Pricing Commodity Options Using Monte Carlo Simulation with Python

Nikita Lavrentyev

November 5, 2024

1 Introduction to Option Pricing in Commodity Markets

As oil prices fluctuate amid uncertainty surrounding the Middle East and U.S. elections, traders are reminded of how unpredictable the energy markets can be. Commodity markets, such as oil, are particularly susceptible to sudden price swings due to a range of external factors, including geopolitical tensions, supply chain disruptions, and shifts in global demand. In such a volatile environment, accurately pricing options is crucial for managing risk and protecting against sudden price shifts.

Traditional models like Black-Scholes, which assume stable volatility, often fail to capture the complexity of real-world commodity markets. In these markets, volatility itself can fluctuate drastically over time, leading to unexpected risk exposures for traders. Monte Carlo simulations, on the other hand, offer a more flexible approach. By simulating thousands of potential price paths, these models provide a realistic and adaptable way to price options, helping firms navigate the challenges of today's unpredictable energy landscape.

2 Theory: Monte Carlo Simulation with Geometric Brownian Motion (GBM)

Monte Carlo simulation generates thousands of potential price paths by introducing randomness into modeled market movements over time. This approach provides a comprehensive view of possible future outcomes, which is particularly valuable in commodity markets like oil, where prices don't follow simple patterns and can experience sudden spikes due to geopolitical crises or supply disruptions.

The simulation typically relies on a stochastic process called Geometric Brownian Motion (GBM) to model price movements. GBM assumes that an asset's price is influenced by:

- **Drift**: The expected upward or downward movement over time, representing the general trend.
- **Volatility**: The degree of random fluctuation around the trend, representing market uncertainty and variability.

In GBM, prices evolve proportionally over time, ensuring they remain positive and follow a log-normal distribution, which allows for large price swings. The continuous form of the GBM formula is:

$$dS = S(r\,dt + \sigma\,dW)$$

where:

- S: The current price of the asset,
- r: The risk-free interest rate (drift),
- σ: The volatility of the asset,
- dW: A Wiener process representing randomness.

For Monte Carlo simulations, we use the discretized version of GBM, which allows us to simulate daily price changes:

$$S_{t+1} = S_t \times \exp\left((r - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\, Z\right)$$

where:

- Δt: The time step (in years),
- Z: A random variable drawn from a standard normal distribution.

This discretized formula generates a variety of potential future price paths, each reflecting unique outcomes based on historical price characteristics. By averaging the discounted payoffs across all paths, Monte Carlo models yield a realistic estimate of option prices, capturing the randomness and volatility that characterize these markets.

3 Python Implementation of Monte Carlo Simulation for Option Pricing

This implementation uses Monte Carlo with GBM to price a European call option on a commodity like oil. The option payoff at maturity depends on whether the commodity's price exceeds the strike price K. We use several formulas to complete this process:

3.1 Calculating Annualized Volatility

Volatility (σ) is a crucial input in the GBM model, as it reflects the expected degree of price variability. We calculate annualized volatility from historical daily returns:

$$\sigma_{\text{annual}} = \text{std(daily returns)} \times \sqrt{252}$$

where std(daily returns) is the standard deviation of daily percentage changes in price, and multiplying by $\sqrt{252}$ annualizes it for a one-year period.

3.2 Payoff Calculation for a European Call Option

For a European call option, the payoff at maturity is:

$$\text{Payoff} = \max(S_T - K, 0)$$

where:

- S_T: Simulated commodity price at maturity,
- K: Strike price of the option.

If $S_T > K$, the payoff is positive; otherwise, it is zero.

3.3 Discounting Payoff to Present Value

The option price represents the expected payoff at maturity, discounted to today's value using the risk-free rate r:

$$\text{Option Price} = e^{-rT} \cdot \mathbb{E}[\text{Payoff}]$$

where:

- T: Time to maturity (in years),
- $\mathbb{E}[\text{Payoff}]$: Average payoff across all simulated paths.

By calculating the present value of the payoff, we obtain a fair price for the option today, accounting for the time value of money.

3.4 Python Code Example: Monte Carlo for European Call Option

Step 1: Load and Prepare Data

```
import pandas as pd

def load_and_prepare_data(file_path):
    data = pd.read_excel(file_path, skiprows=1)
    data.columns = ['Date', 'Price', 'Open', 'High', 'Low', 'Volume', 'Change %']
    data['Date'] = pd.to_datetime(data['Date'])
    data.set_index('Date', inplace=True)
    data['Price'] = pd.to_numeric(data['Price'], errors='coerce')
    data.dropna(subset=['Price'], inplace=True)
    return data
```

Explanation: This function loads and cleans historical price data by setting the `Date` as the index and converting the `Price` column to numeric. This prepares the data for analysis.

Step 2: Calculate Volatility

```
import numpy as np

def calculate_volatility(data):
    data['Return'] = data['Price'].pct_change()
    return np.std(data['Return'].dropna()) * np.sqrt(252)  # Annualized volatility
```

Explanation: This function calculates annualized volatility, which is essential for the Monte Carlo simulation to reflect real-world price fluctuations.

Step 3: Monte Carlo Simulation for Option Pricing

```
def monte_carlo_option_pricing(S0, K, T, r, sigma, n_simulations=10000, n_steps=252):
    dt = T / n_steps  # Time step in years
    price_paths = np.zeros((n_simulations, n_steps+1))
    price_paths[:, 0] = S0

    # Generate random price paths
    for t in range(1, n_steps+1):
        z = np.random.standard_normal(n_simulations)
        price_paths[:, t] = price_paths[:, t-1] * np.exp((r - 0.5 * sigma**2) * dt + sigma *
            np.sqrt(dt) * z)

    # Calculate option payoff
    payoff = np.maximum(price_paths[:, -1] - K, 0)
    return np.exp(-r * T) * np.mean(payoff)
```

Explanation: This function runs the Monte Carlo simulation. It generates multiple potential future paths for the commodity price, calculates the payoff for each path, and averages them, discounted to today's value.

Step 4: Main Execution Block

```
file_path = r'/path/to/your/data/Crude_Oil_WTI_Futures_Historical_Data.xlsx'
K = 75          # Strike price
T = 1.0         # Time to maturity in years
r = 0.035       # Risk-free rate

# Run steps in sequence
historical_data = load_and_prepare_data(file_path)
sigma = calculate_volatility(historical_data)
S0 = historical_data['Price'].iloc[-1]

# Calculate option price
option_price = monte_carlo_option_pricing(S0, K, T, r, sigma)
print(f"European Call Option Price (Monte Carlo): {option_price:.2f}")
```

Explanation: The main block executes each function sequentially. It loads the data, calculates historical volatility, sets the initial price (`S0`), and then runs the Monte Carlo simulation to determine the European call option price.

4 Enhancing the Simulation: Stochastic Volatility

In real-world commodity markets like oil, price movements often defy simple trends. While the basic Monte Carlo simulation using Geometric Brownian Motion (GBM) assumes constant volatility, this simplification can miss crucial market dynamics. In reality, volatility is not static; it fluctuates based on market conditions, geopolitical events, and economic shifts. To account for these factors, we can enhance the Monte Carlo simulation by incorporating *stochastic volatility*.

4.1 What is Stochastic Volatility?

Stochastic volatility refers to a model where volatility is allowed to vary over time instead of being a fixed value. This approach better mirrors how commodity prices behave, with periods of relative calm interspersed with sudden spikes or drops.

Example: Think of the oil market when a sudden geopolitical crisis arises. Prices may surge, but more importantly, the level of uncertainty (volatility) increases. Traditional models that assume constant volatility can't fully capture this shift, potentially leading to mispriced options.

4.2 Why Does Stochastic Volatility Matter?

Key Benefits:

- **Captures Real-World Behavior**: By allowing volatility to change over time, we reflect periods of "volatility clustering," where high volatility tends to be followed by high volatility, and vice versa.
- **Accurate Risk Representation**: Models with stochastic volatility are better equipped to price options during turbulent market periods, reducing the risk of underpricing options when volatility is surging.
- **Improves Hedging Strategies**: Firms that rely on accurate pricing to hedge their risk will find these models offer a more responsive tool during market shifts.

4.3 Introducing Stochastic Models

A widely-used model for simulating stochastic volatility is the *Heston Model*. Unlike GBM, the Heston model assumes:

- **Volatility evolves as a separate process**, potentially correlated with the asset's price.
- **Mean reversion**: Volatility tends to revert to a long-term average over time.
- **Volatility of volatility**: Captures sudden shifts or spikes in market uncertainty.

To understand stochastic volatility, imagine a spring that represents the level of market volatility. The parameter κ determines how tightly the spring pulls the current volatility V_t back to its long-term average θ. The term σ_{vol} controls how erratically the spring stretches, representing sudden changes in volatility. The stochastic component dW_2 adds a random element, showing that even with this mean-reversion tendency, the market remains unpredictable.

Equation Overview:

$$dV_t = \kappa(\theta - V_t)\,dt + \sigma_{\text{vol}}\sqrt{V_t}\,dW_2$$

Where:

- V_t is the current volatility.
- κ controls the speed of reversion to the average (θ).

- σ_{vol} represents how volatile the volatility itself is.
- dW_2 is a stochastic term, which may correlate with the asset price process.

This equation ensures that volatility fluctuates dynamically, capturing the real-world behavior of commodity prices, where volatility tends to revert but can change unexpectedly due to market events.

4.4 Python Code: Monte Carlo Simulation with Stochastic Volatility

To implement stochastic volatility using the Heston model, we break the code into modular functions for clarity. Each function serves a unique purpose, from initializing parameters to simulating price paths and calculating option prices.

Step 1: Initialize Heston Parameters and Volatility Paths

```
import numpy as np

def initialize_heston_parameters(n_simulations, n_steps, V0=0.2, kappa=2.0, theta=0.2,
    sigma_vol=0.1):
    vol_paths = np.zeros((n_simulations, n_steps+1))
    vol_paths[:, 0] = V0  # Set initial volatility
    return vol_paths, kappa, theta, sigma_vol
```

Explanation: This function initializes the parameters for the Heston model, which accounts for stochastic volatility. Parameters include: - `V0`: Initial volatility. - `kappa`: Mean reversion speed, controlling how quickly volatility returns to the average. - `theta`: Long-term average volatility. - `sigma_vol`: Volatility of volatility, allowing for sudden volatility changes.

Step 2: Simulate Stochastic Paths for Price and Volatility

```
def simulate_stochastic_paths(S0, K, T, r, sigma, n_simulations, n_steps, vol_paths, kappa,
    theta, sigma_vol):
    dt = T / n_steps
    price_paths = np.zeros((n_simulations, n_steps+1))
    price_paths[:, 0] = S0

    # Generate two sets of random variables for simulating the stochastic processes
    for t in range(1, n_steps+1):
        z1 = np.random.standard_normal(n_simulations) # Random variable for price movement
        z2 = np.random.standard_normal(n_simulations) # Random variable for volatility
            changes

        # Update volatility path using the Heston model
        vol_paths[:, t] = np.maximum(
        vol_paths[:, t-1] + kappa * (theta - vol_paths[:, t-1]) * dt  # Mean reversion to
            long-term average
        + sigma_vol * np.sqrt(vol_paths[:, t-1] * dt) * z2,  # Random volatility
            fluctuations
        0  # Ensure volatility stays non-negative
    )


```

```
        # Update the price path using current volatility
        price_paths[:, t] = price_paths[:, t-1] * np.exp(
        (r - 0.5 * vol_paths[:, t-1]) * dt  # Drift component
        + np.sqrt(vol_paths[:, t-1] * dt) * z1  # Volatility component
    )

    return price_paths, vol_paths
```

Explanation: This function simulates the price and volatility paths:

- **Volatility Path**: Each time step updates volatility, allowing it to evolve and revert to the mean.
- **Price Path**: Each price step uses the current volatility, reflecting calm or extreme conditions.

Step 3: Calculate Option Price with Stochastic Volatility

```
def calculate_option_price_with_stochastic_volatility(price_paths, K, T, r):
    payoff_stochastic = np.maximum(price_paths[:, -1] - K, 0)
    return np.exp(-r * T) * np.mean(payoff_stochastic)
```

Explanation: This function calculates the option price by discounting the payoffs at maturity, reflecting the present value of the simulated future outcomes.

Step 4: Main Execution Block

```
S0 = 70         # Initial price
K = 75          # Strike price
T = 1.0         # Time to maturity
r = 0.035       # Risk-free rate
n_simulations = 10000
n_steps = 252

# Initialize Heston model parameters and volatility paths
vol_paths, kappa, theta, sigma_vol = initialize_heston_parameters(n_simulations, n_steps)

# Run simulation for price and volatility paths
price_paths, vol_paths = simulate_stochastic_paths(S0, K, T, r, sigma, n_simulations,
    n_steps, vol_paths, kappa, theta, sigma_vol)

option_price_stochastic = calculate_option_price_with_stochastic_volatility(price_paths, K,
    T, r)
print(f"European Call Option Price (Monte Carlo with Stochastic Volatility): {
    option_price_stochastic:.2f}")
```

Explanation: This block ties together each function, loading parameters, running simulations, and calculating the final option price.

5 Comparison of Results: Standard vs. Stochastic Volatility Monte Carlo

Now that we have implemented both the standard Monte Carlo method with constant volatility and the stochastic volatility Monte Carlo method, we can compare how these two approaches differ in pricing the

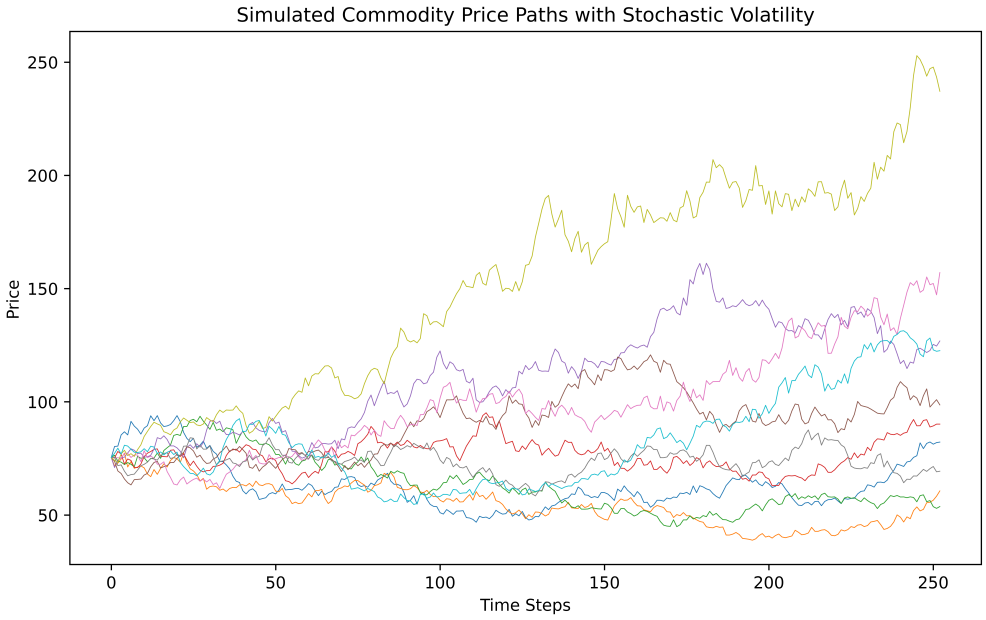
same European call option.

1. **Standard Monte Carlo Simulation (Constant Volatility)**:
   This method assumes that volatility remains fixed throughout the life of the option. While this simplifies the calculations, it is less realistic for commodities like oil, where market conditions are constantly changing.

2. **Stochastic Volatility Monte Carlo Simulation**:
   This method incorporates dynamic volatility that changes over time, reflecting real-world market behavior. It typically leads to different (and often higher) option prices, as the model better captures periods of high risk and sharp price movements.

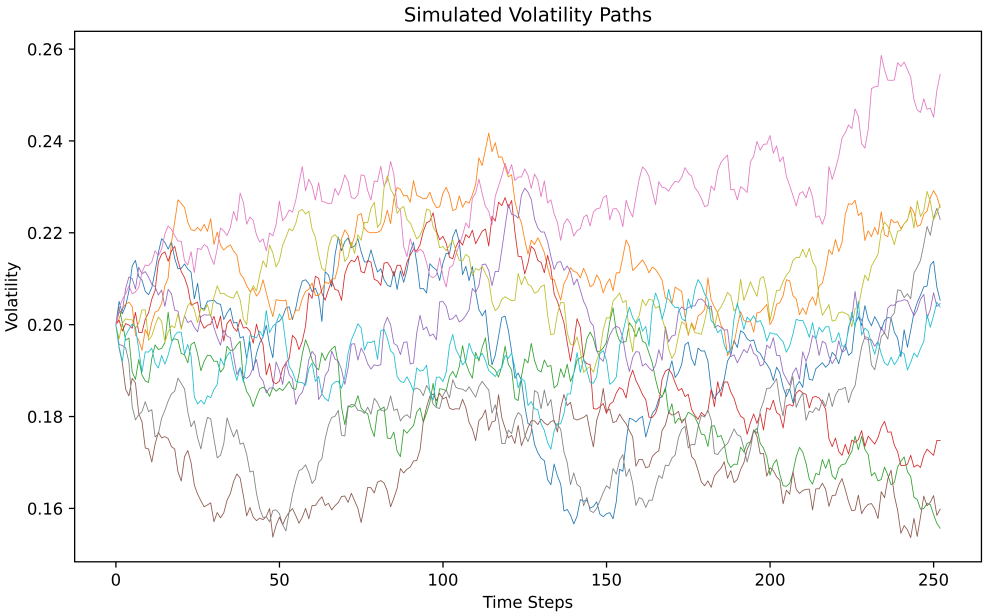
5.1 Simulated Commodity Price Paths with Stochastic Volatility

The plot below shows sample simulated price paths for the commodity (e.g., oil) over time, modeled with stochastic volatility. Each line represents one potential future price path under varying volatility, highlighting the range of possible outcomes.



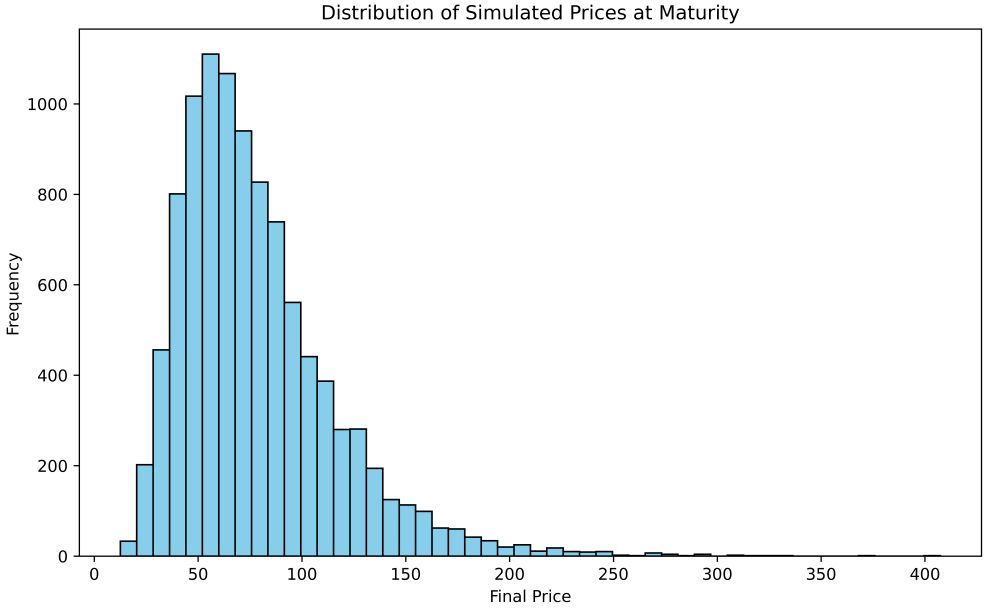
5.2 Simulated Volatility Paths

This plot illustrates sample volatility paths, generated by the Heston model to reflect stochastic behavior in volatility. Volatility fluctuates and tends to revert to a mean, which is typical in commodity markets.



5.3 Distribution of Simulated Prices at Maturity

The histogram shows the distribution of simulated prices at the option's maturity. This reflects the range of possible end-of-period prices under the stochastic volatility model, indicating potential price outcomes and variability.



5.4 Option Pricing Comparison

The output shows the two calculated prices:

- European Call Option Price (Monte Carlo): **12.10**

- European Call Option Price (Monte Carlo with Stochastic Volatility): **14.97**

Typically, the stochastic volatility model will yield a higher price because it incorporates the increased risk from volatility spikes, which the constant-volatility model cannot capture.

6 Real-World Application: Pricing Options in Energy Markets

Monte Carlo simulations with stochastic volatility provide practical advantages in real-world commodity markets, especially for energy trading where prices are highly reactive to external events. Energy firms and traders can utilize these advanced models to gain a more realistic view of potential market risks, helping them price options accurately and make informed hedging decisions.

6.1 How Traders Use Monte Carlo Simulations with Stochastic Volatility

- **Pricing Complex Derivatives:** In addition to standard call and put options, traders frequently price Asian and barrier options in energy markets, which require accurate modeling of volatility to reflect real-world uncertainties.
- **Incorporating Stochastic Volatility for Dynamic Risk Assessment:** Stochastic volatility models allow traders to adjust pricing based on current market conditions. For instance, during geopolitical crises, these models can adapt to higher expected volatility, allowing for more responsive and realistic option prices.

7 Real-World Application: Hedging Risk in Energy Markets

Beyond pricing, Monte Carlo simulations with stochastic volatility are valuable tools for hedging against the unpredictable risks in energy markets. Firms reliant on stable energy prices, such as oil producers, refiners, or logistics providers, can use these models to secure options that safeguard their revenues during volatile periods.

7.1 Strategic Hedging in Volatile Markets

- **Responding to Geopolitical Events:** Geopolitical crises can lead to sudden volatility spikes, affecting both supply and demand dynamics. Stochastic models provide energy firms with more accurate option valuations, helping them to hedge exposure during uncertain times and mitigate potential losses.
- **Managing Supply Disruptions:** Natural disasters or supply chain interruptions frequently impact energy supplies. By pricing options using Monte Carlo simulations that incorporate stochastic volatility, firms can plan for a broader range of price outcomes, supporting more resilient risk management.

8 Future Directions: Advanced Models for Evolving Commodity Markets

While Monte Carlo simulations with stochastic volatility offer a significant advancement over traditional constant-volatility models, the complexity of modern commodity markets requires ongoing model develop-

ment. Below are several emerging models that promise to enhance risk management and pricing accuracy even further.

8.1 Machine Learning and Artificial Intelligence in Option Pricing

Machine learning (ML) models are now being used to further enhance option pricing. By leveraging vast datasets, ML algorithms can uncover complex, non-linear relationships between price movements and external factors, offering real-time adjustments and predictive insights. For example:

- **Reinforcement Learning:** ML models, particularly reinforcement learning, can help optimize dynamic hedging strategies by adapting to sudden price changes in real-time.
- **Neural Networks:** Neural networks are used to capture complex patterns in price and volatility data, outperforming traditional models in markets with heavy tails or asymmetric distributions, which are common in commodities.

8.2 Jump-Diffusion and Variance Gamma Models for Capturing Price Shocks

To capture the sudden, discrete jumps often observed in energy prices due to unexpected events, Jump-Diffusion and Variance Gamma models are increasingly being applied. These models integrate both continuous price changes and discrete jumps, better reflecting the risks and extreme outcomes associated with energy market disruptions.

- **Jump-Diffusion Models:** By incorporating jumps in the price path, these models provide more realistic valuations for scenarios involving large price swings, such as during geopolitical shocks or regulatory changes.
- **Variance Gamma Models:** These models assume returns follow a heavy-tailed process, offering improved accuracy in markets prone to extreme volatility and skewness.

8.3 Rough Volatility Models for Short-Term and High-Frequency Trading

Recent research suggests that "rough" volatility models, such as the Rough Bergomi model, may better capture the irregular volatility paths seen in short-term trading. Unlike traditional models, rough volatility models incorporate abrupt shifts in volatility that are crucial for high-frequency trading and for options with short maturities, common in energy markets.

9 Conclusion: The Future of Commodity Option Pricing

As energy markets continue to grow in complexity, the importance of using adaptive, advanced option pricing techniques becomes clear. Monte Carlo simulations with stochastic volatility have provided a much-needed improvement over traditional models, capturing the high-risk, high-reward nature of commodity markets. However, future models incorporating jump-diffusion processes, machine learning, and rough volatility will likely become essential as markets evolve.

By embracing these advanced pricing methods, firms will be better positioned to manage risk effectively, make informed hedging decisions, and adapt to the unique challenges of global commodity markets. As

new models and computational tools emerge, the potential to refine risk assessments and price options with unprecedented accuracy will reshape risk management in energy trading and beyond.

10 References

Hull, J. C. *Options, Futures, and Other Derivatives* (10th ed.). Pearson.

QuantPy. *Simulating the Heston Model with Python — Stochastic Volatility Modelling.* QuantPy.

QuantPy. *Monte Carlo Simulation for Option Pricing with Python (Basic Ideas Explained).* QuantPy.

Gkinis, S. *Modelling Energy Markets and Pricing Energy Derivatives.* Doctoral dissertation, Cass Business School.

Natenberg, S. *Option Volatility & Pricing: Advanced Trading Strategies and Techniques.* McGraw-Hill.