IEOR E4602: Quantitative Risk Management (Spring 2016)

Columbia University

Instructor: Martin Haugh

Assignment 6: Due Wednesday 20^{th} April 2016

Answer 4 of the 5 questions. MSFE students musts answer Questions 2 to 5.

1. **(Delta-Hedging)**

   Write a computer program that simulates m sample paths of the delta-hedging of a long position in a European option in the Black-Scholes model. Your code should take as inputs the initial stock price S_0, option expiration T, implied volatility σ_{imp}, risk-free rate r, dividend yield q and strike K as well as whether the option is a call or put. You should assume that at time $t = 0$ you have a short position in the option as well as an initial cash position equal to the value for which you sold the option. The total P&L at time T is then equal to the total value of your portfolio (including the short option position) at that time. (Note that the total initial value is 0.)

   Your code should also take as inputs: (i) the number of re-balancing periods N and (ii) the drift and volatility, μ and σ respectively, of the geometric Brownian motion used to simulate a path of the underlying stock price. Note that σ_{imp} and σ need not be the same.

   At the very least your code should output the average option payoff and the average terminal total P&L (across the m paths) from holding the option and executing the delta-hedging strategy. See Section 4 of the *Black-Scholes Model* lecture notes for further details.

   Once you have tested your code answer the following questions:

   (a) When $\sigma_{\text{imp}} = \sigma$ how does the total P&L behave as a function of N? What happens *on average* if $\sigma_{\text{imp}} < \sigma$? If $\sigma_{\text{imp}} > \sigma$?

   (b) For a fixed N, how does the total P&L behave as $\sigma_{\text{imp}} = \sigma$ increases?

   (c) How does the drift, μ, affect the total P&L when $\sigma_{\text{imp}} = \sigma$?

2. **(Option pricing under Merton's jump-diffusion model)**

   Consider the Merton jump-diffusion model described in equation (7) of the *Model Risk* lecture notes. Assume that the given dynamics are under an *equivalent martingale measure* (EMM), $\mathcal{Q}$. Suppose also that $\mathrm{E}^{\mathcal{Q}}[Y_i] = \mu_y$ for all $i \in \mathbb{N}$.

   (a) Find an expression for $\mathrm{E}_0^{\mathcal{Q}}[S_T]$ in terms of S_0, r, q, λ and μ_y. Explain why it must be the case that $r - q = \mu + \lambda(\mu_y - 1)$.

(b) Write the price of a call option in the Merton jump-diffusion model as an infinite weighted sum of Black-Scholes type call option prices. Write a piece of code to compute the value of the option price. (As you obviously cannot compute the infinite sum, you should include just enough terms so that your sum has converged, i.e., so that the resulting approximation error is negligible.)

(b) Set $S_0 = 1$, $r = 2\%$, $q = 1\%$, $\sigma = 20\%$, $\lambda = .1$ and $\log(Y_i) \sim \mathrm{N}(-.05, .1)$. Now write a piece of code that computes call options prices at a series of strikes ranging from 50% to 150% of S_0 and time-to-maturities ranging from 1 week to 1 year.

(c) Compute the Black-Scholes implied volatility of your options prices and create a plot of the resulting implied volatility surface.

3. **(Characteristic function of** $\log(S_T)$ **in time-changed Levy processes)**

   Recall the $\mathcal{Q}$-dynamics for a security price, S_t, with time-changed Levy dynamics can be taken to satisfy

$$S_T \;=\; S_0 e^{(r-q)T} \frac{e^{X_{Y_T}}}{\mathbf{E}_0^{\mathcal{Q}}\left[e^{X_{Y_T}} \,|y_0\right]}.$$

   Show that the characteristic function of the $\log(S_T)$, i.e. the log-stock price is given by

$$\begin{aligned} \phi(u,T) &:= \mathrm{E}\left[e^{iu\ \log(S_T)} \mid S_0, y_0\right] \\ &= e^{iu((r-q)T+\log(S_0))} \,\frac{\varphi(-i\psi_x(u);T,y_0)}{\varphi(-i\psi_x(-i);t,y_0)^{iu}} \end{aligned} \tag{1}$$

   where

$$\psi_x(u) := \log \mathrm{E}\left[e^{iuX_1}\right] \tag{2}$$

   is the characteristic exponent of the Levy process and $\varphi(u;T,y_0)$ is the characteristic function of Y_T given y_0.

   Hint: This can be done in 10 lines or so. At some point you will need an expression for $\mathrm{E}_0\left[e^{iuX_{Y_T}}\right]$*. You can compute this by conditioning on* Y_T *and then using the stationary and independent increments property of a Levy process.*

4. **(Correlation in the Gaussian copula model)** Suppose there are N credits in a portfolio and that the time to default, τ_i, of the i^{th} credit is exponential with parameter λ_i for $i = 1, \ldots, N$.

   (a) Show that $\tau_i = \epsilon_i/\lambda_i$ where ϵ_i is an exponential random variable with mean 1.

(b) Suppose we assume the copula of the τ_i's is Gaussian with an equi-correlation matrix where all off-diagonal elements are equal to $\rho > 0$. What if any relationship is there between the ϵ_i's? Justify your answer.

(c) Without doing any analysis, do you think that in the perfect correlation case, i.e. when $\rho = 1$, that you are likely to see defaults clustered in *time*. Now answer the question again after doing some (very simple) analysis.

(d) What do your results say about the use of the Gaussian copula model for generating defaults clustered in time?

5. **(Some more VBA!)**

   Write a VBA function that uses Monte-Carlo simulation to estimate the expected loss on a CDO tranche with lower and upper attachment points of L and Y, respectively. You can assume that defaults in the reference portfolio follow the Gaussian copula model with a constant correlation, ρ. Your code should take as inputs the following parameters: $N = \#$ names in the portfolio, L, U, ρ, $R =$ recovery rate, $NumSims =$ # of simulations and an $N \times 1$ array of default probabilities, $DefaultArray$. Your code should read all of these inputs in from a spreadsheet and write out to the same spreadsheet the estimated expected loss as well as an approximate 95% confidence interval for the expected loss. (In order to do this you will need to have a VBA function that computes the Cholesky Decomposition of a correlation matrix. It is straightforward to find such a function online using the *Google* search engine, for example. Of course it is always a good idea to test the code first.)