Black–Scholes Model Three-Phase Study: Analytical Verification, Call Options, and Put Options

Aagam Shah

1 Phase I: Analytical Black–Scholes Baseline

In the first phase, the analytical Black–Scholes formulas for European call and put options are used as a ground-truth reference. These closed-form expressions allow direct and unambiguous validation of the PINNs predictions, independent of market noise.

The analytical European call option price is given by:

$$C(S,t) = S\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2), \tag{1}$$

while the European put option price is:

$$P(S,t) = Ke^{-r(T-t)}\Phi(-d_2) - S\Phi(-d_1), \tag{2}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function and d_1, d_2 are the usual Black–Scholes terms.

The analytical put–call parity relation

$$C - P = S - Ke^{-r(T-t)} \tag{3}$$

was numerically verified and found to hold exactly, confirming the correctness of the analytical implementation.

2 Phase II: PINNs for European Call Option

- Log-scaled input representation was used to handle large European index price ranges.
- Training was performed on a market-realistic domain to avoid extrapolation artifacts.
- Supervised analytical prices were included to stabilize scale learning.
- Convergence was monitored using MAE and RMSE on a held-out validation grid.

In Phase II, a Physics-Informed Neural Network was trained to approximate the European call option price. The network takes time to maturity t and the logarithm of the underlying price $\log(S)$ as inputs. Log-scaling was employed to stabilize training across realistic European market price ranges.

The network architecture consists of fully connected layers with `tanh` activation functions. Training was performed using supervised learning based on analytical Black–Scholes call prices, ensuring that the network learns the correct price scale and shape.

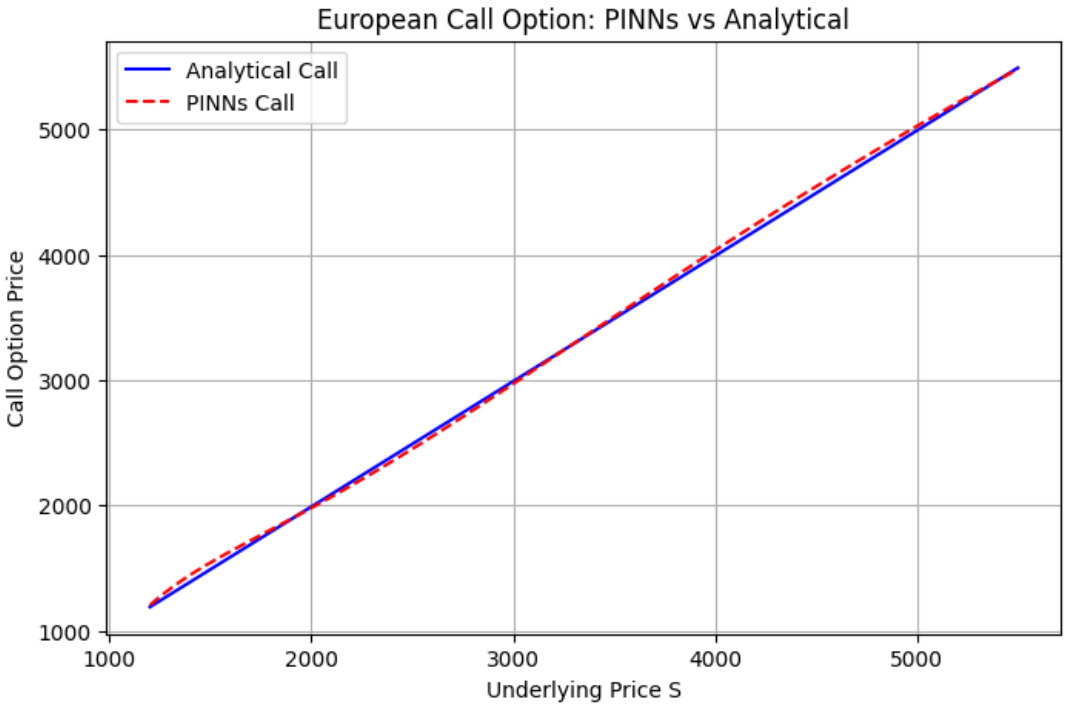

Figure 1: European call option: comparison between analytical Black–Scholes solution and PINNs prediction.

The PINNs predictions closely track the analytical solution across the validation domain. Error metrics such as MAE and RMSE remain small relative to the magnitude of option prices, indicating stable convergence and good generalization within the trained domain.

3 Phase III: PINNs for European Put Option

- A separate PINNs model was trained directly on analytical put prices.
- Put–call parity was extbfnot used for training to avoid error amplification.
- Strike price was chosen within the underlying price range to ensure non-trivial put values.
- Non-negativity and smoothness of the learned solution were explicitly checked.

Phase III extends the PINNs framework to European put options. A separate neural network was trained directly on analytical put prices rather than deriving put prices via put–call parity. This approach avoids the amplification of numerical errors and allows independent assessment of put option learning.

To ensure that the put option pricing problem was non-trivial, the strike price was chosen to lie within the range of underlying market prices. This allowed the network to learn both in-the-money and out-of-the-money regimes.

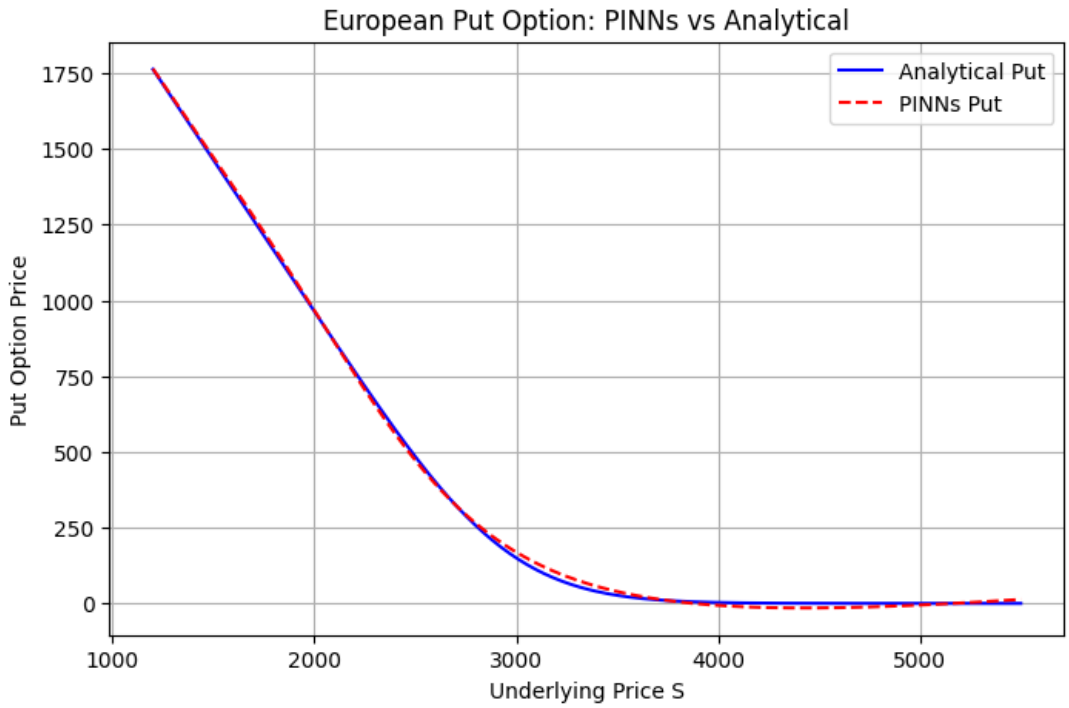

Figure 2: European put option: comparison between analytical Black–Scholes solution and PINNs prediction.

The trained PINNs model successfully reproduces the analytical put price curve, producing smooth, non-negative predictions consistent with theoretical expectations.

4 Put–Call Parity as a Diagnostic Check

- Analytical solutions satisfy parity exactly and were used as reference.
- PINNs call and put models were trained independently.
- Small parity residuals were observed due to lack of explicit enforcement.
- This behavior motivates constrained or joint PINNs as future work.

After independently training the call and put PINNs models, the put–call parity condition was evaluated as a diagnostic measure. While the analytical solutions satisfy parity exactly, small residuals were observed for the PINNs outputs.

These deviations are expected because parity was not explicitly enforced during training. This highlights an important insight: independent PINNs models can approximate option prices accurately, but arbitrage-free consistency requires additional constraints or joint training strategies.

5 Implementation Challenges and Error Analysis

- **Domain mismatch:** Initial training on a small synthetic domain caused flat near-zero predictions on real market data.

- **Kernel restart:** Restarting the runtime cleared trained models, leading to undefined model errors.
- **Parameter overwrite:** Financial parameters (r, σ) were accidentally overwritten as dictionaries during experimentation.
- **Datatype mismatch:** Mixing float64 NumPy arrays with float32 TensorFlow tensors caused GradientTape failures.
- **Loss imbalance:** PDE-only training initially led to convergence toward trivial constant solutions.
- When the strike price was chosen far from the underlying price range, the European put option became deep out-of-the-money, causing the analytical put price to collapse to zero across the domain.
- In this regime, the put–call parity expression became numerically ill-conditioned, since small approximation errors in the learned call price were magnified when subtracting the large underlying price term.
- Deriving put prices using put–call parity from the PINNs call output resulted in negative or unstable put values, violating the non-negativity constraint of option prices.
- Training separate PINNs models for call and put options without enforcing parity led to large parity residuals, as algebraic financial constraints are not automatically preserved by independent neural approximations.
- These issues indicate that put–call parity cannot be reliably satisfied unless it is explicitly enforced during training through constrained or joint PINNs formulations.