Calibrating the Heston Model

Lorenzo Naranjo

March 2026

Introduction

The Heston Model notebook derives the closed-form option pricing formula from the stochastic volatility dynamics and the characteristic function of the log stock price. This notebook focuses entirely on implementation and calibration: we translate that formula into Python, and then find the model parameters that best match observed call prices on Apple (AAPL) as of October 1, 2024.

Calibration means solving an inverse problem. Given a set of observed option prices $C^{\text{obs}}(K_i, T_j)$ across strikes $K_1, \dots, K_m$ and maturities $T_1, \dots, T_n$, we set v_0 from the short-dated ATM implied volatility and search for the parameter vector $(\kappa, \theta, \sigma, \rho)$ that minimizes the root mean-squared pricing error

$$\text{RMSE} = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(C^{\text{obs}}(K_i, T_j) - C^{\text{model}}(K_i, T_j)\right)^2}.$$

Once calibrated, the model can be used to price exotic derivatives, compute sensitivities, or simulate risk scenarios with a volatility structure that is consistent with the market.

This notebook presents the simplest possible calibration exercise: we fit all parameters to a single cross-section of option prices on one date. A more rigorous approach would split the data into an estimation sample and a hold-out sample, calibrate $(\kappa, \theta, \sigma, \rho)$ on the training set, and then backtest the model by repricing the hold-out contracts with those fixed parameters. Using the short-dated ATM implied volatility as a proxy for $\sqrt{v_0}$ is a natural choice in any such exercise, because v_0 is the only parameter that should be updated daily as market conditions change, while the structural parameters $(\kappa, \theta, \sigma, \rho)$ are more stable and can be estimated over a longer horizon.

One-factor stochastic volatility models like Heston face inherent limitations in fitting the full surface simultaneously across strikes and maturities. Multifactor extensions that allow the variance process to have richer dynamics can substantially improve the fit, as shown in Cortazar et al. (2017) in the context of commodity markets.

Implementing the Pricing Formula

The Heston call price is

$$C = Se^{-qT}P_1 - Ke^{-rT}P_2,$$

where P_2 is the risk-neutral probability of expiring in the money and P_1 is the corresponding probability under the stock-numeraire measure. Both are recovered by Fourier inversion of the joint characteristic function of $x(T) = \ln S(T)$ and $v(T)$.

We begin by loading the required libraries.

```
import numpy as np
from numpy import log, exp, pi, real
from scipy.integrate import quad
from scipy.stats import norm
from scipy.optimize import brentq, minimize
import matplotlib.pyplot as plt
import pandas as pd
```

The characteristic function

The joint characteristic function of $x(T)$ and $v(T)$ is

$$f(\phi,\varphi) = \exp\left(i(r-q)\phi T + \left(\delta - \frac{2\gamma}{\sigma^2}\right)\kappa\theta^* T + i\phi x + \delta v \right.$$
$$-\frac{2\kappa\theta^*}{\sigma^2}\ln\left(e^{-\gamma T} - \frac{\sigma^2}{2\gamma}(1-e^{-\gamma T})(i\varphi-\delta)\right)$$
$$\left. + \frac{v(i\varphi-\delta)}{e^{-\gamma T} - \frac{\sigma^2}{2\gamma}(1-e^{-\gamma T})(i\varphi-\delta)}\right),$$

where

$$\gamma = \sqrt{\kappa^2 + (1-\rho^2)\sigma^2\phi^2 + i(\sigma - 2\kappa\rho)\sigma\phi},$$
$$\delta = \frac{\kappa + \gamma - i\rho\sigma\phi}{\sigma^2}.$$

The implementation puts every factor inside the exponential to avoid numerical instabilities from branch cuts in the complex square root.

```
def CF_Heston(phi1, phi2, S0, V0, T, r, q, kappa, theta, sigma, rho):
    gamma = (kappa**2 + (1 - rho**2) * sigma**2 * phi1**2
             + 1j * (sigma - 2 * kappa * rho) * sigma * phi1)**0.5
    delta = (kappa + gamma - 1j * rho * sigma * phi1) / sigma**2

    y = exp(
        1j * (r - q) * phi1 * T
        + (delta - 2 * gamma / sigma**2) * kappa * theta * T
        + 1j * phi1 * log(S0)
        + delta * V0
        + (-2 * kappa * theta / sigma**2)
        * np.log(exp(-gamma * T) - sigma**2 / (2 * gamma)
                 * (1 - exp(-gamma * T)) * (1j * phi2 - delta))
        + V0 * (1j * phi2 - delta)
        / (exp(-gamma * T) - sigma**2 / (2 * gamma)
           * (1 - exp(-gamma * T)) * (1j * phi2 - delta))
```

```
    )
    return y
```

In-the-money probabilities

The risk-neutral probability P_2 that the call expires in the money is recovered by Fourier inversion of $f(\phi, 0)$.

```
def P2(S0, V0, K, T, r, q, kappa, theta, sigma, rho):
    integrand = lambda phi1: real(
        exp(-1j * phi1 * log(K))
        * CF_Heston(phi1, 0, S0, V0, T, r, q, kappa, theta, sigma, rho)
        / (1j * phi1)
    )
    y = 0.5 + 1 / pi * quad(integrand, 0, np.inf, full_output=1)[0]
    return y
```

The probability P_1 under the stock-numeraire measure is obtained by evaluating the characteristic function at $\phi - i$ instead of ϕ, then dividing by the forward price $F = Se^{(r-q)T}$.

```
def P1(S0, V0, K, T, r, q, kappa, theta, sigma, rho):
    F = S0 * exp((r - q) * T)
    integrand = lambda phi1: real(
        exp(-1j * phi1 * log(K))
        * CF_Heston(phi1 - 1j, 0, S0, V0, T, r, q, kappa, theta, sigma, rho)
        / (1j * phi1)
    )
    y = 0.5 + 1 / pi / F * quad(integrand, 0, np.inf, full_output=1)[0]
    return y
```

Call price

The European call price follows directly from the two probabilities.

```
def heston_call_price(S0, V0, K, T, r, q, kappa, theta, sigma, rho):
    call = (S0 * exp(-q * T) * P1(S0, V0, K, T, r, q, kappa, theta, sigma, rho)
            - K * exp(-r * T) * P2(S0, V0, K, T, r, q, kappa, theta, sigma, rho))
    return call
```

Black-Scholes implied volatility

We also need the Black-Scholes call price and its numerical inversion, both for computing the initial variance v_0 before calibration and for plotting the implied volatility surface afterward.

```
def bs_call(S, K, T, r, q, sigma):
    d1 = (log(S / K) + (r - q + 0.5 * sigma**2) * T) / (sigma * T**0.5)
    d2 = d1 - sigma * T**0.5
    return S * exp(-q * T) * norm.cdf(d1) - K * exp(-r * T) * norm.cdf(d2)

def bs_implied_vol(S, K, T, r, q, market_price):
    try:
        return brentq(lambda sig: bs_call(S, K, T, r, q, sig) - market_price,
                      1e-6, 5.0)
    except ValueError:
        return np.nan
```

Verification

We verify the implementation with a standard set of parameter values before moving to real data.

```python
S0_test  = 100
K_test   = 100
T_test   = 1.0
r_test   = 0.05
q_test   = 0.02
kappa_t  = 2.0
theta_t  = 0.04
sigma_t  = 0.5
rho_t    = -0.7
V0_test  = 0.04

call_test = heston_call_price(S0_test, V0_test, K_test, T_test,
                              r_test, q_test, kappa_t, theta_t, sigma_t, rho_t)
print(f"Test call price: {call_test:.4f}")
```

```
Test call price: 8.7526
```

AAPL Option Data

The option prices used here come from the OptionMetrics (OPTIONM) database. OPTIONM records the highest closing bid and lowest closing ask across all exchanges for every listed contract. We use the midpoint $\frac{\text{bid}+\text{ask}}{2}$ as the market price for each contract. The midpoint is the standard choice in empirical options research because it is free from the direction of trading and minimizes the impact of the bid-ask bounce on estimated parameters.

AAPL options are American-style. For call options with a small dividend yield the early-exercise premium is negligible, so we treat the observed prices as European call prices without adjustment. On October 1, 2024, AAPL closed at $226.21. We use a dividend yield of 0.5% per year and a continuously compounded risk-free rate of 5%.

```
S0 = 226.21
r  = 0.05
q  = 0.005
```

We select seven strikes spanning roughly $±$15% around the current stock price and six standard expiries ranging from 45 days to 198 days, giving 42 contracts in total. Time-to-maturity is measured in years from October 1, 2024.

```
strikes = [200, 210, 220, 230, 240, 250, 260]

# Expiration dates and exact days to expiry from 2024-10-01
# Nov 15 2024:  45 days; Dec 20 2024:  80 days; Jan 17 2025: 108 days;
# Feb 21 2025: 143 days; Mar 21 2025: 171 days; Apr 17 2025: 198 days
maturities = [45/365, 80/365, 108/365, 143/365, 171/365, 198/365]
maturity_labels = ['Nov-24', 'Dec-24', 'Jan-25', 'Feb-25', 'Mar-25', 'Apr-25']
```

The midpoint prices below are taken directly from OPTIONM for the contracts described above.

```
# Midpoint prices: rows = strikes (200 to 260), columns = maturities (Nov-24 to Apr-25
market_prices = np.array([
    [28.950, 30.775, 32.125, 33.925, 35.325, 36.475],  # K = 200
    [20.450, 22.625, 24.175, 26.300, 27.725, 29.025],  # K = 210
    [13.050, 15.475, 17.075, 19.475, 21.000, 22.400],  # K = 220
    [ 7.275,  9.625, 11.250, 13.725, 15.225, 16.625],  # K = 230
    [ 3.400,  5.350,  6.800,  9.075, 10.525, 11.850],  # K = 240
    [ 1.395,  2.680,  3.750,  5.700,  6.975,  8.175],  # K = 250
    [ 0.540,  1.260,  1.970,  3.425,  4.450,  5.450],  # K = 260
])
```

We display the data as a table before calibrating.

```
df_market = pd.DataFrame(
    market_prices,
    index=pd.Index(strikes, name='Strike'),
    columns=maturity_labels,
)
df_market
```

Table 1: AAPL call option midpoint prices from OPTIONM on October 1, 2024. Rows are strikes; columns are expiration months.

Strike	Nov-24	Dec-24	Jan-25	Feb-25	Mar-25	Apr-25
200	28.950	30.775	32.125	33.925	35.325	36.475
210	20.450	22.625	24.175	26.300	27.725	29.025
220	13.050	15.475	17.075	19.475	21.000	22.400
230	7.275	9.625	11.250	13.725	15.225	16.625
240	3.400	5.350	6.800	9.075	10.525	11.850
250	1.395	2.680	3.750	5.700	6.975	8.175
260	0.540	1.260	1.970	3.425	4.450	5.450

Calibration

We calibrate four Heston parameters $(\kappa, \theta, \sigma, \rho)$ by minimizing the RMSE between observed and model call prices, holding v_0 fixed at a value extracted directly from market data.

Fixing the initial variance

Treating v_0 as a free parameter creates a degeneracy: a very large κ paired with a v_0 far from the short-term implied variance can produce nearly the same option prices as a moderate κ with a well-anchored v_0, because the two parameters compensate each other. Fixing v_0 breaks

this degeneracy, reduces the problem from five to four free parameters, and anchors the short end of the volatility surface to a directly observable market quantity.

We set v_0 equal to the squared Black-Scholes implied volatility of the shortest-maturity contract whose strike is nearest to the forward price $F = Se^{(r-q)T}$.

```
T_short = maturities[0]
F_short = S0 * exp((r - q) * T_short)
atm_idx = int(np.argmin(np.abs(np.array(strikes) - F_short)))
iv_atm  = bs_implied_vol(S0, strikes[atm_idx], T_short, r, q, market_prices[atm_idx, 0
V0_fixed = iv_atm**2

print(f"ATM strike (Nov-24): {strikes[atm_idx]}")
print(f"ATM implied vol:     {iv_atm:.4f}  ({iv_atm*100:.2f}%)")
print(f"V0 fixed to:         {V0_fixed:.6f}")
```

```
ATM strike (Nov-24): 230
ATM implied vol:     0.2662  (26.62%)
V0 fixed to:         0.070862
```

Running the optimizer

The objective function loops over all 42 contracts using the fixed v_0.

```
def rmse_heston(params):
    kappa, theta, sigma, rho = params
    total = 0.0
    for i, K in enumerate(strikes):
        for j, T in enumerate(maturities):
            model = heston_call_price(S0, V0_fixed, K, T, r, q, kappa, theta, sigma, r
            total += (market_prices[i, j] - model) ** 2
    return np.sqrt(total / (len(strikes) * len(maturities)))
```

```python
bounds = [
    (0.001, 20.0),   # kappa
    (0.001,  1.0),   # theta
    (0.001,  5.0),   # sigma
    (-1.0,   1.0),   # rho
]

x0 = [2.0, V0_fixed, 0.5, -0.7]

result = minimize(rmse_heston, x0, method='L-BFGS-B', bounds=bounds)

kappa_cal, theta_cal, sigma_cal, rho_cal = result.x

print(f"kappa = {kappa_cal:.4f}")
print(f"theta = {theta_cal:.4f}  (sqrt: {theta_cal**0.5:.4f})")
print(f"sigma = {sigma_cal:.4f}")
print(f"rho   = {rho_cal:.4f}")
print(f"V0    = {V0_fixed:.4f}  (sqrt: {V0_fixed**0.5:.4f})  [fixed]")
print(f"RMSE  = {result.fun:.4f}")
```

```
kappa = 3.4879
theta = 0.0627  (sqrt: 0.2504)
sigma = 0.8174
rho   = -0.4823
V0    = 0.0709  (sqrt: 0.2662)  [fixed]
RMSE  = 0.2070
```

Interpreting the calibrated parameters

The four calibrated parameters have natural economic interpretations.

The speed of mean reversion κ controls how quickly the instantaneous variance returns to its long-run level θ. Larger values imply faster reversion; as $\kappa \to \infty$ the variance process becomes almost deterministic and the model collapses toward a constant-volatility world.

The long-run variance θ determines the volatility level that options with long maturities are priced against. Its square root $\sqrt{\theta}$ is directly comparable to the implied volatility of long-dated at-the-money options.

The volatility of volatility σ governs the curvature of the implied volatility surface. A larger σ produces a more pronounced smile because the variance process itself is more uncertain.

The correlation ρ captures the leverage effect: negative values make price drops and variance increases co-move, which tilts the implied volatility surface into the downward-sloping skew that is characteristic of equity markets. The sign should be negative for AAPL.

Implied Volatility Analysis

We compute the Black-Scholes implied volatility for every contract in the data set.

```
iv = np.zeros_like(market_prices)
for i, K in enumerate(strikes):
    for j, T in enumerate(maturities):
        iv[i, j] = bs_implied_vol(S0, K, T, r, q, market_prices[i, j])
```

The figure below shows the implied volatility surface. The axes are strike and days to expiry; the height and color show the level of implied volatility.

```
from mpl_toolkits.mplot3d import Axes3D

K_grid = np.array(strikes)
T_grid = np.array(maturities) * 365   # days to expiry for the axis label

KK, TT = np.meshgrid(K_grid, T_grid)
```

```
fig = plt.figure(figsize=(8, 5))
ax  = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(KK, TT, iv.T * 100, cmap='inferno', alpha=0.9,
                       linewidth=0.3, edgecolor='0.4')
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10, pad=0.1, label='IV (%)')
ax.set_xlabel('Strike')
ax.set_ylabel('Days to Expiry')
ax.set_zlabel('Implied Vol (%)')
ax.set_title('AAPL Implied Volatility Surface - October 1, 2024')
plt.tight_layout()
plt.show()
```

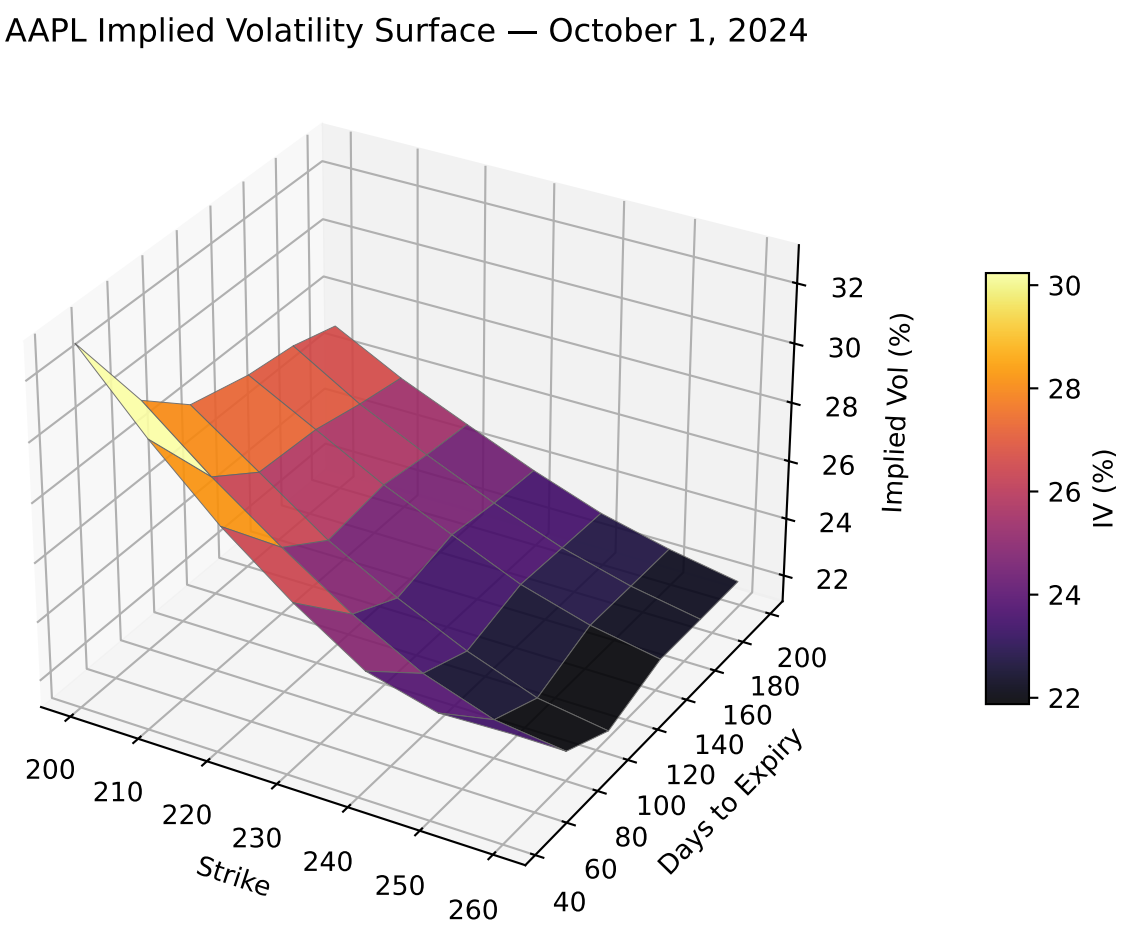

Figure 1: Black-Scholes implied volatility surface for AAPL options from OPTIONM on October 1, 2024.

The surface shows the typical equity volatility skew: implied volatilities are highest for low strikes and decrease monotonically as the strike rises. The skew is steeper for short maturities and flattens as maturity increases, which is exactly the pattern produced by the Heston model with a negative ρ.

By construction, $\sqrt{v_0}$ equals the implied volatility of the nearest-ATM Nov-24 option, so the short-dated ATM level is matched exactly. Long-dated options are priced closer to $\sqrt{\theta}$, reflecting the mean reversion of v toward its long-run level.

Pricing Errors

We compute model prices and pricing errors for all 42 contracts.

```
model_prices = np.zeros_like(market_prices)
for i, K in enumerate(strikes):
    for j, T in enumerate(maturities):
        model_prices[i, j] = heston_call_price(
            S0, V0_fixed, K, T, r, q, kappa_cal, theta_cal, sigma_cal, rho_cal
        )

abs_errors = np.abs(market_prices - model_prices)
```

```
rows = []
for i, K in enumerate(strikes):
    for j, (label, T) in enumerate(zip(maturity_labels, maturities)):
        rows.append({
            'Strike': K,
            'Expiry': label,
            'Observed': f"{market_prices[i, j]:.3f}",
            'Model': f"{model_prices[i, j]:.3f}",
            'Abs. Error': f"{abs_errors[i, j]:.3f}",
        })

df_errors = pd.DataFrame(rows).set_index(['Strike', 'Expiry'])
df_errors
```

Table 2: Observed prices, Heston model prices, and absolute pricing errors for AAPL calls on October 1, 2024.

Strike	Expiry	Observed	Model	Abs. Error
200	Nov-24	28.950	28.488	0.462
	Dec-24	30.775	30.630	0.145
	Jan-25	32.125	32.191	0.066
	Feb-25	33.925	33.984	0.059
	Mar-25	35.325	35.325	0.000
	Apr-25	36.475	36.557	0.082
210	Nov-24	20.450	19.908	0.542
	Dec-24	22.625	22.515	0.110
	Jan-25	24.175	24.300	0.125
	Feb-25	26.300	26.302	0.002
	Mar-25	27.725	27.780	0.055
	Apr-25	29.025	29.128	0.103
220	Nov-24	13.050	12.474	0.576
	Dec-24	15.475	15.383	0.092
	Jan-25	17.075	17.310	0.235
	Feb-25	19.475	19.448	0.027
	Mar-25	21.000	21.017	0.017
	Apr-25	22.400	22.443	0.043
230	Nov-24	7.275	6.726	0.549
	Dec-24	9.625	9.579	0.046
	Jan-25	11.250	11.490	0.240
	Feb-25	13.725	13.628	0.097
	Mar-25	15.225	15.207	0.018
	Apr-25	16.625	16.648	0.023
240	Nov-24	3.400	3.022	0.378
	Dec-24	5.350	5.368	0.018
	Jan-25	6.800	7.052	0.252
	Feb-25	9.075	9.009	0.066

Table 2: Observed prices, Heston model prices, and absolute pricing errors for AAPL calls on October 1, 2024.

Strike	Expiry	Observed	Model	Abs. Error
250	Mar-25	10.525	10.488	0.037
	Apr-25	11.850	11.859	0.009
	Nov-24	1.395	1.146	0.249
	Dec-24	2.680	2.731	0.051
	Jan-25	3.750	4.024	0.274
	Feb-25	5.700	5.633	0.067
	Mar-25	6.975	6.906	0.069
	Apr-25	8.175	8.119	0.056
260	Nov-24	0.540	0.387	0.153
	Dec-24	1.260	1.301	0.041
	Jan-25	1.970	2.179	0.209
	Feb-25	3.425	3.372	0.053
	Mar-25	4.450	4.377	0.073
	Apr-25	5.450	5.372	0.078

We summarize the mean absolute error for each strike-expiry combination, with marginal averages by row and column.

```
df_mae = pd.DataFrame(
    abs_errors.round(4),
    index=pd.Index(strikes, name='Strike'),
    columns=maturity_labels,
)
df_mae['Avg'] = abs_errors.mean(axis=1).round(4)
df_mae.loc['Avg'] = abs_errors.mean(axis=0).tolist() + [abs_errors.mean().round(4)]
df_mae
```

Table 3: Mean absolute pricing error by strike and expiry. The Avg row and column contain the marginal means.

Strike	Nov-24	Dec-24	Jan-25	Feb-25	Mar-25	Apr-25	Avg
200	0.462100	0.144600	0.065800	0.059200	0.000200	0.081700	0.1356
210	0.541500	0.110100	0.124700	0.002400	0.054900	0.102700	0.1560
220	0.575800	0.091700	0.234900	0.026900	0.016500	0.042600	0.1647
230	0.549100	0.045900	0.239500	0.096700	0.018000	0.022800	0.1620
240	0.377700	0.017700	0.252000	0.066500	0.036900	0.008500	0.1265
250	0.248900	0.051000	0.274300	0.067100	0.068900	0.056000	0.1277
260	0.152600	0.041400	0.209100	0.052800	0.073400	0.077500	0.1011
Avg	0.415384	0.071775	0.200037	0.053076	0.038411	0.055978	0.1391

The figure below plots the pricing errors by strike for each expiry.

```
pricing_errors = model_prices - market_prices

fig, axes = plt.subplots(2, 3, figsize=(9, 5), sharey=True)
for j, (label, ax) in enumerate(zip(maturity_labels, axes.flat)):
    ax.bar(strikes, pricing_errors[:, j], color=['#d62728' if e > 0 else '#1f77b4'
                                                 for e in pricing_errors[:, j]])
    ax.axhline(0, color='0.3', lw=0.8, ls='--')
    ax.set_title(label, fontsize=10)
    ax.set_xticks(strikes)
    ax.set_xticklabels(strikes, fontsize=7)
    ax.spines[['top', 'right']].set_visible(False)

fig.supxlabel('Strike', fontsize=9)
fig.supylabel('Error ($)', fontsize=9)
plt.tight_layout()
plt.show()
```

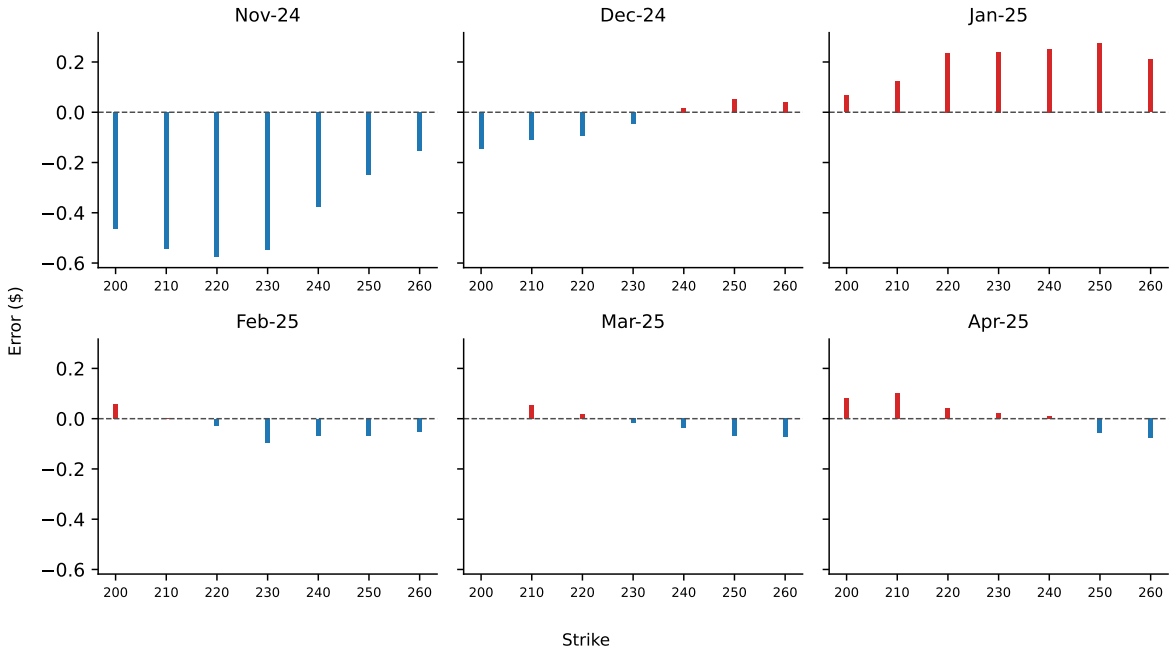

Figure 2: Heston model pricing errors (model minus market) for AAPL calls on October 1, 2024. Each panel corresponds to one expiry.

The largest errors occur in short-maturity contracts. This is a structural limitation of the Heston model: the model-implied skew is proportional to $\sqrt{T}$ as maturity shrinks, so it flattens too quickly near expiry and cannot match the steep short-dated skew observed in the market. The RMSE treats all contracts equally, but near-the-money options have larger dollar prices, so their squared errors receive more implicit weight and the optimizer tilts the fit toward those contracts across all maturities.

The overall fit demonstrates that the Heston model can reproduce the main features of the equity implied volatility surface—the level, the skew, and the term structure of volatility—with just four free parameters.

References

Cortazar, Gonzalo, Matias Lopez, and Lorenzo Naranjo. 2017. “A Multifactor Stochastic Volatility Model of Commodity Prices.” *Energy Economics* 67: 182–201.