Case 7: Option Pricing with the Heston Model

Asha de Meij - i6254733

December 12, 2024

Question 1. Implement a Monte-Carlo simulation to compute the price of call-options with $T = 10$ (i.e., option maturity 10 years) and strikes 50, 60, 70, ..., 180, 190, 200. Use the Black-Scholes formula to convert the Heston option prices to implied volatilities, and plot the implied volatilities in a graph (like in Slide 6).

This Monte Carlo simulation differs from the approaches used in previous weeks because it involves simulating paths for both the variance (V) and the stock price (S) for the Heston model. This allows us to incorporate stochastic volatility into the option pricing model, providing more realistic price estimates. The implemented code is shown in Figure 1.

```
def mc_heston(V0, S0, kappa, theta, T, sigma_v, N, M, rho, r):

    dt = T / N

    Z_S = np.random.normal(size=(N, M))
    Z_V = np.random.normal(size=(N, M))

    W_S = Z_S
    W_V = rho * Z_S + np.sqrt(1 - rho**2) * Z_V

    S = np.full(M, np.log(S0))
    V = np.full(M, V0)

    for t in range(N):
        sqrt_v = np.sqrt(np.maximum(V, 0))
        S += (r - 0.5 * V) * dt + sqrt_v * np.sqrt(dt) * W_S[t]
        V += kappa * (theta - V) * dt + sigma_v * sqrt_v * np.sqrt(dt) * W_V[t]
        V = np.maximum(V, 0)

    S_T_heston = np.exp(S)

    S_T_heston = mc_heston(V0, S0, kappa, theta, T, sigma_v, time_steps, numSim, rho, r)

    call_prices = []
    SE = []
    discount_factor = np.exp(-r*T)

    for K in strikes:
        payoffs_heston = np.maximum(S_T_heston - K, 0)
        price_heston = np.exp(-r * T) * np.mean(payoffs_heston)
        payoff_std = np.std(payoffs_heston)
        se = discount_factor * payoff_std / np.sqrt(numSim)

        call_prices.append(price_heston)
        SE.append(se)


    return call_prices, SE
```

Figure 1: Heston Monte Carlo simulation

The results of the Monte Carlo simulation with M=1000000 and N=100 are shown in the table below. The results show that the call prices decrease as the strike price increases. Furthermore, the error bounds are relatively small.

Strike (K)	Monte Carlo Heston Call Price	Standard Error (SE)
50	59.058624034014656	0.03160239767820745
60	50.99648375645509	0.0313868696852583
70	43.15487232945375	0.03087075870482115
80	35.71388657104994	0.02991358446511778
90	28.86653447971453	0.028446832472632924
100	22.779734998253307	0.02649273349934063
110	17.5596331599988	0.02415355353201692
120	13.2371750235883	0.02157734406033555
130	9.770686600088235	0.01892317290180875
140	7.076595142500846	0.016325354247522253
150	5.036555853408148	0.013887563498385457
160	3.5287091995157565	0.011674782939316366
170	2.438862083829533	0.009718252955977143
180	1.6658794001833364	0.008023828581459188
190	1.1252150939437102	0.006581047793712232
200	0.7532277287787588	0.005368603346793097

Table 1: Monte Carlo Heston Call Prices for Different Strikes (K)

Furthermore, the Black-Scholes formula was used to convert the Heston option prices into implied volatilities. The code used to calculate the implied volatilities can be found in Figure 2.

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

def impliedVolatility(price, S, K, T, r):
    def objFunc(sigma):
        return BlackScholes_call(S, K, T, r, sigma) - price
    try:
        # brentq root-finding (numerical method) used to solve for sigma
        return brentq(objFunc, 1e-6, 5)
    except ValueError:
        return np.nan

impliedVolatilities = []
for K, price in zip(strikes, heston_prices):
    impVol = impliedVolatility(price, S0, K, T, r)
    impliedVolatilities.append(impVol)
```

Figure 2: Implied volatilities computation

Then I proceeded to plot the computed implied volatilities that results in a volatility 'smile' which is shown in Figure 3.

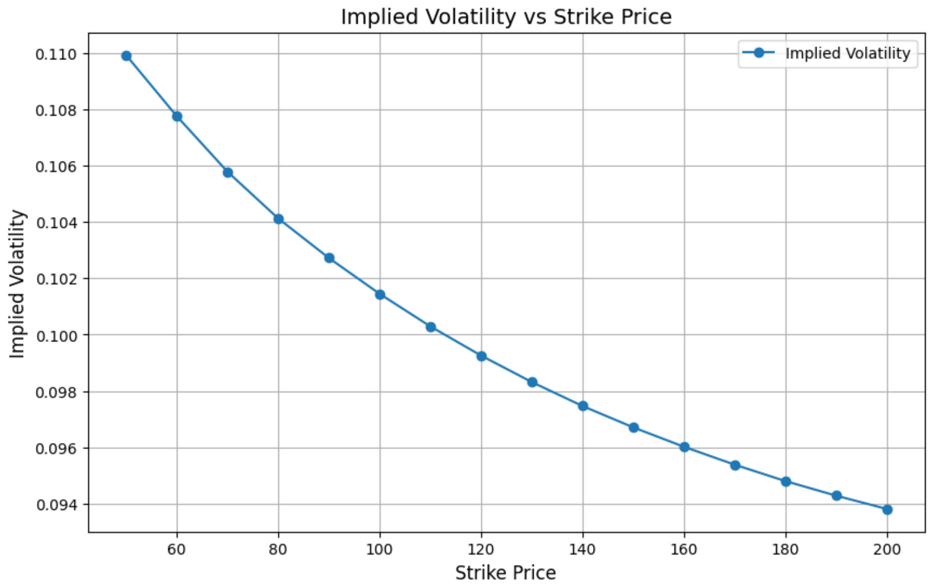

Figure 3: Volatility Smile

Question 2. Calculate the Fourier transform of the "tilted" call-option payoff $e^{-rT}e^{-\alpha x}\max\{e^x - K, 0\}$. Use the tilting parameter α to ensure that the Fourier transform converges for $x \to \infty$. Investigate how the optimal choice for α changes for different values of K between 50 and 200.

The Fourier transform of the tilted call-option payoff is expressed as follows:

$$f(x) = e^{-rT}e^{-\alpha x}\max(e^x - K, 0).$$

Which we can also express as:

$$f(x) = \begin{cases} e^{-rT}\left(e^{(1-\alpha)x} - Ke^{-\alpha x}\right) & \text{if } x > \ln K, \\ 0 & \text{otherwise} \end{cases}$$

The Fourier transform of $f(x)$ is defined as:

$$\mathcal{F}\{f\}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-i\xi x}\,dx = \int_{\ln K}^{\infty} e^{-rT}\left(e^{(1-\alpha)x} - Ke^{-\alpha x}\right)e^{-i\xi x}\,dx$$

$$= e^{-rT}\left(\int_{\ln K}^{\infty} e^{(1-\alpha-i\xi)x}\,dx - K\int_{\ln K}^{\infty} e^{(-\alpha-i\xi)x}\,dx\right).$$

We can start by evaluating the first term and get:

$$\int_{\ln K}^{\infty} e^{(1-\alpha-i\xi)x}\,dx = \frac{e^{(1-\alpha-i\xi)\ln K}}{1-\alpha-i\xi}.$$

By setting $e^{(1-\alpha-i\xi)\ln K} = K^{1-\alpha-i\xi}$ for simplification purposes, we get:

$$\int_{\ln K}^{\infty} e^{(1-\alpha-i\xi)x}\,dx = \frac{K^{1-\alpha-i\xi}}{1-\alpha-i\xi}.$$

Now we can move on to the second term:

$$\int_{\ln K}^{\infty} e^{(-\alpha-i\xi)x}\,dx = \frac{e^{(-\alpha-i\xi)\ln K}}{-\alpha-i\xi}.$$

Again by setting $e^{(-\alpha-i\xi)\ln K} = K^{-\alpha-i\xi}$ we get:

$$\int_{\ln K}^{\infty} e^{(-\alpha-i\xi)x}\,dx = \frac{K^{-\alpha-i\xi}}{-\alpha-i\xi}.$$

Substituting these results back into the Fourier transform, we find:

$$\begin{aligned}\mathcal{F}\{f\}(\xi) &= e^{-rT}\left(\frac{K^{1-\alpha-i\xi}}{1-\alpha-i\xi} - K\cdot\frac{K^{-\alpha-i\xi}}{-\alpha-i\xi}\right)\\ &= e^{-rT}\left(\frac{e^{(1-\alpha+i\xi)\ln K}}{1-\alpha+i\xi} + \frac{K\cdot e^{(-\alpha+i\xi)\ln K}}{-\alpha+i\xi}\right)\end{aligned}$$

The Fourier transform of the "tilted" call-option payoff aswell as redefined functions from Week 4 have been implemented in python shown in the image below.

```python
def heston_charfct(xi, V_t, x_t, T, t, r, kappa, theta, sigma_v, rho):
    d = np.sqrt((rho * sigma_v * 1j * xi - kappa)**2 + sigma_v**2 * (1j * xi + xi**2))
    C = (
        ((1j * xi + xi**2) * (1 - np.exp(d * (T - t)))) /
        ((rho * sigma_v * 1j * xi - kappa) * (1 - np.exp(d * (T - t))) + d * (1 + np.exp(d * (T - t))))
    )

    A = (
        1j * xi * r * (T - t)
        + ((kappa * theta) / (sigma_v**2)) * (
            (d - (1j * rho * sigma_v * xi - kappa)) * (T - t)
            - 2 * np.log(
                ((1 - np.exp(d * (T - t))) * (1j * rho * xi * sigma_v - kappa) + d * (1 + np.exp(d * (T - t)))
            )
        )
    )

    B =  1j * xi * x_t

    return np.exp(A + C * V_t + B)

def FT_tilted_payoff(alpha, xi, K, T, r):
    term1 = -np.exp((1 - alpha + 1j * xi) * np.log(K)) / (1 - alpha + 1j * xi)
    term2 = K * np.exp((-alpha + 1j * xi) * np.log(K)) / (-alpha + 1j * xi)
    return np.exp(-r * T) * (term1 + term2)

def integrate_function(xi, alpha, V_t, x_t, T, t, K, r, kappa, theta, sigma_v, rho):
    cf = heston_charfct(-xi - 1j * alpha, V_t, x_t, T, t, r, kappa, theta, sigma_v, rho)
    payoff = FT_tilted_payoff(alpha, xi, K, T, r)
    return np.real(cf * payoff / (2 * np.pi))
```

Figure 4: Adapted functions for Fourier transform

After redefining the necessary functions, the optimal α values for different strike prices (between 50-200) were determined, as illustrated in figure 5.

```python
strike_prices = np.arange(50, 201, 10)

optimal_alphas = []

for K in strike_prices:
    alpha_values = np.linspace(1.01, 15.0, 1000)
    xi = 0

    real = [integrate_function(xi, a, V_t, x_t, T, t, K, r, kappa, theta, sigma_v, rho) for a in alpha_values]

    # Find the optimal alpha
    optimal_alpha = alpha_values[np.argmin(real)]
    optimal_alphas.append(optimal_alpha)
    print(f"K={K}, Optimal Alpha={optimal_alpha:.5f}")
```

Figure 5: Optimal Alpha computation

The computed optimal α values for different strike prices (K) can be seen in the table below. These values illustrate the relationship between the strike price and the optimal tilting parameter.

Strike Price (K)	**Optimal Alpha**
50	2.4804204204204208
60	2.7885085085085084
70	3.110600600600601
80	3.4747047047047044
90	3.866816816816817
100	4.300940940940941
110	4.763073073073073
120	5.239209209209209
130	5.743353353353354
140	6.261501501501502
150	6.793653653653654
160	7.339809809809809
170	7.8859659659659656
180	8.432122122122122
190	8.978278278278278
200	9.524434434434434

Table 2: Optimal α values for different strike prices (K).

Furthermore, the convergence of optimal values of α with respect to K is visualized in the plot shown in Figure 6. The plot aswell as the table demonstrate that the optimal values of α increase as the strike price increases. We can say that there is a proportional relationship between the strike price and the optimal tilting parameter.

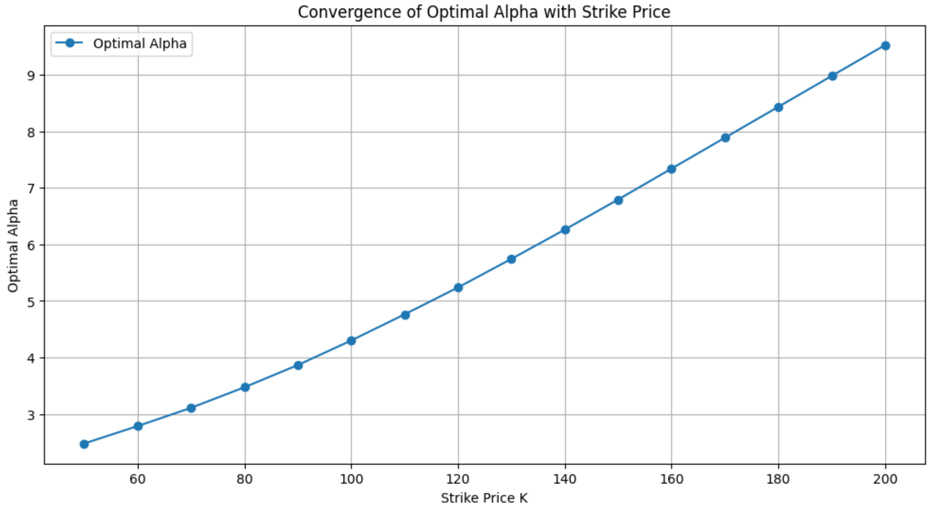

Figure 6: Convergence of optimal values of α w.r.t K

Question 3. Use Fourier inversion to compute the prices of the same call options from the characteristic function of the random variable x_T in the Heston model via numerical integration. Compare the Fourier prices with the Monte-Carlo results.

The implementation for this calculation is shown in Figure 7.

```python
def call_price_heston(alpha, V_t, x_t, T, t, K, r, kappa, theta, sigma_v, rho):

    def integrand(xi):
        return integrate_function(xi, alpha, V_t, x_t, T, t, K, r, kappa, theta, sigma_v, rho)

    integral_result, _ = quad(integrand, -100, 100)
    return integral_result

call_option_prices = []

for K in strike_prices:
    optimal_alpha = optimal_alphas[strike_prices.tolist().index(K)]
    price = call_price_heston(optimal_alpha, V_t, x_t, T, t, K, r, kappa, theta, sigma_v, rho)
    call_option_prices.append(price)
    print(f"Call Option Price for K={K}: {price:.5f}")
```

Figure 7: Computation of Fourier prices

The computed Fourier prices as well as the comparison with the Monte Carlo results can be found in Table 3. The table demonstrates that call option prices computed by the Fourier and Monte Carlo methods are very close, with small absolute differences across all strike prices. This indicates consistency between the two approaches.

Strike (K)	MC (Heston Call Price)	Fourier (Call Option Price)	Absolute Difference
50	59.05862403	59.09994103	0.04131700
60	50.99648376	51.03639314	0.03990938
70	43.15487233	43.19218215	0.03730982
80	35.71388657	35.74674834	0.03286177
90	28.86653448	28.89594260	0.02940812
100	22.77973500	22.80699185	0.02725685
110	17.55963316	17.58459598	0.02496282
120	13.23717502	13.25823853	0.02106351
130	9.77068660	9.78927787	0.01859127
140	7.07659514	7.08995228	0.01335714
150	5.03655585	5.04568281	0.00912696
160	3.52870920	3.53458656	0.00587736
170	2.43886208	2.44137641	0.00251433
180	1.66587940	1.66535263	0.00052677
190	1.12521509	1.12358349	0.00163160
200	0.75322773	0.75081936	0.00240837

Table 3: Comparison of Call Option Prices for Different Strikes (K)