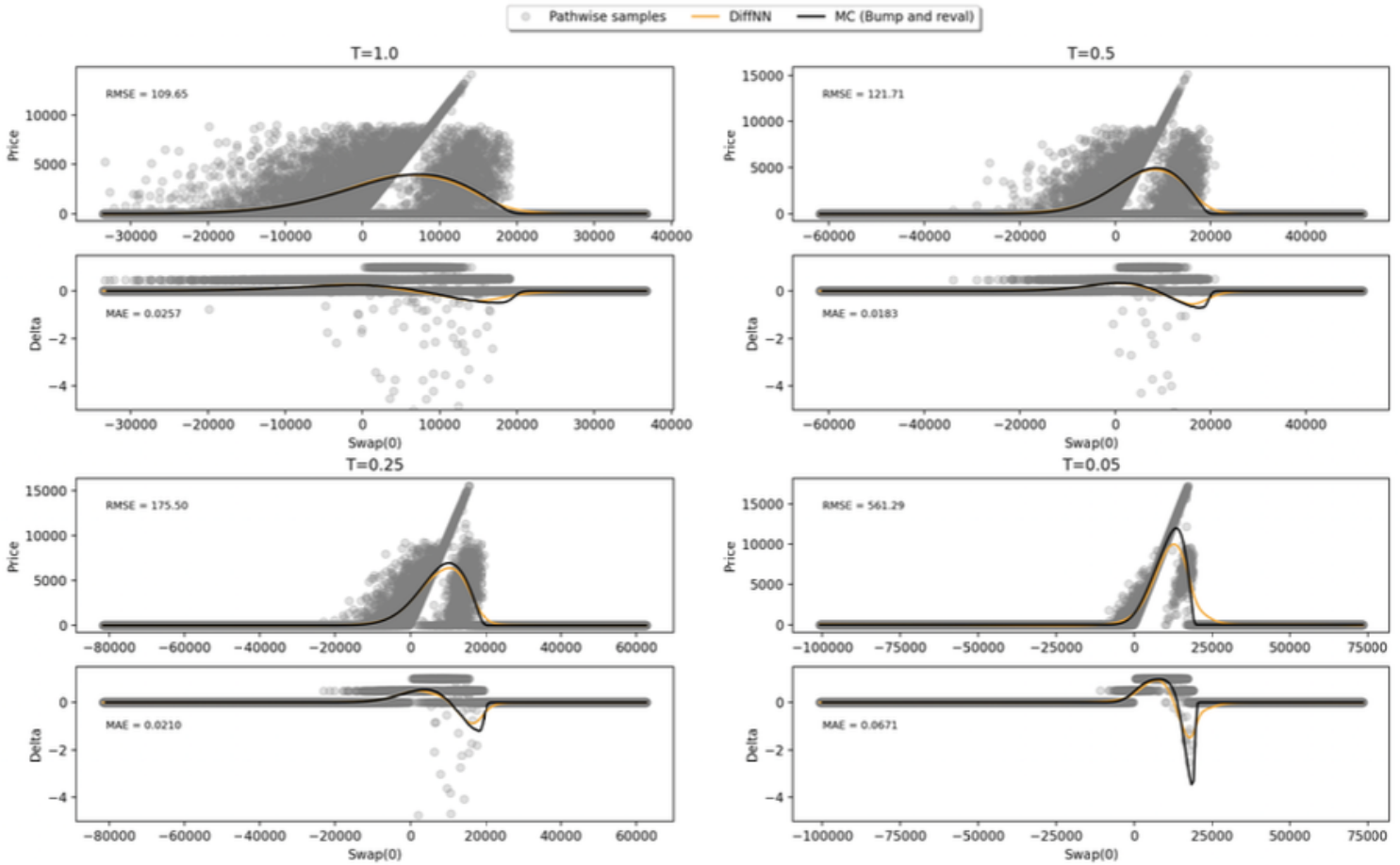


Figure 24: Delta and price estimation for varying maturities of European up-and-out barrier swaptions using differential neural network trained on 16,384 training samples. The barrier is $B = 20,000$ with smoothing interval $\pm 5\%$ of B using sigmoid function. The swaption is monitored every 2.5 day, with a notional of 1,000,000 and strike rate $K = 8.71\%$. The MC price and delta are estimated without any smoothing.

To find the optimal weights for the regression coefficient, β , we zero the gradient of the objective function wrt. β and solve for the estimate $\hat{\beta}$ which gives the adjusted normal equation:

$$
\begin{aligned}
& -\frac{2}{N}\left(\psi^\top\psi\beta - \psi^\top \boldsymbol{y} + \sum_j \alpha_j \left(D\psi_j^\top D\psi_j \beta - D\psi_j \boldsymbol{Z}_j\right)\right) = 0 \\
\Leftrightarrow \quad & \left(\psi^\top\psi + \sum_j \alpha_j D\psi_j^\top D\psi_j\right)\beta = \psi^\top \boldsymbol{y} + \sum_j \alpha_j D\psi_j^\top \mathbf{Z}_j \\
\Leftrightarrow \quad & \hat{\beta} = \left(\psi^\top\psi + \sum_j \alpha_j D\psi_j^\top D\psi_j\right)^{-1}\left(\psi^\top \boldsymbol{y} + \sum_j \alpha_j D\psi_j^\top \mathbf{Z}_j\right).
\end{aligned}
$$

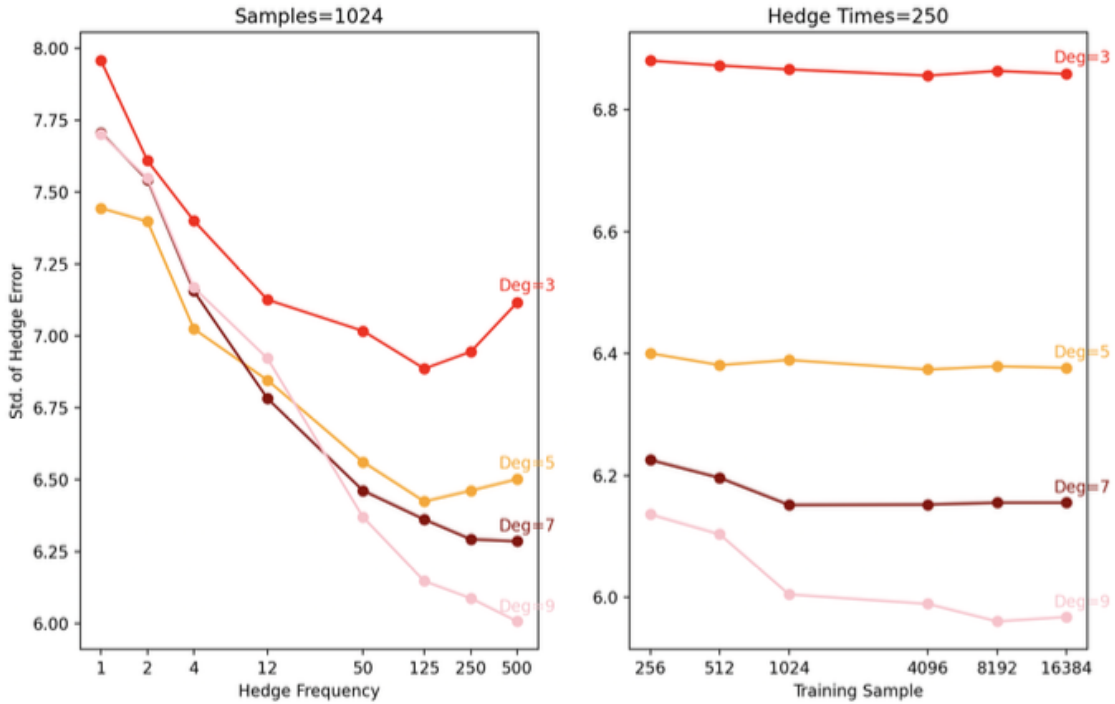


Figure 12: Convergence plots of hedge error for a 1Y6Y3M European payer swaption with strike rate $K = 8.71\%$ for different polynomials including interactions on log-log scale. Simulated from single spot value.

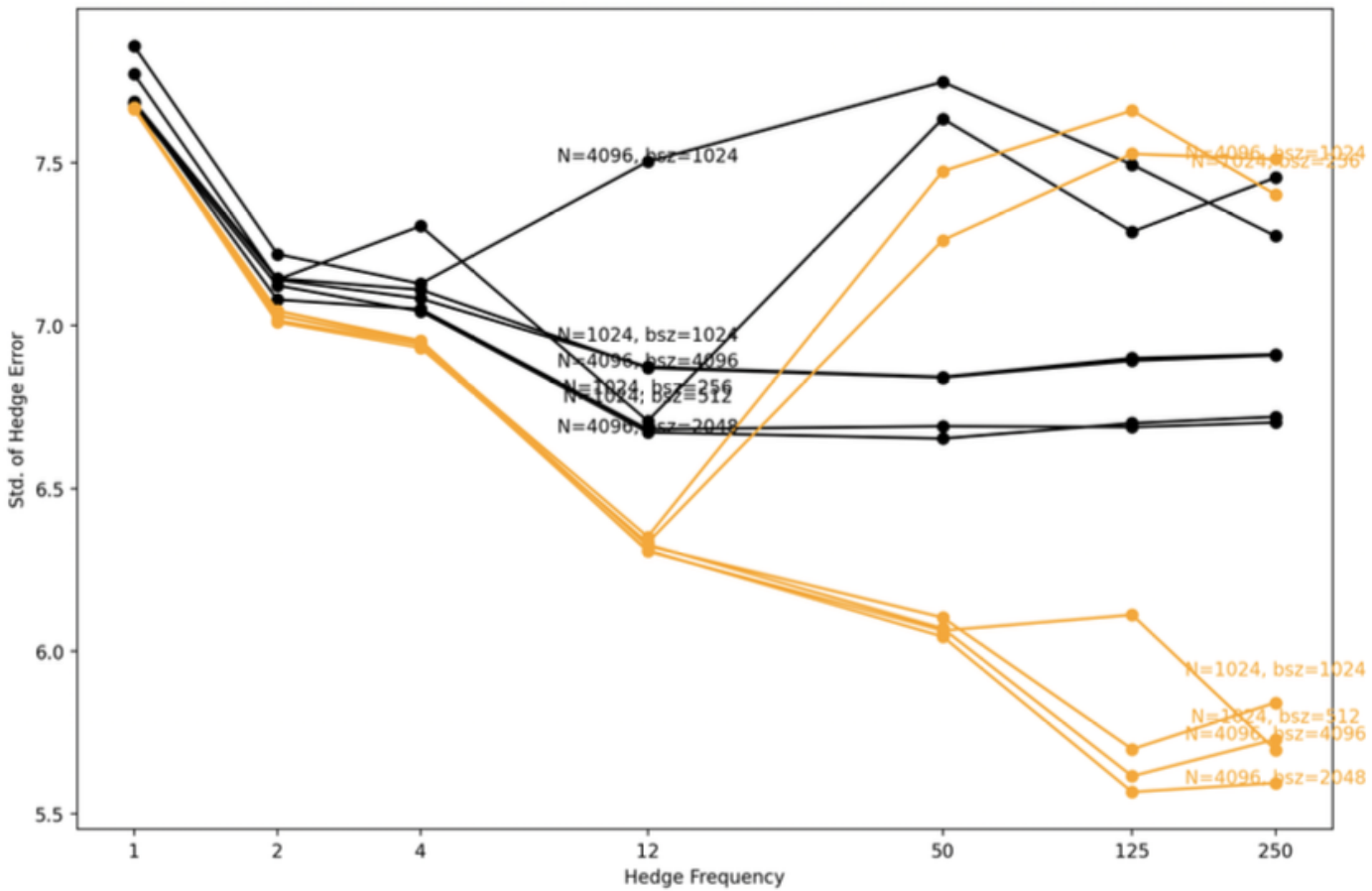


Figure 23: Convergence plot of hedge error for a 1Y6Y3M European payer swaption for standard-(black) and differential neural network (orange) on log-log scale. Simulated from single spot value with strike rate $K = 8.71\%$, varying training samples, $N = \{1024, 4096\}$, batch sizes $\{25\%, 50\%, 100\%\}$ of N and hedge frequency per year.

Proposition: Deferred payments valued back to the time of measurability

Let $t < T < T + \delta$ be three distinct times and let $CF(T + \delta, \boldsymbol{x}_T)$ be the cashflow that becomes known, i.e. measurable, at time T but paid at a deferred time $T + \delta$. The value of this cashflow at time t is

$$V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T+\delta)} CF(T+\delta, \boldsymbol{x}_T)\right] = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)} CF(T+\delta, \boldsymbol{x}_T) P(T, T+\delta)\right].$$

Proportion: Payment valued by pushing it forward

Let $t < T < T + \delta$ be three distinct times, and let $CF(T, \boldsymbol{x}_T)$ be the cashflow that become known, i.e. measurable, and paid at time T then the value of such a cashflow can equivalently be expressed by the identity

$$V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)} CF(T, \boldsymbol{x}_T)\right] = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T+\delta)} \frac{CF(T, \boldsymbol{x}_T)}{P(T, T+\delta)}\right].$$