mini Project: variance swap

Minglei Wang[1]

[1] *Yale University, New Haven, CT*

(Dated: March 31, 2017)

Answers are provided to all the three questions in this Project. In Question 1, I implemented replication valuation strategy using C++ and Python. The solution set should contain all the source codes, compilation file, sample data file, and sample input files. The data is real market data taken from CBOE website [4]. The C++ implementation uses naive integration method and pair-wise approximation method, and the Python uses only naive integration. All the codes and method give similar result, which is around $(12.6927\%)^2 \sim (12.9354\%)^2$. For Question 2, three factors are accounted for, spacing of strike ΔK, range of strike K, and liquidity of options under consideration. The result is that, emergence of larger ΔK over-estimates variance strike, the incomplete market under-estimates variance strike, and liquidity factor gives ambiguous influence. For Question 3, the variance swap is superior in that it provides *pure* exposure to variance, which doesn't depends on spot price. On the other hand, the sensitivity of vanilla option values to variance depends on spot price and would be vanishing for deep in the money or out of the money options. The intuition, as well as equations for $\partial V/\partial\sigma^2$, are given in the end.

I. QUESTION 1

Write a self-contained Python script to calculate via replication the current Fair value Strike of the SPX variance swap that expiries on April 21st. The relevant data needed is on the CBOE website if you need vanilla option quotes(assume you are pricing as of this past Fridays close (March 24)).

After reviewing the literature [1–3], the replication strategy for variance swap consists of a rebalanced stock position and a log contract. The log contract payoff can be further decomposed into a forward contract, and a series of OTM standard puts and calls weighted to $1/K^2$, where K is strike price for each options. The fair value strike can be then obtained by [1]:

$$K_{var} = \frac{2}{T}\left(rT - \left(\frac{S_0}{S_\star}e^{rT} - 1\right) - \log\frac{S_\star}{S_0} + e^{rT}\left(\int_0^{S_\star}\frac{1}{K^2}P(K)dK + \int_{S_\star}^{\infty}\frac{1}{K^2}C(K)dK\right)\right) \tag{1}$$

where K_{var} is the fair value strike, T time to maturity, r continuously compounded annualized risk-free rate, S_0 the spot price, $S_\star$ reference price typically chosen to be close to spot price to allow access to liquid options information, $P(K)C(K)$ represents current price for puts and calls respectively at a given strike level. We use Π_{cp} as the present value of the option portfolio with future payoff $f(S_T)$:

$$\Pi_{cp} = \int_0^{S_\star}\frac{1}{K^2}P(K)dK + \int_{S_\star}^{\infty}\frac{1}{K^2}C(K)dK \tag{2}$$

$$f(S_T) = \frac{S_T - S_\star}{S_\star} - \log\frac{S_T}{S_\star} \tag{3}$$

There are two methods to calculate Π_{cp}, a naive integration method as in [2] and pair-wise approximation of future payoff function $f(S_T)$ outlined in Appendix A in [1]. In the C++ implementation, variSwap class member function variSwap::varianceStrike() uses the naive integration method, while the other member function variSwap::varianceStrikeBetter() uses the pair-wise approximation method. The python implementation uses only the naive integration method to confirm the result.

This paragraph explains relevant data acquired from website. On March 24, the SPX calls and puts prices, expiring at April 21, were obtained from CBOE website [4]. The strike ranges from 1500 to 2560 with 5 points apart, sorted in ascending order in sample text data files, odd lines for calls even lines for puts. The option closing prices on March 24 were used as input to programs. The spot price of SPX was determined as average of bid and ask. The continuously compounded annualized risk-free rate is estimated by Libor rate on March 24 for USD 12 months maturities [5], $r = \ln(1 + 1.80289\%) = 0.0178683$. Time to maturity $T = 28/365 = 0.076712328$.

A. C++ implementation

The main function (main.cpp) is very simple, which is responsible to instantiate class variSwap object and call class variSwap member function variSwap::varianceStrike() and variSwap::varianceStrikeBetter() to give fair variance strike. The corresponding volatility value($\sqrt{\text{variance strike}}$) is also given for ease of comparison with market volatility swaps.

The class variSwap is created for variance strike calculations. It has private data members, forwardPrice for SPX forward price on March 24 forwarded to April 21, spotPrice for SPX spot price on March 24, r continuously compounded annualized risk-free rate, T time to maturity, and two vectors of struct node to store strike and price pair. Private member function f() is the payoff of log contract at expiration. Public member function variSwap::varianceStrike() uses naive integration, while variSwap::varianceStrikeBetter() uses pair-wise approximation. These two return values should be very similar, and variSwap::varianceStrikeBetter() should give more reliable result.

A Makefile is supplemented for compilation. To compile the program, just type make to the terminal and press enter. If it compiles successfully, you should see an executable named variSwap. The suitable input parameters for this problem have been listed in input file already. To run the code, type ./variSwap < input to the terminal and press enter. The fair strike values are returned onto the screen.

B. Python implementation

This Python program was coded in sequential fashion. Overall flow of the program is straightforward. The code requires the same 5 input parameters as in the C++ implementation. The default values are preset, such that if you don't type in anything and press enter the default values are taken as input. The program then works as follows:

Spot price is calculated as the average of bid and ask on March 24.
Three python lists are then created strike, price_call, and price_put.
Read data from the data file into the appropriate lists.
Determine the $S_\star$ that separates OTM calls and puts. Select those OTM calls and puts.
Calculate the average price weighted by dK/K^2. This average price, together with a dynamic stock position, produces the fair value variance strike using naive integration method.

C. Result

The fair value variance strike by naive integration method is 0.0167324 or $(12.9354\%)^2$ for both C++ and Python.

The fair value variance strike by a better pair-wise approximation method gives 0.0161104 or $(12.6927\%)^2$ for C++. This result should be better.

II. QUESTION 2

Can you think of at least 3 reasons as to why the replication based algorithm you have above may be different from the real fair value strike level? Can you think of ways to account for these factors?

A. spacing between strikes, ΔK effect

In reality, the strike price is discrete not continuous, which contributes to an over-estimated fair variance strike σ_K^2. In Fig. 1, it shows that as ΔK gets larger, the errors are increasing. Generally, the pair-wise algorithm outperforms the naive integration method.

To account for this error, we could either try to find more precise(ΔK small) data, or when the improvement from data is not possible, always use pair-wise approximation algorithm.

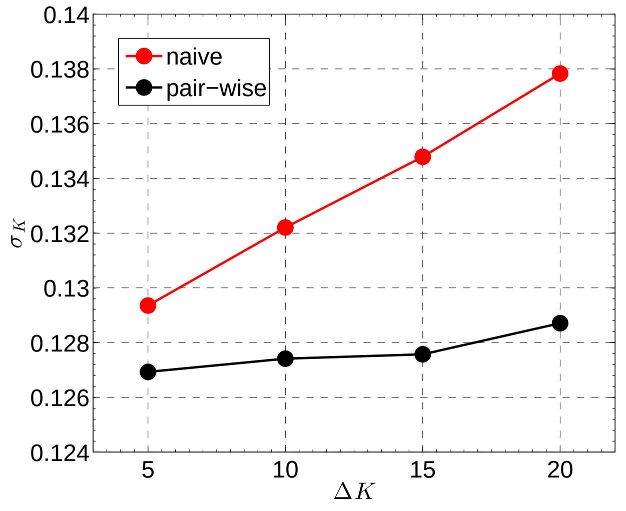

FIG. 1: The corresponding volatility strike σ_K vs. spacing between puts calls strikes ΔK. The red line means the naive integration scheme, while the black line represents a better pair-wise approximation method. Firstly, we can see that as ΔK gets smaller, both algorithms approach the true σ_K. Secondly, when spacing ΔK is large, the pair-wise approximation method outperforms the naive integration method.

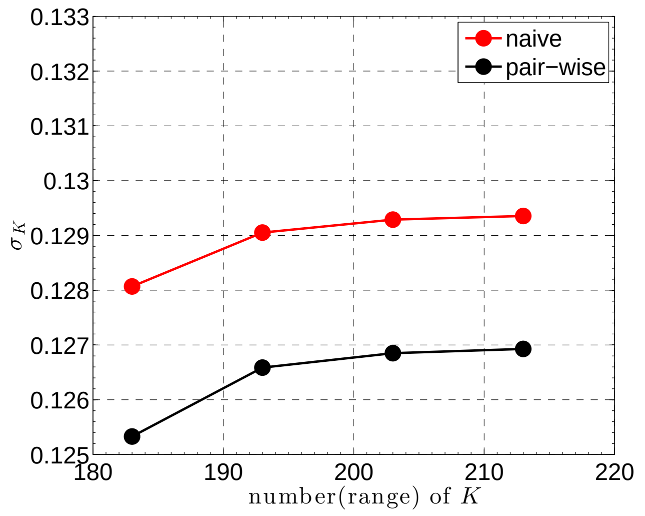

FIG. 2: The corresponding volatility strike σ_K vs. the number or range of strikes. The red line means the naive integration scheme, while the black line represents a better pair-wise approximation method. The full dataset contains 213 Ks. When we progressively remove some high K data, the strike σ_K is lowered. Both algorithms suffer from this defect to relatively the same degree.

B. available strikes, range of K effect

The range effect comes from the incompleteness of market, as not all prices for all possible strikes exist. When we miss some strikes, the model gives an under-estimated variance strike. As in Fig. 2, when removing more of our strike points, the model gives more appreciable errors, and the effect is always an under-estimated variance strike. The pair-wise method doesn't outperform the naive method in this case, because the competitive edge of pair-wise method usually comes from a better handling of integration, not in case of missing data.

To account for this errors, we could estimate the missing price first, for example by Black–Scholes models. Then integrate the model generated data with real market data to obtain a full picture of prices.

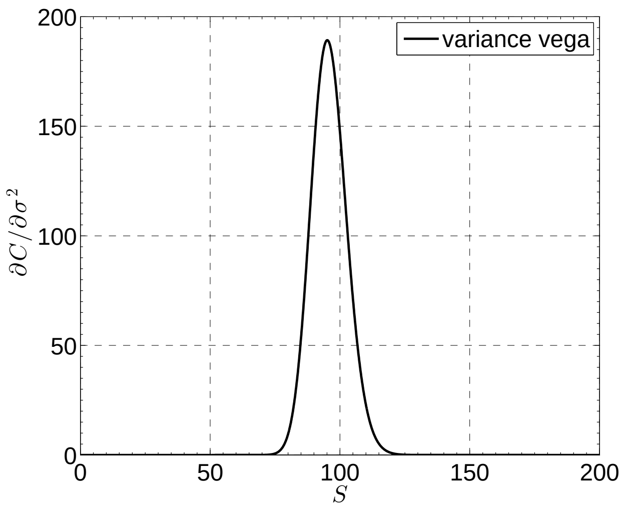

FIG. 3: Variance vega vs. spot price. Time to maturity $\tau = 1$ year, volatility $\sigma = 0.1$, strike price $K = 100$, risk-free rate $r = 0.05$. The sensitivity of option value to variance is maximized when spot price is close to strike. However, this sensitivity falls off quickly as spot price moves away from strike. It shows stock option can sometimes become a poor variance trading tool.

C. liquidity effect

In a liquid option market, the option value can be determined by the market prices. But when trading is not very active, the trading price may no longer reveal the true value of options. Since the mismatch between the true value and price can be positive or negative, the effect on variance strike is ambiguous.

To account for the liquidity problem, one can use option pricing theory like Black–Scholes model or value the illiquid option by replication just as what we did for the log contract.

III. QUESTION 3

Why do you think investors want to trade variance as opposed to vanilla options? Can you give an intuitive as well as a mathematical answer?

Because variance swap gives investors or traders *pure* exposure to variance. The sensitivity of vanilla option values to variance also depends on spot price. What's even worse, this sensitivity may vanish entirely when spot is distant away from strike.

The intuitive answer is that when the option is deep in the money, investors believe that this option will end up in the money eventually regardless of volatility, this effectively removes the volatility as a contributing factor. Thus the sensitivity of vanilla option value to variance is zero for deep in the money option. Same argument holds for deep out of the money option.

Calculate $\frac{\partial C}{\partial \sigma^2}$ to confirm indeed when S is distant away from K, this sensitivity equation is zero.

$$
\begin{aligned}
\frac{\partial C}{\partial \sigma^2} &= S\frac{\partial N(d_1)}{\partial \sigma^2} - Ke^{-r\tau}\frac{\partial N(d_2)}{\partial \sigma^2} \\
&= S\frac{\partial N(d_1)}{\partial d_1}\frac{\partial d_1}{\partial \sigma^2} - Ke^{-r\tau}\frac{\partial N(d_2)}{\partial d_2}\frac{\partial d_2}{\partial \sigma^2} \\
&= S\frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}\left(-\frac{1}{2}\frac{\ln(S/K)+r\tau}{\sqrt{\tau}}\sigma^{-3} + \frac{\sqrt{\tau}}{4}\sigma^{-1}\right) - Ke^{-r\tau}\frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}\frac{S}{K}e^{r\tau}\left(-\frac{1}{2}\frac{\ln(S/K)+r\tau}{\sqrt{\tau}}\sigma^{-3} - \frac{\sqrt{\tau}}{4}\sigma^{-1}\right) \\
&= S\frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}\left(\frac{\sqrt{\tau}}{2}\sigma^{-1}\right) \\
&= \frac{S\sqrt{\tau}}{2\sigma}\phi(d_1) \\
d_1 &= \frac{\ln\frac{S}{K} + \left(r+\frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}
\end{aligned}
$$

where $\phi(x)$ is standard normal random variable probability distribution function. We can plot $\partial C/\partial\sigma^2$ in Fig. 3. It confirms our intuitive idea that when the option is deep in the money, the option value is independent of variance anymore, thus making it a poor variance trading tool.

Even if we delta-hedge the vanilla option, thus remove the spot price dependence, it is still not as good as variance swap. The hedge is from Black–Scholes theory, and may not be very accurate in practice. Also delta-hedge requires constantly rebalancing, resulting in more transaction cost.

So variance swap is superior to vanilla options in variance trading.

[1] K. Demeterfi, E. Derman, M. Kamal, and J. Zou, *"More Than You Ever Wanted To Know About Volatility Swaps"* (1999)
[2] S. Bossu, *"Introduction to Variance Swaps"* (2005)
[3] Bossu, Strasser, and Guichard, *"Just What You Need To Know About Variance Swaps"* (2005)
[4] www.cboe.com
[5] http://www.global-rates.com/interest-rates/libor/american-dollar/usd-libor-interest-rate-12-months.aspx