Python Option Pricing with Fourier Transform Methods

Andrew Wu

September 2020

1 Introduction

This project implements Fourier Transform Pricing methods in Python (notably, the Fast Fourier Transform approach by Carr and Madan 1999) to price European Call Options. Two different underlying stock processes are used: the traditional Geometric Brownian Motion (GBM) process used in the Black-Scholes-Merton Model, and the Variance-Gamma (VG) Process (Madan, Carr and Chang 1998). The computation times for these methods are then compared in Python.

2 Methods

NOTE: In all the following sections, we denote C_0 as the initial call price, S_t as the price of the underlying asset at time $0 \leq t \leq T$, T as the option's maturity, K as the strike-price, r as the risk-free rate and $\mathcal{Q}$ as the risk-netural measure. We assume the underlying asset does not pay dividends. The price of a call option is given by

$$C_0 = e^{-rT}\mathbb{E}^{\mathcal{Q}}\left[\max(S_T - K, 0)\right]$$

2.1 Black-Scholes-Merton Model

In the traditional Black-Scholes-Merton Model, the underlying stock is assumed to follow a Geometric Brownian Motion:

$$S_t = S_0 \exp\left[\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t\right],$$

where W_t is a $\mathcal{Q}$-Brownian Motion. The resulting price of the European call is

$$C_0 = S_0 N(d_1) - Ke^{-rT}N(d_2),$$

where $N(\cdot)$ is the cumulative distribution function of a standard normal random variable and

$$d_1 = \frac{\ln\left(S_0/K\right) + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}, \qquad \text{and} \quad d_2 = d_1 - \sigma\sqrt{T}.$$

This price can be calculated in the Python code by defining a call option, setting the underlying stock `S` to an instance of the `GBM` class and using the `.BlackScholesPrice` method.

2.2 Fourier Inversion Method

In general, for any underlying process, the initial price of a call option can be written as

$$C_0 = S_0\Pi_1 - Ke^{-rT}\Pi_2,$$

where Π_1, Π_2 are the option's delta and the risk-neutral probability of finishing in the money respectively. In the Black-Scholes-Merton model, we have simple formulas for these terms; $\Pi_1 = N(d_1)$ and $\Pi_2 = N(d_2)$.

A flaw in the Black-Scholes Model is that from empirical observation, log-returns are not normally distributed; they are leptokurtic. To address this shortfall, we can replace the Arithmetic Brownian Motion process $\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t$ with a more complicated stochastic process. Madan, Carr and Chang advocate using a Variance-Gamma Process in their 1998 paper, which allows for random jumps:

$$S_t = S_0 \exp\left[rt + X_t(\sigma, \theta, \nu) + \omega t\right],$$

where $X_t(\sigma, \theta, \nu)$ is a VG process. For processes such as these, the cumulative distribution functions are extremely complicated, and either have no closed form, or require special mathematical functions such as Bessel functions and the confluent hypergeometric function (Matsuda 2004).

Thus, a more general approach is calculate Π_1 and Π_2 using the connection between characteristic functions and densities through the Fourier Transform. In fact, the characteristic function Φ is exactly the Fourier Transform of the density function f of a random variable X:

$$\phi(u) := \mathbb{E}[e^{iuX}] = \int_{-\infty}^{\infty} e^{iux} f(x) dx.$$

Characteristic functions uniquely define a probability distribution and tend to be simpler than density functions. Carr and Madan briefly mention a variation of the Gil-Pelaez inversion theorem, which allows the call delta and probability to be calculated as (Scott 1997):

$$\Pi_1 := \frac{1}{2} + \frac{1}{\pi}\int_0^{\infty} \Re\left(\frac{e^{-iu\ln K}\phi_T(u-i)}{iu\phi_T(-i)}\right) du.$$

$$\Pi_2 := \mathbb{P}\left(S_T > K\right) = \frac{1}{2} + \frac{1}{\pi}\int_0^{\infty} \mathrm{Re}\left(\frac{e^{-iu\ln K}\phi_T(u)}{iu}\right) du.$$

Note: $\phi_T(\cdot)$ is the characteristic function of S_T, the terminal price of the underlying asset. This price is given by `.cdfFTPrice` and can be used with any underlying Stock process.

2.3 Modified-Call Fourier Inversion

In their 1999 paper, Carr and Madan developed a new method to address the issue of the singularity at $u = 0$ in the previous Fourier-Inversion method. This addresses numerical integration issues and allows the application of the Fast-Fourier Transform, a more efficient method to estimate the integrals. First, Carr and Madan work in the log-space to simplify notation; define $k = \ln K$, $s_t = \ln S_t$ and let ϕ_T be the characteristic function of the terminal log-asset price s_T. Furthermore, define a modified-call price $c_T(k)$

$$c_T(k) = e^{\alpha k} C_T(k),$$

where $C_T(k)$ is the initial price of the call option with maturity T and log-strike k. The factor $e^{\alpha k}$ addresses the singularity issues when an appropriate value of α is chosen. Let ψ_T be the Fourier transform of $c_T(k)$, then it can be shown that

$$\psi_T(v) = \frac{e^{-rT}\phi_T(v - (\alpha+1)i)}{\alpha^2 + \alpha - v^2 + i(2\alpha+1)v}.$$

As the characteristic function ϕ_T is relatively simple, ψ_T can easily be calculated before applying the inverse Fourier transform and dividing by $e^{\alpha k}$ to recover the call price:

$$C_T(k) = \frac{e^{-\alpha k}}{\pi} \int_0^\infty e^{-ivk} \psi_T(v) dv.$$

This is already a valid method to calculate the call price if we perform quadrature on the above integral and is obtained through the code using `.CMFTPrice` (Call-Modified Fourier Transform).

2.4 Fast-Fourier Transform

Carr and Madan improve the speed of the above method by using the Fast-Fourier Transform algorithm in the quadrature of the integral. Define a uniform partition of N points where $v_j = \eta j$ for $j = 0, 1, 2...N-1$ and η is the spacing size. This implies truncating the improper integral at $N\eta$. Then, using Simpson's rule weights to increase accuracy and a range of log-strikes defined by

$$k_u = -b + \lambda u, \quad \text{for } u = 0, 1, ...N-1,$$

where $b = \frac{1}{2} N\lambda$ and $\lambda\eta = \frac{2\pi}{N}$, the Fast-Fourier Transform can be directly applied to compute a range of call prices.

$$C(k_u) \approx \frac{e^{-\alpha k_u}}{\pi} \sum_{j=0}^{N-1} e^{-2\pi iju/N} e^{ibv_j} \psi_T(v_j) \cdot \frac{\eta}{3} \left[3 + (-1)^{j+1} - \delta_j\right]$$

where the Kronecker delta is defined as $\delta_j = 1$ if $j = 0$ and 0 otherwise. The drawback of the FFT method is that relies on a predefined strike-price spacing, not only is it difficult to calculate call prices for arbitrary strikes, but most of the prices are not useful (these options are very far in or out-of the money). Furthermore, it should be note there is a tradeoff between pricing accuracy by selecting higher N and small strike-spacing (lower N).

Due to its dependence on a predefined strike partition, the `FFTPrice` function is included outside the call option class with separate helper functions to define the log-strike partition and desired upper/lower bounds for the strike prices.

2.5 Monte-Carlo Simulations

Monte-Carlo simulations are an appropriate benchmark to compare our pricing methods against, especially when the underlying process is very complex. In this project, a default value of `N=200` subintervals are used to generate a sample path for the underlying Stock. The payoff of the call under the i^{th} $\max\left(S_T^{(i)} - K, 0\right)$ is then calculated and averaged over `n` simulations (500 by default) and discounted, resulting in a suitable estimate for the call option price:

$$\hat{C}_0 = \frac{e^{-rT}}{n} \sum_{i=1}^{n} \max\left(S_T^{(i)} - K, 0\right).$$

The sample paths under both the GBM and Variance Gamma Processes are generated by a similar algorithm:

1. Set $dt = T/N$ and generate a uniform partition $t_j := j \cdot dt$, for $k = 0, 1, 2, ...N$ on $[0, T]$.

2. (a) If the underlying process is a GBM, then generate N random numbers from $\mathcal{N}(0, dt)$ and store in the vector ΔW. These will serve as the Wiener Process increments.

(b) If the underlying is a VG-Process, then generate N random numbers from $\Gamma(dt/\nu, \nu)$ and store in the vector ΔG. Also Generate N standard-normal random numbers in the vector Z. (Note: The gamma parameters are defined consistently with Numpy)

3. Create a $N+1$ sized vector, setting $W_0 = 0$ for the GBM or $X_0 = 0$ for a VG process. Use a cumulative sum function to obtain the simulated value for each process for $t_k, \quad k = 1, 2, 3...N$.

   (a) Brownian Motion:

$$W(t_k) = \Delta W_k + W(t_{k-1}) = \sum_{j=1}^{k} \Delta W_j$$

   (b) VG-Process (Korn and Korn 2010):

$$X(t_k) = X(t_{k-1}) + \theta \Delta G_k + \sigma \sqrt{\Delta G_k} Z_k = \theta \sum_{j=1}^{k} \Delta G_k + \sigma \sum_{j=1}^{k} \sqrt{\Delta G_j} Z_j$$

4. Apply the appropriate exponential transformation (see definitions of GBM/VG in the previous sections) to generate a sample path of the underlying asset. Note, in the call-price calculations, only the terminal value of the asset is used.

3 Results

Each pricing method was applied to price a range of call options of strike prices between 70 and 130, using underlying process models (GBM and VG). The parameters for GBM were $S_0 = 100, r = 0.05$, $\sigma = 0.1$ and $T = 1$. The parameters for VG were the same as case 4 in Carr and Madan's paper: $S_0 = 100, r = 0.05, \sigma = 0.25, \nu = 2, \theta = -0.1$, $T = 1$. The table below displays the absolute and relative errors of the FFT pricing method:

	Absolute	**Relative**
GBM	2.1708e-07	0.79%
VG	2.4931e-08	0.06%

Table 1: Table comparing the absolute and relative errors of the FFT pricing method when the underlying stock process is a Geometric Brownian motion or Variance-Gamma process.

The methods were timed using Python's `time.time()` and averaged over 10 runs. The average time used by each method in seconds is shown in the tables below:

Method	**Time (s)**
Fast Fourier Transform	0.00
Black-Scholes Merton	0.01
Fourier Inversion	0.61
Monte Carlo	3.96

Method	**Time (s)**
Fast Fourier Transform	0.00
Modified Call	2.64
Fourier Inversion	5.35
Monte Carlo	5.74

Tables 1 and 2: (*left*) times for the GBM methods, (*right*) times for the VG methods.

Clearly, the Fast Fourier Transform vastly outperformed the other methods in terms of efficiency, even with the calculation of additional unnecessary call prices. Monte Carlo simulations performed the worst each time, suggesting this method is not particularly viable for real-time option pricing. It was noted that even with 500 simulations, the Monte Carlo price still performed worse than the Fourier Inversion

method in the case of the GBM underlying in terms of accuracy when compared to the analytic formula. The Fourier Inversion method however, performed very poorly on a VG underlying possibly due to issues with the singularity at 0. The FFT approach both accurate and extremely fast and it should definitely be considered as an alternative pricing method.

4 References

1. Carr, P, Madan, D.B, 1999, "Option Valuation using the Fast Fourier Transform", *Journal of Computational Finance*, 2, 61-63.

2. Scott, L., 1997, "Pricing stock options in a jump diffusion model with stochastic volatility and interest rates: Application of Fourier inversion methods.", 7, 413-426.

3. Madan, D.B., Carr, P., and Chang, E.C., 1998, "The variance gamma process and option pricing.", *European Finance Review*, 2, 79-105.

4. Matsuda, K, 2004, "Introduction to Option Pricing with Fourier Transform: Option Pricing with Exponential Lévy Models", PhD Thesis, The City University of New York.

5. Korn, E, Korn, R, 2010, "Monte-Carlo Methods and Models in Finance and Insurance", *Chapman and Hall/CRC*, Section 7.3.3