Learning the *difference*

Differential Machine Learning for Option Pricing: A Monte Carlo Study under the Bates Model

Mathias Porsgaard

Faculty of Social Sciences
University of Copenhagen

1 Introduction

Pricing of financial derivates is a fundamental problem in quantitative finance. Even for plain instruments such as European options, imposing realistic dynamics for the underlying asset(s) - stochastic volatility and or jumps - does not admit closed form solutions in which practitioners must rely on numerical methods in complex mathematical models. Among these, Monte Carlo (MC) appears attractive due to its flexibility but accuracy comes at a high computational price due to the MC estimators standard error being root-N convergent.

This paper explores the integration of Monte Carlo simulation with differential machine learning techniques (MCDML) introduced in the Huge and Savine 2020 to improve the pricing of financial derivatives.

Accurate pricing of these instruments is crucial for risk management, investment strategies, and market efficiency.

Pricing derivative securities is a central problem in quantitative finance. Even for plain instruments such as European options, realistic dynamics for the underlying—stochastic volatility and/or jumps—rarely admit closed forms, forcing practitioners to rely on numerical methods. Among these, Monte Carlo (MC) remains attractive for its flexibility and dimension-insensitivity, but its computational cost is substantial: MC estimators converge at the root-N rate, with standard error $\mathcal{O}(N^{-1/2})$, so obtaining tight confidence intervals can be expensive.

This paper investigates whether *differential machine learning* (DML) can serve as a fast, accurate surrogate for MC-based pricing and risk. By *differential* we mean training a parametric approximator not only on prices but also on their sensitivities (Greeks), using algorithmic/automatic differentiation to provide derivative targets. The learning problem is then to approximate the pricing map

$$(x \mapsto P(x)), \qquad x = (\text{contract features, market/model parameters}),$$

while simultaneously matching selected components of $\nabla P(x)$ (e.g., Δ, Vega). Incorporating derivatives in the loss can improve data efficiency, enforce local smoothness consistent with financial theory, and—critically for risk—yield stable Greeks at inference.

Our approach is deliberately pragmatic. We generate training data by MC under a widely used non-Gaussian dynamics (e.g., stochastic volatility with jumps), employ standard variance-reduction techniques to obtain high-quality labels, and then compare two learned surrogates: (i) a price-only model ("baseline ML") and (ii) a differential model trained on prices *and* Greeks ("DML"). We evaluate them on accuracy versus MC, quality of Greeks, and computational efficiency.

Contributions and scope.

1. We build a Monte Carlo data-generation pipeline for a realistic option-pricing model (baseline Black–Scholes and one advanced model), including variance reduction and Greek estimators suitable for that model.
2. We implement and compare *price-only* ML against *differential* ML (prices + Greeks in the loss), using automatic differentiation to obtain derivative targets during data generation.
3. We propose a clear evaluation protocol: (a) RMSE/MAE versus high-precision MC with confidence intervals, (b) Greek accuracy and smoothness (finite-difference stability), (c) wall-clock inference speedup relative to MC, and (d) generalization across strikes/maturities and out-of-training parameter boxes.
4. We document when DML provides material gains over price-only learning and when it does not, highlighting limitations (e.g., jump-driven discontinuities) and practical trade-offs.

Why this is interesting. Production systems need fast pricing and *fast risk*. MC delivers accuracy but is too slow for large books and intraday recalculation; closed forms are rare under realistic dynamics. Learned surrogates can offer orders-of-magnitude speedups at inference, but naive price-only training often yields noisy Greeks. DML directly targets this bottleneck by using derivative information during

training, aiming to preserve the accuracy of MC while stabilizing Greeks and reducing the amount of MC data required.

What this paper is not. We do not attempt market calibration or historical backtests. Our focus is methodological: surrogate learning for pricing and risk under simulated dynamics, with transparent measurement of accuracy and compute.

2 Theoretical Framework

2.1 Financial Models for Option Pricing

To price financial derivatives, it is necessary to specify a model for the stochastic evolution of the underlying asset. While real markets are driven by complex, often non-stationary dynamics, simplified continuous-time models provide a tractable foundation for pricing and risk management. We begin with the classical Black–Scholes model, which admits a closed-form solution and thus serves as a natural benchmark for evaluating Monte Carlo and differential machine learning (MCDML) approaches.

2.1.1 Black-Scholes

In the BS framework, consider the price of a non-dividend paying asset that follows a Geometric Brownian Motion (GBM):

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{2.1.1}$$

where μ is the drift, $\sigma > 0$ is the volatility and W_t is standard Brownian motion under the measure $\mathbb{P}$.

The analytical solution to the Stochastic Differential Equation (SDE) above is,

$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t\right).$$

which implies that the stock price S_T is lognormally distributed,

$$S_T = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) T + \sigma\sqrt{T} Z\right), \quad Z \sim N(0,1).$$

Under the risk-neutral measure $\mathbb{Q}$, the drift μ is replaced by the risk-free rate r, and the time-0 price of a European call option with strike K and maturity T is

$$C_0 = e^{-rT}\, \mathbb{E}^{\mathbb{Q}}[\max(S_T - K, 0)]\,. \tag{2.1.2}$$

This expectation admits the closed-form Black–Scholes formula, but it also provides a natural Monte Carlo estimator that generalizes to models without analytic solutions. Consequently, we will use the BS model as a benchmark for assessing the accuracy of MCDML in both pricing and Greek estimation.

2.1.2 Bates

While the BS model serves as a benchmark for assessing the validity of MCDML method as a pricing and Greek approximator, it is easily simulated using regular MC. To properly assess the performance and potential of MCDML we need a model that requires full path simulation and can benefit from different discretization schemes and variance reduction techniques. For this purpose, we implement a version of the Bates (1996) model[1]. It is a combination of the jump diffusion process mixture model in Merton (1976) and the Heston (1993) stochastic volatility model. This setting allow volatility clustering, leptokurtic

[1]Technically, the European Call type options do have a semi-analytical solution, using the underlying characteristic function through a Fourier price transform.

returns and skewness, and volatility smiles - all empirical features the BS model cannot capture. The discontinuity inherent in jump processes provides an additional challenge for Greek estimation which we will cover later. Originally introduced for Deutsche Mark options, the dynamics are described by the following set of SDEs under the risk-neutral measure $\mathbb{Q}$:

$$\frac{dS_t}{S_t} = (r - \lambda\bar{k})dt + \sqrt{v_t}dW_t^S + kdN_t, \tag{2.1.3}$$

$$dN_t = \lambda dt + dJ_t, \tag{2.1.4}$$

$$\frac{dS_t}{S_t} = (r - \lambda\bar{k})dt + \sqrt{v_t}dW_t^S + dJ_t, \tag{2.1.5}$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t}dW_t^v, \tag{2.1.6}$$

$$dW_t^S dW_t^v = \rho dt, \tag{2.1.7}$$

where N_t is a Poisson process with intensity λ, k is the random jump size with mean $\bar{k}$ (subtracted to keep the discounted price a martingale), v_t is the instantaneous variance, $\kappa > 0$ is the rate of mean reversion, $\theta > 0$ is the long-term variance, σ is the volatility of volatility, and ρ is the correlation between the asset price and its variance. It follows that (**??**) is BS stock price dynamics augmented with a jump component, with expected relative jump-size $\bar{k} = \mathbb{E}[e^Y - 1] = e^{\mu_J + \frac{1}{2}\delta_J^2} - 1$ where $Y \sim \mathcal{N}(\mu_J, \delta_J^2)$ i.e. log-normal jump-sizes, while (**??**) is the Heston variance process, following a Cox-Ingersoll-Ross (cite) (CIR) (or mean-reverting square root process) ensuring positivity as long as the Feller condition $2\kappa\theta > \sigma^2$ holds. Lastly, (**??**) allows for correlation between the two Brownian motions, which is essential to capture the leverage effect observed in real markets.

Following Cont and Tankov (2003), applying Itô's lemma to (**??**) yields the SDE for the log-price process,

$$dX_t = \left(r - \lambda\bar{k} - \frac{1}{2}v_t\right)dt + \sqrt{v_t}dW_t^S + d\tilde{J}, \tag{2.1.8}$$

where $X_t = \log(S_t)$ and $\tilde{J} = \sum_{i=1}^{N_t} Y_i$ is a compound Poisson process with normally distributed jump sizes $Y_i \sim \mathcal{N}(\mu_J, \delta_J^2)$.

2.1.3 Greeks

In addition to pricing, a key aspect of option risk management involves computing sensitivities of the option price to various model parameters, commonly referred to as the "Greeks". The primary Greeks include Delta (Δ), Gamma (Γ), Vega (ν), Theta (Θ), and Rho (ρ).

WRITE SOMETHING REAL ABOUT WHY THESE ARE IMPORTANT FOR HEDGING ETC.

2.2 Monte Carlo Methods

The problem that arises, is that we often cannot compute the expectation of the discounted payoffs analytically as in (**??**). So instead, we simulate. As per Glasserman (2003), Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. In the context of Option pricing, we use Monte Carlo methods to estimate the expected value of the discounted payoff by simulating a large number of possible future asset price paths, computing the payoff for each path, and then averaging these payoffs. By a Law of Large Numbers (LLN) the monte carlo estimator converges to the true expected value as the number of simulations increases, that is,

$$\hat{C}_N = \frac{1}{N}\sum_{i=1}^{N} e^{-rT}\max(S_T^{(i)} - K, 0) \xrightarrow{a.s.} C, \quad \text{as } N \to \infty, \tag{2.2.1}$$

where $S_T^{(i)}$ is the simulated stock price at maturity for the i-th simulation under the respective model SDE.

2.2.1 Schemes

To simulate the underlying asset price paths, we discretize the SDEs using numerical schemes. The simplest and most commonly used scheme is the Euler-Maruyama method, which approximates the continuous-time SDE by a discrete-time process. For example, for the Black-Scholes model in (**??**), the Euler-Maruyama discretization over a time grid $0 = t_0 < t_1 < \ldots < t_M = T$ with step size $\Delta t = T/M$ is given by:

$$S_{t_{n+1}} = S_{t_n} + \mu S_{t_n} \Delta t + \sigma S_{t_n} \Delta W_n, \tag{2.2.2}$$

where $\Delta W_n \sim N(0, \Delta t)$ are independent increments of the Brownian motion. For more complex models like the Bates model in (**??**)-(**??**), we need to use more sophisticated schemes that can handle jumps and stochastic volatility, such as the Milstein scheme or jump-adapted schemes.

Potentially present both simulation algorithms for BS and Bates here, for Bates we use multiple.

2.2.2 Variance Reduction Techniques

One concern regarding Monte Carlo is efficiency. Since we are averaging random numbers, estimates are inherently prone to noise. It can be shown that the MC estimator for a R.V. Z, defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with finite variance $\sigma^2 = \text{Var}(Z) < \infty$, is unbiased, $\mathbb{E}[\hat{Z}_N] = \mathbb{E}[Z]$, and that a CLT applies for $N \to \infty$, such that the sample mean converges to a normal distribution with mean z and standard error $\sigma/\sqrt{N}$. Thus, the standard error converges towards zero at a rate of $\sqrt{N}/2$ and as such, to half the standard error (or double the precision), we need to increase the number of simulations by a factor of 4. This can be computationally expensive, and so variance reduction techniques are prominently used to reduce σ^2 and thereby improve precision without having to increase N.

This directly benefits the learning process of any ML model due to less variation in outputs resulting in the NN more easily learning the mapping, since there is less noise that needs to be filtered out, and it should be non-negligible wrt. computational cost. We utilize two simple variance reduction techniques as proposed in Glasserman (2003); Antithetic Variates and Control Variates.

Antithetic Variates: Antithetic Variates exploits negative correlation in pairs to reduce variance for a given number of simulations. It is most simply illustrated in the case of a R.V. $Z \sim U[0,1]$ in which case $Z' = 1 - Z$ is also uniformly distributed over $[0,1]$, and together they form an antithetic pair. This extends to any distribution through the inverse transform method as per Glasserman (2003), in fact for any distribution symmetric around the origin, $F^{-1}(1-\mu) = -F^{-1}(\mu)$ i.e. they have same magnitudes, but opposite signs[2], where F is the cumulative distribution function (CDF) of Z.

In the case of (normal) GBM, it is assumed that the increments are normally distributed, $Z \sim N(0,1)$ and we can use *antithetic variates*. That is, instead of simulating one $Z \sim N(0,1)$, we create antithetic pairs by drawing and Z and computing $Z' = -Z$. Given that these two draws are negatively correlated, averaging them cancels out some of the noise, leading to a lower variance estimate, i.e.,

$$\begin{aligned}
\text{Var}\left(\frac{Z+Z'}{2}\right) &= \frac{1}{4}\left(\text{Var}(Z) + \text{Var}(Z') + 2\text{Cov}(Z,Z')\right) \\
\text{cov}(Z,Z') &= -\sigma^2 \\
&= \frac{1}{4}\left(2\sigma^2 - 2\sigma^2\right),
\end{aligned}$$

Either actually write the math correctly, or go with footnote and just refer to glasserman, it is really simple so i think that will suffice. So for each simulation i, in the case of payoffs, we compute two

[2] A formal comparison between variance estimators would warrant two draws for a pair, in which case the antithetic variate estimator has smaller variance provided the pair elements are negatively correlated i.e. $\text{Var}(Z+Z') < 2\text{Var}(Z)$ if $\text{cov}(Z,Z') < 0$ - see Glasserman (2003) for details, but note that a simple condition ensuring this is monotonicity of the mapping from inputs to outputs. In the case of symmetric distributions we can simply flip the sign to ease computational burden by 50 % immediately, along with lowering the variance for the same relative amount of draws in the regular MC case, provided they are negatively correlated.

payoffs, $\Pi^{(i)}$ using Z_i and $\Pi^{(i)'}$ using $-Z_i$, and then average them:

$$\hat{C}_N = \frac{1}{2N} \sum_{i=1}^{N} \left(\Pi^{(i)} + \Pi^{(i)'} \right).$$

Control Variates: Say we want to estimate $E(X)$ but that X is noisy. We also observe another (related) variable Y, for which we know $E(Y)$. If X and Y are correlated, we can use Y to reduce the noise in our estimate of $E(X)$. The idea is to consider the new variable,

$$X' = X - \beta(Y - E(Y)),$$

Now, we don't know the Option price exactly, but we do know the expectation of the underlying stock price itself, $E(S_T) = S_0 e^{rT}$ which we can use as a control variate. Another often used control variate when outside the BS framework, is the BS price itself.

Reference glasserman. Talk about it essentially being a regression problem when parameters are unknown. And that the effectiveness of control variates can vary greatly depending on params (Table 4.1). Then look at his examples. Underlying asset, tractable dynamics (i want to do this), "any instrument that serves as a an effective hedge for Y is an effective control variate".

No arbitrage.

2.2.3 Greek Estimation

2.3 Machine Learning

Monte Carlo is flexible, but slow (especially at high precision). Say we want Option prices for many different sets of parameters (e.g. S_0, K, T, r, σ), the idea is then to use Machine Learning to learn the mapping from model parameters (e.g. S_0, K, T, r, σ) to Option prices. This is a regression problem, where we want to learn a function $f : \mathbb{R}^5 \to \mathbb{R}$, that maps the 5 input parameters to the Option price. Without ML, we would have to rerun the simulation for each new set of parameters. With ML, we can train a model on a large dataset of simulated Option prices, and then use the trained model to predict Option prices for new sets of parameters almost instantly.

One-path pairs -¿ unbiased but very noisy estimates.

2.3.1 Standard Surrogate Models

A standard approach is to use a feedforward Neural Network (NN) as a surrogate model. The NN is trained on a dataset of simulated Option prices, where the input features are the model parameters (state variables) (e.g. S_0, K, T, r, σ) and the target variable is the Option price. The NN learns to approximate the mapping from input features to Option prices by minimizing a loss function, typically the Mean Squared Error (MSE) between the predicted and true Option prices. This NN is a parametric function $f_\theta : \mathbb{R}^d \to \mathbb{R}$, where θ are the parameters (weights and biases) of the network, and d is the number of input features. The architecture of the NN consists of multiple layers of interconnected neurons, where each neuron applies a non-linear activation function to a weighted sum of its inputs. Due to the Universal Approximation Theorem, a sufficiently large NN can approximate any continuous function to arbitrary accuracy, making it a powerful tool for function approximation in high-dimensional spaces.

Formally, given a training dataset $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ are the input features and $y_i \in \mathbb{R}$ are the target Option prices, the NN is trained to minimize the following loss function:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2, \tag{2.3.1}$$

where f_θ is the NN with parameters θ. Once trained, the NN can be used to predict Option prices for new sets of parameters by simply feeding the input features into the network.

2.3.2 Differential Machine Learning

The core idea presented in Huge and Savine (2020) is to enhance the training of a Neural Network (NN) by computing pathwise gradients using AAD Giles and Glasserman (n.d.) and augmenting the loss function, not unlike other regularization methods as noted in Frandsen, Pedersen, and Poulsen (2022). (ref some paper, maybe lasso, maybe the springer easy paper).

The augmented loss function incorporates both the standard MSE loss for Option prices and an additional term that penalizes discrepancies between the NN's predicted gradients (Greeks) and the true gradients obtained from pathwise differentiation. Extending (**??**), the augmented loss function can be expressed as:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big((f_\theta(x_i) - y_i)^2 + \lambda\|\nabla_x f_\theta(x_i) - g_i\|^2\Big), \tag{2.3.2}$$

where $\nabla_x f_\theta(x_i)$ represents the gradient of the NN output with respect to the input features, g_i are the true Greeks obtained via pathwise differentiation, and λ is a hyperparameter that controls the trade-off between fitting the Option prices and matching the gradients.

This encourages the NN to not only fit the Option prices accurately but also to capture the sensitivity of the prices with respect to the input parameters, leading to improved generalization, robustness and faster convergence rates. The additional information residing in the gradients can loosely be interpreted as the NN not only learning the function mapping from parameters to prices, but also learning the local geometry of this mapping, i.e., how small changes in the input parameters affect the output prices.

2.3.3 Integration with Monte Carlo

How does this relate to Monte Carlo? Under the assumption of a known functional form of the underlying SDE's and payoff structure, Monte Carlo simulation can provide unbiased estimates of both Option prices and their corresponding Greeks. This provides a controlled environment where data amount is unlimited. The claim of unbiased estimates are important. As in Huge and Savine (2020), we train models on so-called one-path pairs, i.e. for a given draw of initial parameters we compute one path from which the Monte Carlo estimate is computed. This is very different from traditional MC estimation, where we would compute the average over many paths for a given set of parameters to increase accuracy.

This does however provide unbiased, though very noisy estimates, which puts high demands on the model's ability to generalize from limited and noisy data. The differential training approach helps mitigate this issue by leveraging the additional information contained in the pathwise gradients, allowing the NN to learn more effectively from each training example.

This is done in batches of (256, 512, 1024, 8048, pairs) per epoch, in which the training of a model amounts to approximately one MC price, simulation count wise.

The framework lends itself to online training - by online we mean step-wise batch training on continuously simulated data. This could in theory extend to reinforcement learning, where the model is continuously updated as new data arrives, allowing it to adapt to changing market conditions in real-time.

Hmmm. What about the underlying SDE that are simulated, those would have to change?

What about real-time data - could it approximate with no underlying SDE (CBS thesis?).

What about real-time hedging?

2.4 Evaluation Metrics

To assess the performance of our models, we will use several evaluation metrics that capture different aspects of accuracy and reliability in both pricing and Greek estimation. These metrics include:

- Clock time / Computational Efficiency: We will measure the time taken to train the model and to make predictions, as computational efficiency is crucial in practical applications.
- **Pricing Accuracy:** We will evaluate the accuracy of the option price predictions using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2).
- **Greek Estimation Accuracy:** Similar to pricing accuracy, we will assess the accuracy of Greek estimates (Delta, Gamma, Vega, etc.) using MAE, RMSE, and R^2.
- Smoothness of Greek Surfaces: We will visually inspect the smoothness of the estimated Greek surfaces over a range of input parameters to ensure that the model captures the expected behavior.
- **Mean Absolute Error (MAE):** This metric measures the average absolute difference between the predicted and true values. It is defined as

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \tag{2.4.1}$$

  where $\hat{y}_i$ is the predicted value, y_i is the true value, and N is the number of samples.
- **Root Mean Squared Error (RMSE):** This metric measures the square root of the average squared differences between predicted and true values. It is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}. \tag{2.4.2}$$

- **Coefficient of Determination (R^2):** This metric indicates the proportion of variance in the dependent variable that is predictable from the independent variables. It is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{2.4.3}$$

  where $\bar{y}$ is the mean of the true values.
- **Relative Error:** This metric measures the error relative to the true value, providing insight into the scale of errors. It is defined as

$$\text{Relative Error} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{|y_i|}. \tag{2.4.4}$$

We will benchmark the MCDML model against standard Monte Carlo estimates and traditional neural network surrogates using these metrics. This comprehensive evaluation will help us understand the strengths and limitations of the MCDML approach in the context of option pricing and Greek estimation.

References

Merton, Robert C. (Jan. 1976). "Option Pricing When Underlying Stock Returns Are Discontinuous". In: *Journal of Financial Economics* 3.1-2, pp. 125–144. ISSN: 0304405X. DOI: 10.1016/0304-405X(76)90022-2. (Visited on 10/17/2025).

Heston, Steven L. (1993). "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options". In: *The Review of Financial Studies* 6.2, pp. 327–343. ISSN: 0893-9454. JSTOR: 2962057. (Visited on 10/17/2025).

Bates, DS (Mar. 1996). "Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options." In: *Review of Financial Studies* 9.1, p. 69. ISSN: 0893-9454. DOI: 10.1093/rfs/9.1.69. (Visited on 10/17/2025).

Cont, Rama and Peter Tankov (Dec. 2003). *Financial Modelling with Jump Processes*. New York: Chapman and Hall/CRC. ISBN: 978-0-429-20478-4. DOI: 10.1201/9780203485217.

Glasserman, Paul (2003). *Monte Carlo Methods in Financial Engineering*. Stochastic Modelling and Applied Probability 53. New York: Springer. ISBN: 978-0-387-21617-1.

Huge, Brian and Antoine Savine (Sept. 2020). *Differential Machine Learning*. DOI: 10.48550/arXiv.2005.02347. arXiv: 2005.02347 [q-fin]. (Visited on 10/17/2025).

Frandsen, Magnus Grønnegaard, Tobias Cramer Pedersen, and Rolf Poulsen (Mar. 2022). "Delta Force: Option Pricing with Differential Machine Learning". In: *Digital Finance* 4.1. read, pp. 1–15. ISSN: 2524-6186. DOI: 10.1007/s42521-021-00041-7. (Visited on 10/17/2025).

Giles, Michael and Paul Glasserman (n.d.). "Smoking Adjoints: Fast Evaluation of Greeks in Monte Carlo Calculations". In: ().