WTI Crude Oil Vanilla Option Pricing using the Ornstein–Uhlenbeck Model

August 21, 2025

1 The Ornstein–Uhlenbeck (OU) Process

The OU process captures mean reversion:

$$dX_t = \theta(\mu - X_t)\,dt + \sigma\,dW_t, \tag{1}$$

where $\theta > 0$ is the mean–reversion rate, μ the long–run mean, σ the diffusion scale, and W_t a Wiener process. Key properties:

- **Mean reversion:** $\theta(\mu - X_t)$ pulls X_t toward μ.
- **Constant volatility:** Shock size is governed by constant σ.
- **Gaussian marginals:** X_t is normally distributed at fixed t.

Transition distribution. For $T > t$,

$$X_T = \mu + (X_t - \mu)e^{-\theta(T-t)} + \sigma \int_t^T e^{-\theta(T-s)}\,dW_s, \tag{2}$$

$$\mathbb{E}[X_T \mid X_t] = \mu + (X_t - \mu)e^{-\theta\tau}, \tag{3}$$

$$\mathrm{Var}[X_T \mid X_t] = \frac{\sigma^2}{2\theta}\big(1 - e^{-2\theta\tau}\big), \tag{4}$$

with $\tau = T - t$.

2 Why OU for Commodities

Futures prices for storable commodities exhibit forces (storage, convenience yield, production/consumption dynamics) that create pull toward equilibria. Modeling either the futures price F_t or its transform (e.g., log) as OU injects mean reversion absent from geometric Brownian motion.

3 European Options on Mean-Reverting Futures

Assume the *futures price* follows OU under the risk-neutral measure:

$$dF_t = \theta(\mu - F_t)\,dt + \sigma\,dW_t. \tag{5}$$

Then $F_T \mid F_t \sim \mathcal{N}(m,\, v)$ with

$$m = \mu + (F_t - \mu)e^{-\theta\tau}, \qquad v = \frac{\sigma^2}{2\theta}\big(1 - e^{-2\theta\tau}\big). \tag{6}$$

Call/put valuation. For a European option with strike K and maturity T,

$$C = \mathbb{E}\big[(F_T - K)^+\big] = (m - K)\,\Phi(d) + \sqrt{v}\,\phi(d), \tag{7}$$
$$P = \mathbb{E}\big[(K - F_T)^+\big] = (K - m)\,\Phi(-d) + \sqrt{v}\,\phi(d), \tag{8}$$
$$d = \frac{m - K}{\sqrt{v}}, \tag{9}$$

where Φ and ϕ are the standard normal CDF and PDF. These formulas are analogous to Black–Scholes on futures but use the OU transition mean/variance. (If discounting is needed, multiply by $e^{-r\tau}$.)

4 Differences vs. Black–Scholes

- **Mean reversion:** OU embeds pull toward μ; GBM does not.
- **Distributional shape:** OU yields normal F_T (or normal log if modeling $\log F_t$); GBM yields lognormal.
- **Volatility structure:** OU uses level-independent σ; GBM uses proportional (percent) volatility.
- **Estimation:** OU parameters (θ, μ, σ) are typically estimated by time-series regression/MLE on futures data.

5 Parameter Estimation by Regression (Discretized OU)

Let $\Delta X_t = X_{t+\Delta} - X_t$ with step Δ. From the Euler discretization,

$$\frac{\Delta X_t}{\Delta} \approx \theta(\mu - X_t) + \frac{\sigma}{\sqrt{\Delta}}\varepsilon_t,$$

suggesting an OLS of $(\Delta X_t/\Delta)$ on a constant and X_t:

$$\frac{\Delta X_t}{\Delta} = a + bX_t + \varepsilon_t \quad \Rightarrow \quad \theta = -b, \;\; \mu = \frac{a}{\theta}, \;\; \sigma^2 \approx \mathrm{Var}(\varepsilon_t)\,\Delta.$$

6 Python Reference Implementation

```python
import numpy as np
import statsmodels.api as sm
from scipy.stats import norm

def estimate_ou_parameters(time_series, dt):
    """
    Estimate OU parameters (theta, mu, sigma) from a price series.
    dt in years (e.g., 1/252 for daily trading steps).
    """
    x = np.asarray(time_series, dtype=float)
    dX = np.diff(x)
    dX_dt = dX / dt

    X_t = sm.add_constant(x[:-1])         # [const, X_t]
    model = sm.OLS(dX_dt, X_t).fit()
    a, b = model.params                    # dX/dt = a + b X_t + eps

    theta = -b
    mu = a / theta if theta != 0 else np.mean(x)
    sigma = np.sqrt(model.resid.var() * dt)
    return float(theta), float(mu), float(sigma)

def ou_option_price_on_futures(F_t, K, tau, theta, mu, sigma, option_type=
    "call"):
    """
    Price a European option on a futures assumed to follow OU under Q.
    Returns the undiscounted expectation E[(F_T-K)^+] or E[(K-F_T)^+].
    """
    if tau <= 0:
        intrinsic = max(F_t - K, 0.0) if option_type == "call" else max(K
    - F_t, 0.0)
        return float(intrinsic)

    m = mu + (F_t - mu) * np.exp(-theta * tau)
    if theta > 1e-12:
        v = (sigma**2 / (2.0 * theta)) * (1.0 - np.exp(-2.0 * theta * tau)
    )
    else:
        v = (sigma**2) * tau

    v = max(v, 0.0)
    s = np.sqrt(v) if v > 0 else 0.0
    d = np.inf if (s == 0 and m > K) else (-np.inf if (s == 0 and m <= K)
    else (m - K) / s)

    if option_type.lower() == "call":
        return (m - K) * norm.cdf(d) + s * norm.pdf(d)
    elif option_type.lower() == "put":
        return (K - m) * norm.cdf(-d) + s * norm.pdf(d)
    else:
```

```
        raise ValueError("option_type must be 'call' or 'put'")
```

Listing 1: OU parameter estimation and option pricing on futures

7 Illustrative Workflow

1. Estimate (θ, μ, σ) from a historical WTI futures series with `estimate_ou_parameters`.
2. For a given date, strike K, and maturity τ (in years), compute call/put via `ou_option_price_on_futur`
3. Repeat across dates/strikes to populate OTM/ATM/ITM columns for analysis.

8 Notes and Caveats

- If modeling $\log F_t$ as OU, transform inputs/outputs accordingly; option formulas then use log-OU mean/variance.
- Discounting by $e^{-r\tau}$ can be applied if pricing off spot rather than futures or if required by the use case.
- Real markets may show time-varying volatility, jumps, seasonality, or term-structure effects; extensions (e.g., OU with stochastic σ, multi-factor models) can be layered as needed.