1 Option Calibration

The problem of calibrating option prices to market values (the "inverse problem") is non-trivial especially with complex pricing models with many parameters. A naive approach is to perform optimization by minimizing a distance between the prices provided by the market and the modeled prices by varying the input parameters. However, this can be computationally intensive. The problem is not convex and there may be a plethora of local minima. The parameter surface may have many "flat" areas leading to unstable parameter solutions.

In our study, we focus on calibrating models defined in the Option Calculation paper. We use a jump-diffusion a la Merton (1976) while extending the model to allow for a time-changed diffusion component. The models thus incorporate popular models like Heston's as a special case.

The code which runs the results shown below is available at the following Github repo: fang_oost_cal_charts.

2 The model

Levy processes can be constructed with relatively simple characteristic exponents. In this paper we will focus on processes with characteristic exponents of the following form:

$$\begin{aligned}\psi(u) &= -\frac{\sigma^2 u^2}{2} + \lambda\left(1 - e^{\mu_j + \sigma_j^2/2}\right) + \lambda\left(e^{\mu_j ui - \sigma_j^2 u^2/2} - 1\right) \\ &= -\frac{\sigma^2 u^2}{2} - \psi_j(-i) + \psi_j(u)\end{aligned}$$

Where $\psi_j(u) := \lambda\left(e^{\mu_j ui - \sigma_j^2 u^2/2} - 1\right)$ is the characteristic function of the jump component. $\psi(u)$ is the characteristic exponent of a Merton jump-diffusion model. Following Carr and Wu (2004), we also introduce a random time change so that the characteristic function of the normalized log asset is as follows:

$$\mathbb{E}[e^{uiX_t}] = g(-\psi(u), a, a - \sigma_v \rho u \sigma, \sigma_v, v_0)$$

Where $X_t = \log\left(\frac{S_t}{S_0}\right) - rt$, and g is the moment generating function of an integrated CIR process:

$$g(x, a, \kappa, \sigma_v, v_0) = e^{-b(t)v_0 - c(t)}$$

Where

$$\begin{aligned}b(t) &= 2x\left(1 - e^{-\delta t}\right) / \left(\delta + \kappa + (\delta - \kappa)e^{-\delta t}\right) \\ c(t) &= \left(\frac{a}{\sigma^2}\right)\left(2\log\left(1 + (\kappa - \delta)\left(1 - e^{-\delta t}\right)/2\delta\right) + \left(1 - e^{-\delta t}\right)(\kappa - \delta)\right) \\ \delta &= \sqrt{\kappa^2 + 2x\sigma_v^2}\end{aligned}$$

3 Calibration

Calibration has traditionally taken the following form:

$$\min_{\theta} \sum_{k} w_k \left(C_k - C(k;\theta)\right)^2$$

Where w_k is a weight, θ are the parameters describing the (risk-neutral) asset process, C_k is the observed option prices at log-strike k, and $C(k,\theta)$ is the modeled price.

As mentioned in the introduction, this problem is not trivial. See Cont and Tankov (2006) for details. Since we are dealing with Levy processes, we instead consider minimizing the following:

$$\min_{\theta} ||\psi(u_l;\theta) - \hat{\psi}(u_l;k)||$$

We can borrow from Carr and Madan (1999) and Belomestny and Reiss (2006) to create the estimate $\hat{\psi}$ from the observed option data:

$$\log\left(1 + iu(iu+1)\int_{-\infty}^{\infty} e^{uix} O(x)dx\right) = \psi(u-i,t)$$

Where $O(x) = C_{x+\log(S_0)+rt}/S_0 - (1-e^x/S_0)^+$ and $x = k - \log(S_0) - rt$. Since we do not observe a continuum of option prices in the market, we use a monotonic spline to interpolate the option prices. To preserve accuracy, we use a two part spline fit. The first fit uses the realized $O_k = C_k/S_0 - \left(1-e^{k-rT}\right)^+$ from the minimum log strike until the last strike that satisfies $e^k/S_0 < 1$. The second fit is the log of normalized option prices $\log\left(C_k/S_0\right)$ and is fit from the last strike that satisfies $e^k/S_0 < 1$ until the maximum log strike.

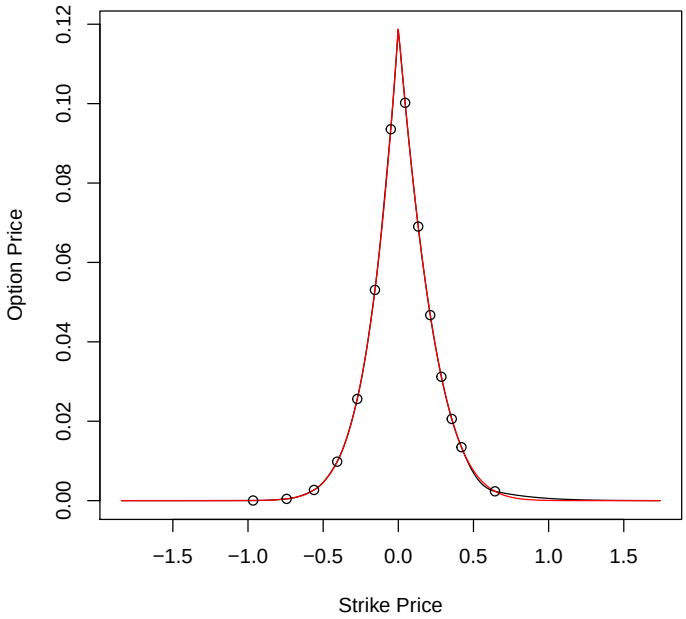
There are three sources of error in the estimate:

1. The observation error in the options themselves
2. The error in the spline approximation
3. The error in numerically integrating the observed options prices

3.1 Spline Error

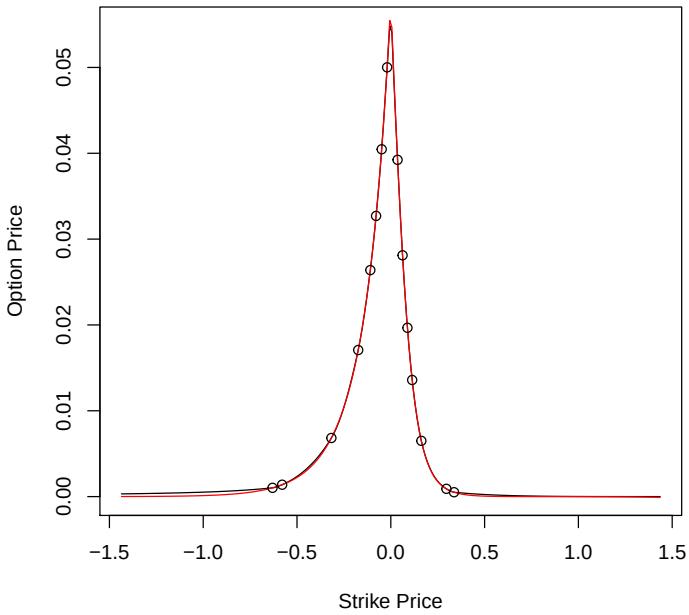
3.1.1 Black Scholes

We first test the spline error on a Black-Scholes model. The parameters chosen are $S = 10, r = 0, t = 1, \sigma = .3$, and the strike array $4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16$. This model shows very good alignment when choosing a spline on the log of the option price against the strike as can be seen in the following charts. Note that the black line is estimated, while the red line is actual. The red line nearly completely covers the black line.



3.1.2 Heston

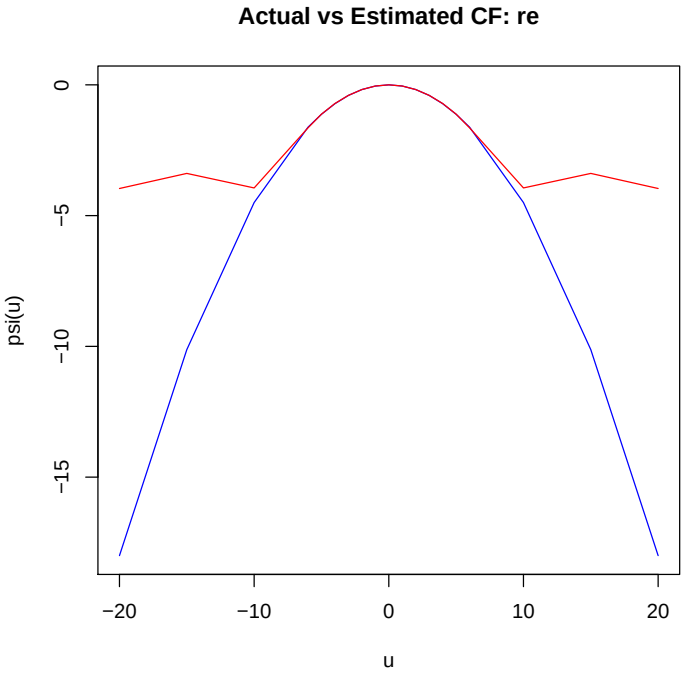
We then fit the spline for a Heston model. The parameters chosen are the strike array $(95, 100, 130, 150, 160, 165, 170, 175, 185, 190, 195, 200, 210, 240, 250)$.

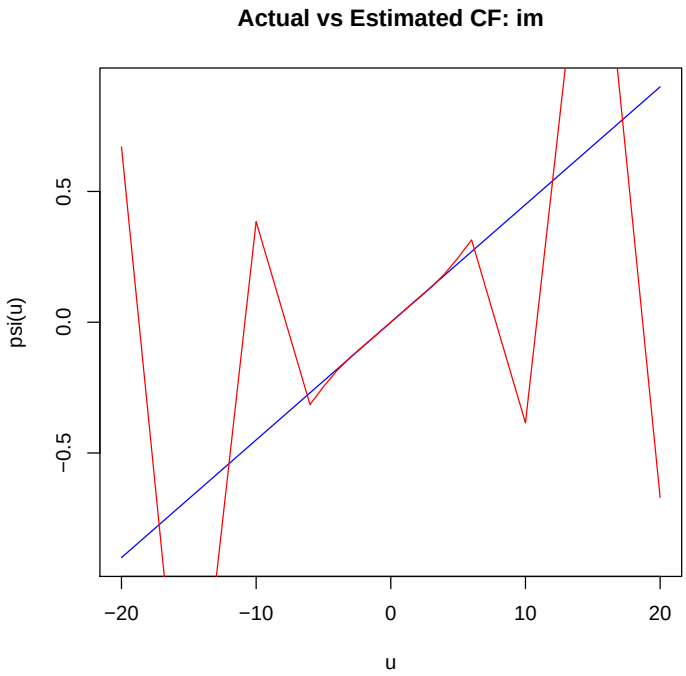


3.2 Numerical Integration Error

3.2.1 Black Scholes

The numerical integration error can be seen as follows:

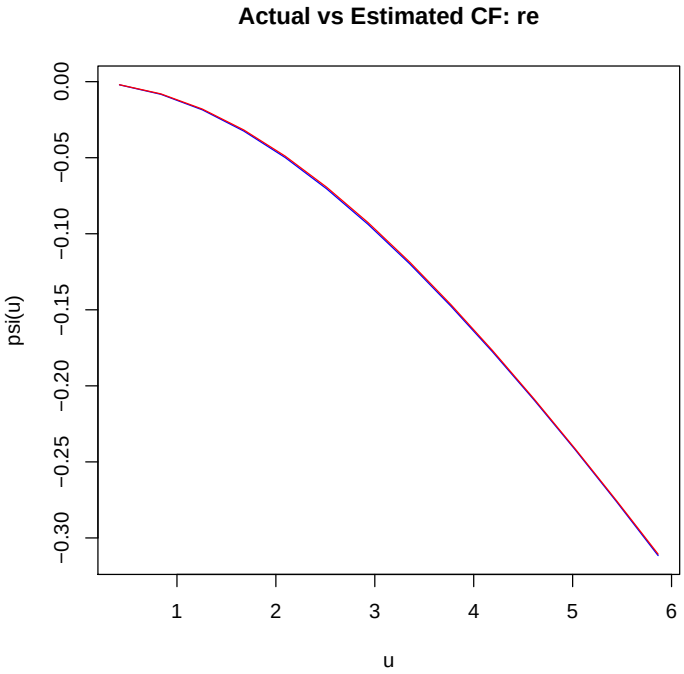

Actual vs Estimated CF: re
psi(u)
0
−5
−10
−15
−20
−10
0
10
20
u

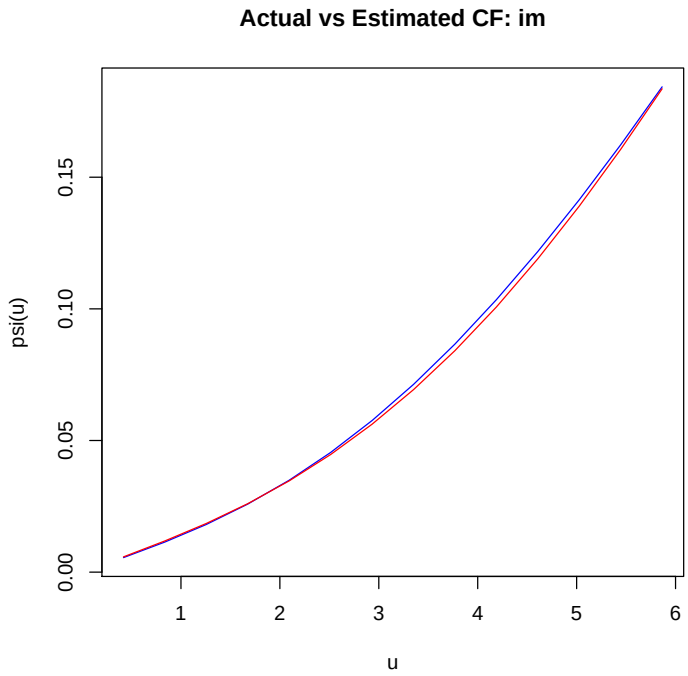


While in theory the integration should be valid over all real numbers, it appears that it is only accurate from $(-2\pi, 2\pi)$. However, since the function is even, we don't benefit from using all the domain and truncate it from $(0, 2\pi)$.

3.2.2 Heston

The following plot shows the integration comparison for a Heston model.


Actual vs Estimated CF: re
psi(u)
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
1
2
3
4
5
6
u



The model shows very good alignment.

3.3 Estimates

3.3.1 Black Scholes

Estimating the characteristic function at points u between negative and positive 2π, we use the cuckoo search algorithm to minimize the sum of the norms of the differences between the estimated and modeled characteristic functions. For a simple model like Black Scholes we can also use gradient descent, however for more complicated models the objective function may have many local minima.

```
  actual   optimal parameter
1    0.3 0.3007071     sigma
```

3.3.2 Heston

The search for a Heston model shows the following results:

```
      actual    optimal parameter
1  0.1994994  0.1520214     sigma
2  1.5768000  0.5438498     speed
3  2.8827158  2.2027331     ada_v
```

```
4 -0.5711000 -0.5870491       rho
5  0.4396985  1.2327491    v0_hat
```

Note that the objective function's surface can be very "flat", leading to high variance in the parameters.

3.4 Cuckoo Search

The cuckoo search algorithm is a derivatives-free optimization algorithm which uses Monte Carlo simulations to traverse high-dimensional space. The algorithm scales well to high dimensions and can achieve an acceptable degree of accuracy quickly.