Options Pricing in Lévy Models

Joseph Loss[1], Yuchen Duan[1], Daniel Liberman[2]

We consider options pricing in Lévy models, specifically the implementation of two algorithms listed in Feng[3] and compared our results with those published in the original paper. Like Feng, we assume that the asset price follows a geometric Lévy process under a given and equal martingale measure. This is expressed as:

$$S_t = S_0 e^{X_t}$$

Note that X_t is the Lévy process at $t = 0$ and that S_0 is the asset price at $t = 0$.

Algorithm 1: Normal Inverse Gaussian

Our first algorithm is an implementation of the normal inverse Gaussian (NIG) process, which we simulated as a subordinated Brownian motion, to compute the price of a European put option contract. The NIG process is characterized by:

$$X_t = \mu t + \beta z_t + B_{z_t}$$

where B_t is a standard Brownian motion, z_t is an independent inverse Gaussian process and $\mu = r - q + \delta\left(\sqrt{\alpha^2 - (\beta+1)^2} - \sqrt{\alpha^2 - \beta^2}\right)$.

Note that $\alpha, \beta, \delta, r, q$ are inputs given by Feng and we discuss these on the following page.

For $t > 0$, we simulate X_t with the following steps:

1. Generate a standard normal random variable using the Box-Muller algorithm described on pg. 66 of Glasserman[4]. With $\gamma = \sqrt{\alpha^2 - \beta^2}, Z = \frac{G_1^2}{\gamma}$, we compute

$$\zeta = 1/\gamma(\delta t + \frac{1}{2}Z - \sqrt{\delta t Z + \frac{Z^2}{4}}$$

2. Generate a uniform random variable U on $(0, 1)$. If $\left(U < \frac{\delta t}{\delta t + \gamma\zeta}\right)$, then $z_t = \zeta$. Otherwise, $z_t = \frac{\delta^2 t^2}{\gamma^2 \zeta}$.
3. Generate a standard normal random variable G_2. Compute $X_t = \mu t + \beta z_t + \sqrt{z_t} G_2$.

---

[1] University of Illinois at Urbana-Champaign, M.S. Financial Engineering

[2] University of Illinois at Chicago, Finance

[3] Feng, Liming, et al. "Simulating Lévy Processes from Their Characteristic Functions and Financial Applications." University of Illinois, 30 July 2011.

[4] Glasserman, Paul. "Monte Carlo Methods in Financial Engineering." Springer-Verlag, 2003.

To compute the price of a European vanilla put option, we use the inputs given in Section 6.1 (pg. 22) of Feng: $\alpha = 15, \beta = -5, \delta = 0.5, r = 0.05, q = 0.02, S_0 = K = 100, T = 0.5$.

The price of the option at $t = 0$ can be calculated as:

$$V = S_0 e^{-rT} \mathrm{E}[\mathrm{f}(\mathrm{X_T})]$$

where X_t is a Lévy process and $f(x) = \max(0, K/S_0 - e^x)$.

Using the inputs above, we computed the value of the option, $6.25836. We compared this with the Black-Scholes model for a European option in the "RQuantLib" package:

```
> put_prices
       nig.put.value ql.put.value
value  6.258536      6.261323
```

Algorithm 2: Inverse Fourier Transform

The second option pricing model is an implementation of the inverse transform method from tabulated probabilities which, depending on the desired accuracy required, could be multiple times faster than the normal inverse gaussian process.

Note tInput parameters are taken from Section 6.1, pg. 22 of Feng's paper.

First, we begin by initializing arrays for the lists of variables χ and $\hat{F}$. Brute-Force-Search is used in place of the binary search originally prescribed in Section 3.1. This can be seen in the function templates for chi and Fhat.

There are two more function templates that build the foundation for this algorithm. The first, "Fhat Distribution Function", is the direct implementation of the distribution in equation 3.12 on pg. 8 of Feng's paper. The second, termed "Inverse Transform Function", implements the approximation to $F^{-1}(U)$ on pg. 8 of the paper:

$$x_k + \frac{x_{k+1} - x_k}{\hat{F}_{k+1} - \hat{F}_k}(U - \hat{F}_k).$$

This performs the inverse transform function for each generated U between 0 and 1, utilizing brute-force search to find $0 \le k \le K - 1$ so that $\hat{F}_k \le U < \hat{F}_{k+1}$:

$$\hat{F}(x) = \begin{cases} 0, & x < x_0 \\ \hat{F}_{k-1} + \frac{\hat{F}_k - \hat{F}_{k-1}}{\eta}(x - x_{k-1}), & x_{k-1} \le x < x_k, 1 \le k \le K \\ 1, & x \ge x_K \end{cases} .$$

```
> Fhat.list
 [1] 0.00000000 0.04545455 0.09090909 0.13636364 0.18181818 0.22727273
 [7] 0.27272727 0.31818182 0.36363636 0.40909091 0.45454545 0.50000000
[13] 0.54545455 0.59090909 0.63636364 0.68181818 0.72727273 0.77272727
[19] 0.81818182 0.86363636 0.90909091 0.95454545 1.00000000
```

The Inverse Transform algorithm begins with its parameters on line 102; these parameters are taken from Section 6.1 on pg. 22 of Feng's paper. In computing the European put price, there are two formulas used. The first method uses the formula for "European Vanilla Put Options" given in Section 5.4, pg. 19 of Feng's paper (note the max function $= \max\left(0, \frac{strike}{S_0} - e^x\right)$. The second method uses the more general, $\max(0, strike - S_0 e^{X_T})$.

The computed option prices are averaged over the number of MonteCarlo iterations and an example output is shown below:

```
> values.table
     put.prc.fft1 put.prc.fft2
[1,]     4.573026     4.585146
```

Note that both prices converge to a value of $4.58 for the European put option. This is extremely close to the $4.589 value of Feng's implementation (listed on pg. 24).

Final Thoughts

We are currently working on making both algorithms available in a package format, which can be downloaded, used, and improved by the community.