NYU Tandon School of Engineering

Capstone Project

Special Topics in Asset Pricing: FRE-GY 9713

Yuhao Zheng
2022-11-3

Volatility Arbitrage Strategy with SABR

Yuhao Zheng
yz3328@nyu.edu

Introduction

In the course 9713, Asset Pricing, Professor Benveniste Ritter covered the topic of market microstructure, trading cost from price impact, and how dynamic programming can be applied in positions management. An order book snapshot data can provide more information deep into the microstructure of the market, including the bid-ask spread, the market depth, and some hidden orders that can be told from the bid/ask size. With this high-frequency data, we can have a better prediction for the asset price's direction in the next period, not to mention executing orders in a more precise way to reduce trading cost by avoiding price impact. So, in this paper, we are going to research and develop an option trading strategy with the dynamic programming technique as a method for risk management in everyday positions adjustment. Besides, two different backtest system structures would be provided and compared from the aspects of efficiency, accuracy, and applicability for high-frequency trading strategies.

The type of strategies we can have for option trading is various, including momentum, market making and volatility arbitrage. However, the data we have is a six-year end-of-day historical price of a Chinese ETF, which makes it hard to generate an efficient prediction for the price of tomorrow without using any other intraday or fundamental data. So, we left alone the momentum strategy. About market making strategy, that was supposed to be a profitable strategy, especially when high-frequency data is available. But for the existing backtest system, a limit order is hard to achieve since we assume to execute an order with the mean price or the best bid/ask price in the next trading period. That would eat up almost all the profit we could earn from the bid-ask spread. So, in this case, volatility arbitrage becomes our last choice. In this paper, we will provide you with a simple example about how to achieve volatility statistical arbitrage between the theoretical value and its market value based on the SABR model.

Content

Strategy

The basic logic behind the volatility arbitrage is to trade the mispriced volatility compared to the real volatility with option and wait until the market pushes the price back to its normal region. However, the real volatility cannot be directly observed from the market. So, there are different ways to define volatility, including historical volatility (HV), realized volatility (RV) and implied volatility (IV). The simplest arbitrage strategy is built upon the difference between HV and IV due to the accessibility of HV, which can be easily described by the standard deviation of the historical return data. But HV is lower than IV in most of the time according to the statistical result, not to mention that HV lacks predicting ability in the future. So, most traders are trying to use machine learning algorithms in predicting RV for a much more profitable arbitrage opportunity in a long term. That is full of challenges in both the variety of data and complexity of models.

The statistical arbitrage based on the difference between the theoretical value of IV and its market value we are going to conduct is different from the former two ideas talked about. With the assumption of stochastic volatility (SV) model, we are going to use the SABR model to best fit the IV surface and generate the SV surface as the theoretical value of IV. The outlier of IV compared to the SV surface can be caused by the liquidity or some other mispricing reasons, which can be eliminated by arbitrage. In a word, our volatility arbitrage trading strategy is to short the most overpriced option contract, whose IV is significantly higher than SV, and long the underpriced option to achieve delta hedge. More details about how we pick up these two mispriced contracts every day will be discussed in the following paper.

Data

50ETF is the first exchange-traded fund in mainland China, whose tracking index is SSE 50, and its options are the most actively traded assets in the Chinese derivative market. China 10-year bond yield would be used as the risk-free rate, and the time period for all datasets is from 01/04/2016 to 03/04/2022.

“opt.csv” stores all the available data with a composite primary key (Symbol, index) to locate the market information for every tradable option contract in every single day. We need to preprocess the data to generate some necessary derivative variables for the following trading signal generation and positions management in our strategy. Besides, some functions for calculating option greeks and implied volatility are stored in class Greek in file “calculator.py”. Below is the description for each function:

Function	**Return**	**Description**
get_atm()	at-the-money option’s volatility	Sort by abs(“UnlyPrice” – “StrikePrice”) to get the at-the-money option contract and then calculate its implied volatility
Greek.bsm()	option’s price	Calculate option’s price with the Black-Scholes-Merton model
Greek.delta()	option’s delta	Calculate option’s first-order derivative of BSM price to underlying asset’s price
Greek.vega()	option’s vega	Calculate option’s first-order derivative of BSM price to implied volatility
Greek.vix()	option’s implied volatility	Calculate option’s implied volatility with rational approximation[1]

In an event-driven backtest system, we should generate the data we need for signal generation at the end of each trading period, like every day in this case. So, we do not need, and in most cases are even not allowed, to generate the derivative variables for the whole period ahead of backtest in case of using future data. However, in our data processing steps, every derivative variable for each single day is generated by exactly only using the data available for that specific moment, so there is no future data in our derivative variables. The reason why we want to generate derivative variables for the whole period ahead is that we want to take the benefit of vectorization to save time from calculating in our process of signal generation and backtest, which are conducted in a for loop. An event-driven backtest system would be provided at the end of this paper as a more applicable system for high-frequency trading strategies.

Model

The SABR model of Hagan et al.[2] is described by the following 3 equations:

$$df_t = \alpha_t f_t^{\beta} dW_t^1$$
$$d\alpha_t = v\alpha_t dW_t^2$$
$$\mathrm{E}[dW_t^1 dW_t^2] = \rho dt$$

In these equations, $f_t = S_0 e^{\int_0^t r(s)ds} \approx S_0 e^{r_f t}$ is the forward rate, α_t is the volatility. There are 4 parameters need estimating to calculate the theoretical value for implied volatility surface. Below shows the necessary parameters and their estimating method:

Parameter	**Definition**	**Estimating Method**
β (Beta)	The exponent for f	1. = 1 (S~Geometric Brownian Motion) 2. Use the parameter of the rolling linear regression, $ln\sigma \approx ln\alpha - (1-\beta)lnf$, for ATM options 3. See "Hedge"
α (Alpha)	The initial volatility of S	$ln\sigma \approx ln\alpha - (1-\beta)lnf$ for ATM options
v (Vega)	The volatility of α_t	1. Optimize with minimum mean squared error between SV and IV 2. See "Hedge"
ρ (Rho)	The correlation between two Brownian motions	

Our target is to develop a stochastic volatility model for best fitting the implied volatility surface, and Vega and Rho are two parameters needed to estimate every day in our SABR model. So, we use the same filtering conditions in function get_signal() excludes the delta limitation, see chapter "Signal" for more details, to get all the available options and their implied volatility and set the mean squared error of IV and SV estimated by the SABR model as an optimize function to better our choice of Vega and Rho. This parameter estimating process would be conducted at the end of every day with all the available data within that day, which means that the theoretical SV surface is changing every day. Function Greek.sabr() helps us achieve this process by using the equation below:

$$
\begin{aligned}
\sigma_B(K,f) &= \frac{\alpha\left\{1+\left[\frac{(1-\beta)^2}{24}\frac{\alpha^2}{(fK)^{1-\beta}}+\frac{1}{4}\frac{\rho\beta v\alpha}{(FK)^{(1-\beta)/2}}+\frac{2-3\rho^2}{24}v^2\right]T\right\}}{(fK)^{(1-\beta)/2}\left[1+\frac{(1-\beta)^2}{24}\ln^2\frac{f}{K}+\frac{(1-\beta)^4}{1920}\ln^4\frac{f}{K}\right]} \\
&\quad \times\frac{z}{\chi(z)} \\
z &= \frac{v}{\alpha}(fK)^{(1-\beta)/2}\ln\frac{f}{K} \\
\chi(z) &= \ln\left[\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}\right].
\end{aligned}
$$

Hedge

In 2016, Mengfei Zhang[3] provided another parameters estimation idea for the SABR model by taking the risk hedging into consideration. Instead of calculating the mean squared error between the IV and SV to describe how perfectly the SABR model fits the real IV surface, Mengfei proposed that the SABR model should be used in managing the risk instead of predicting the volatility. So, he creatively used his self-designed indicator, the hedging error (HE), as the loss function to optimize the parameters.

$$
\begin{aligned}
&SSRHE_t = \sum_k w_{k,t}HE_{k,t}^2 \\
&HE_{k,t} = \frac{\hat{C}_{k,t}}{C_{k,t}} - 1, w_{k,t} = \frac{1}{1+(f-k)^2} \\
&\hat{C}_{k,t} = C_{k,t-1} + \Delta_{k,t-1}df_t + \phi_{k,t-1}d\alpha_t \\
&\Delta = \frac{\partial C}{\partial f}, \phi = \frac{\partial C}{\partial \alpha}
\end{aligned}
$$

Mengfei believed that if the market forward price f and volatility dynamics α were assumed to be following the SABR model, with the correct parameters, the hedging error should be equal to 0. That is the reason why we can use the hedging error as a loss function to optimize the parameters. Hagan et al.[2] provided the equations for calculating option greeks of the SABR model in 2002. Besides, w_k is the weight for the option contract strikes at k, who gives the at-the-money contract the highest weight.

Since we estimate the parameters at the end of every single day, the theoretical SV surface and option greeks are calculated by the SABR model every day as well so as to help us achieve dynamic hedging.

Signal

The trading signal is generated every day using all the available data by the end of that specific moment. So, we group the whole dataset by "index" to get all the tradable option contracts and their market information for every single day and apply the function get_signal() to generate the trading signal for the next trading day. In function get_signal(), we need to apply some filtering conditions to help us better the choice of our arbitrage option pair. Below is the description of each filter:

Filter	**Condition**	**Description**
Implied Volatility	not nan value	IV for deep OTM option may be inaccessible, need filtering out
Maturity	> 5 calendar days	Liquidity and other unknown factors may significantly fluctuate the option price within a very close maturity days, need rolling
Delta	0.1 ~ 0.9	Extreme delta value makes hedging difficult, need avoiding

After that, all the available option contracts have accessible implied volatility for factor value calculation, and there are not too many unknown risk factors that might significantly fluctuate the option price, among which we can conduct our arbitrage pair selection. The selecting logic is shown as below:

Factor	**Condition**	**Description**
Maturity	with the least maturity days	SABR model is more accurate for a short-term volatility, so we only trade in the closest-mature option

Arbitrage Opportunity	with the larger arbitrage space $max(IV - SV)$	For overpriced options, whose IV is higher than its theoretical value, we pick up the one with the highest overestimated value.

The trading signal is generated mainly by comparing the arbitrage opportunity (AO) of the current option pair and that of the new option pair. If we define

$$AO = (IV_{call} - SV_{call}) + (IV_{put} - SV_{put}),$$

we can compare the AO of the current option pair with a given arbitrage threshold to determine whether to cover this current position or not. Same for opening a new position, if the AO of the new option pair is larger than both the threshold and the AO of the current option position, we should open a new position and cover the current one if it exists. Specially, since we want to achieve delta hedge for every single trade, we should always short the most overpriced call option contract and a corresponding volume of a put option according to their delta ratio. More details about the signal generating process are shown as below:

Trading Action	**Condition**	**Description**
Cover Position	1. $AO_{curr} < Threshold$ 2. Or $AO_{curr} < AO_{new}$ 3. Or $Maturity < 3$	Cover the currently trading option when there is no arbitrage space any more
Open Position	1. $AO_{new} > Threshold$ 2. And $AO_{new} > AO_{curr}$	Update a new option pair with a larger arbitrage space

Position

After we get the trading signal for every single day, it is time for us to determine how many contracts we should trade for each time. Function get_position() takes current available capital, delta and some other market information for the new option pair generated from function get_signal() as inputs and calculates the maximum volume we

can have for trading in the next trading day. The position we are going to have in the next trading day must meet the following requirements:

Variable	Requirement	Description
Rest Capital	> 0	After executing a new position, the money left must be larger than 0
Delta	= 0	The total delta value for this option pair must be 0 to achieve hedging
Order Volume	< market size	The maximum volume could be executed is the best bid/ask size

Backtest

Currently, we have all the functions we need to get the target option contracts and their corresponding position by the end of every trading day. What we need now is a backtest function with a big for loop to execute these trading orders day by day and save all necessary information including daily profit and capital left for further analysis. There are two different way to conduct the backtest process.

First, as we have mentioned in chapter “Data”, we can calculate everything we need including signals, positions and the everyday trading orders separately beforehand to take the advantage of vectorization to save more time. In this case, the backtest system would only need to finish the last step, to conduct a for loop for calculating daily profit.

However, separately getting positions and executing trading orders make the first backtest system impossible to dynamically get the capital left and the risk exposure for current positions. That is the reason why we need the second backtest system, who strictly follows the rules of the event-driven system to avoid using future data. In this case, the backtest system conducts a huge for loop, and for each iteration, all the derivative data, trading signals, and target positions would be calculated one by one for exactly one day with all the available data by the end of that day. Of course, daily profit would be calculated on the next day morning, and capital left would be calculated after the new order is executed by the end of the day.

Backtest 1	**Backtest 2 (Event-Driven)**
signal = data.**apply**(get_signal) position = signal.**apply**(get_position) for i from 1 to n: profit = get_profit(position[i]) capital = get_capital(position[i])	for i from 1 to n: profit = get_profit(position) signal = get_signal(data[i]) position = get_position(signal, **risk,**) capital = get_capital(position)
Pros: . - apply() use vectorization to faster the speed in big data calculation **Cons**: - Hard for dynamic risk hedging and position management - Easy to mistakenly use future data	**Pros**: - Avoid using future data - Flexible for dynamic programming **Cons**: - Time consuming - One single for loop is hard for error backtracking and troubleshooting

Optimization

The SABR model is validated to be more efficient for the short-term IV surface simulation. That is the reason why we only pick up the option contracts with the shortest maturity day larger than 5 trading days (T). Besides, the arbitrage opportunity (AO) determines the upper limit of the profit we can earn from that trading, so the threshold for opening the position plays a key role in the final performance of the whole strategy. Last but not least, we use a range (R) to help us filter out some contracts with an extreme delta value to avoid unknown risk factors. All these three parameters can be optimized according to the in-sample backtest performance.

Grid search is an algorithm can help us find the best parameters. Even though it is time consuming since it will calculate the result for all possible parameter combinations, in this case, we only have $5 \times 3 + 9 = 24$ combinations, which is acceptable.

Parameter	Definition	Range
T	the shortest maturity day	[5, 10, 21, 42, 63]
AO	threshold for opening positions	range(0, 0.9, 0.1)
R	delta limitation	[(0.1, 0.9), (0.2, 0.8), (0.3, 0.7)]

Both T and R play as a filter for selecting available option contracts for trading at that day, so it would be better to apply grid search for optimizing them at the same time. For AO, it can be optimized once we pick up the best combination for T and R, in which case we can save more time in grid searching.

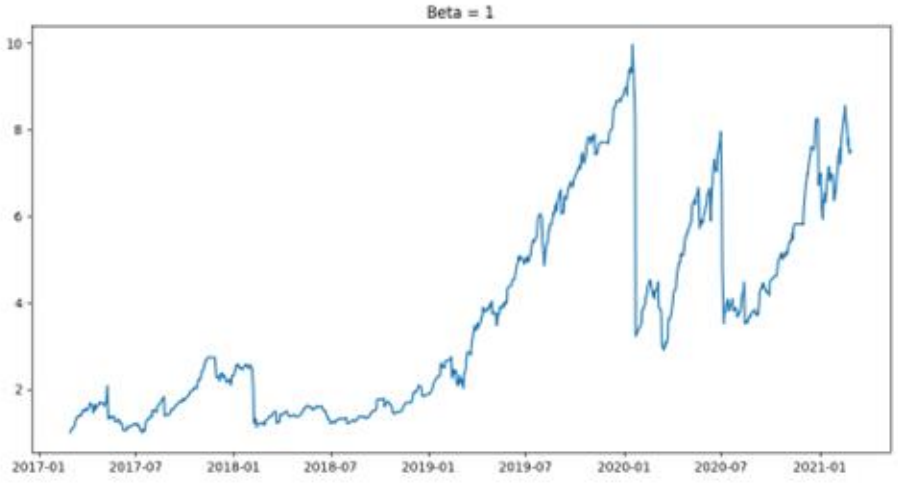
Performance

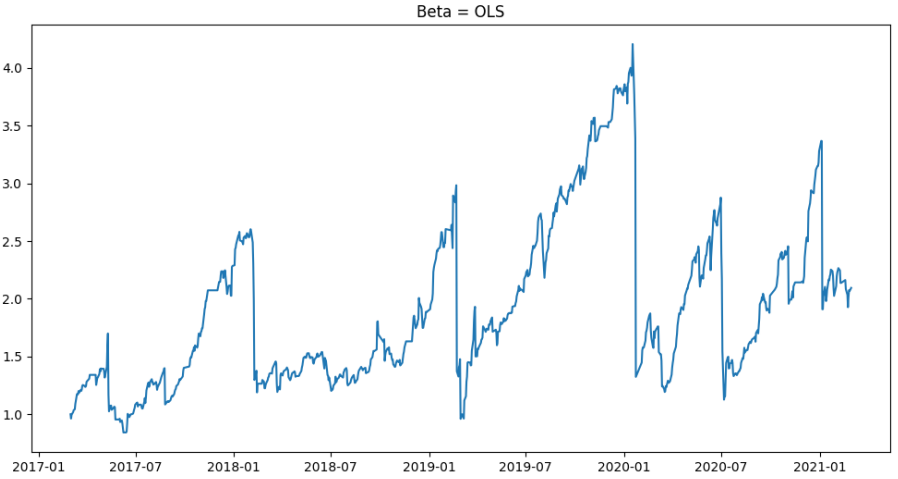
In chapter "Model", there are three different ways in estimating the parameter β for the SABR model, which would determine the stochastic process of the forward price. So, we separately optimize the strategy for each different selection of the parameter β.

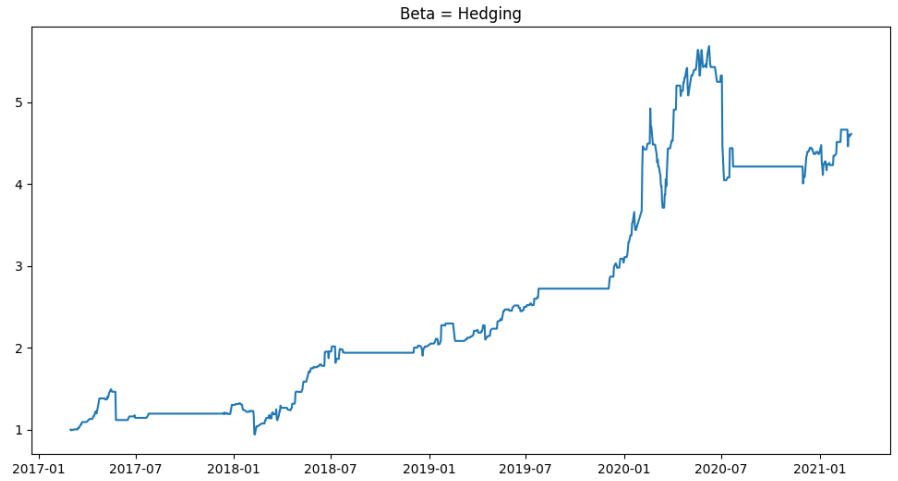
In-Sample

Time Period: 2017-03-01 ~ 2021-03-01

Beta	Annual Return	Annual Std	Sharpe	Max Drawdown
1	65.54%	71.11%	0.92	705.48%
OLS	20.29%	80.09%	0.25	308.01%
Hedge	46.51%	32.44%	1.43	167.98%


Beta = 1
10
8
6
4
2
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01

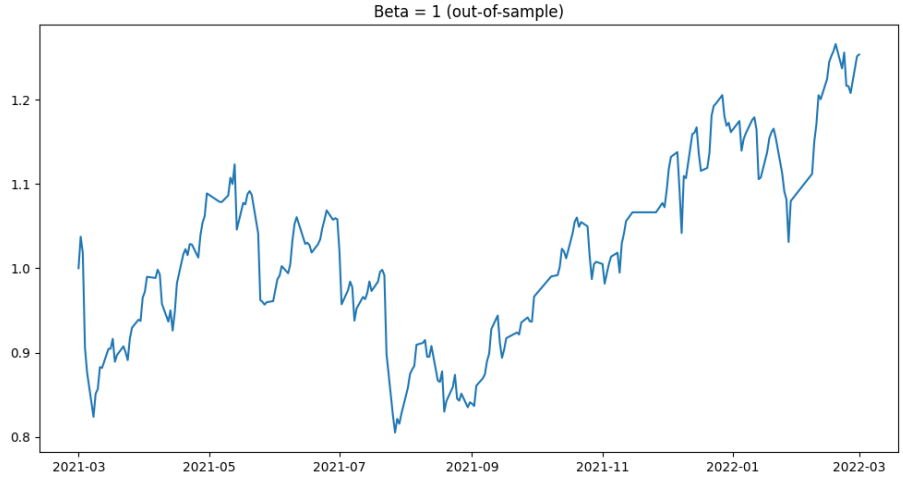

Beta = OLS
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01

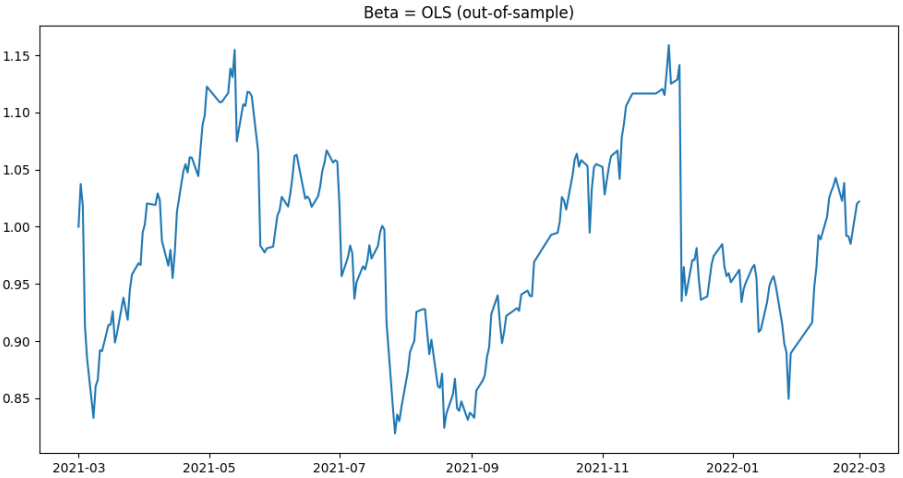

Beta = Hedging
5
4
3
2
1
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
2021-01

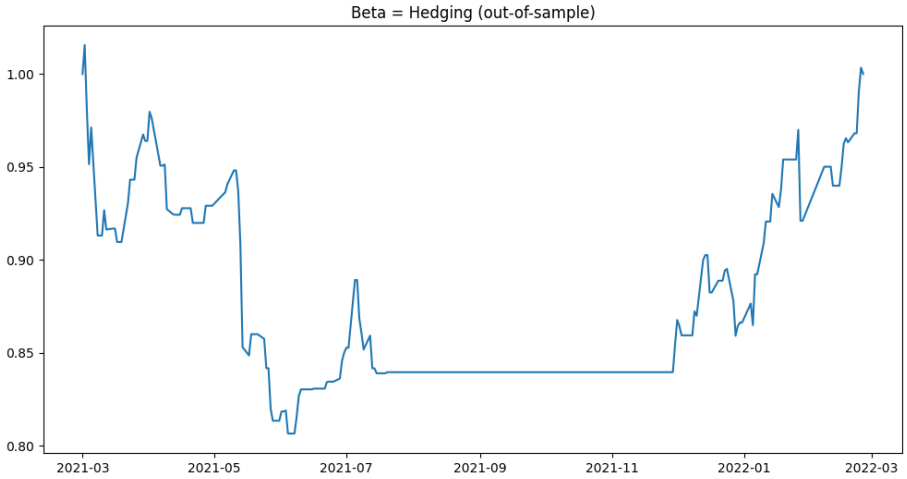
Out-of-Sample

Time Period: 2021-03-01 ~ 2022-03-01

Beta	Annual Return	Annual Std	Sharpe	Max Drawdown
1	25.36%	37.35%	0.68	31.82%
OLS	2.20%	40.66%	0.05	33.55%
Hedge	0.00%	17.94%	0.00	20.90%







Conclusion

Obviously, by assuming the parameter β to be 1, our strategy gets the best performance in both the in-sample and the out-of-sample backtest. This is reasonable since the SABR model assumes that the forward price is following a Geometric Brownian Motion, which is following a popular stochastic process assumption for the asset price. In this case, there is no doubt that the second strategy, whose β is estimated by a rolling linear regression, has the worst performance. The linear relationship may not be always held for the at-the-money option's implied volatility and the forward price, not to mention that the outlier of the rolling samples can also vibrate the regression result. The last strategy with a risk-hedging β has the best Sharpe ratio, which shows that a risk-hedging oriented parameter estimation method does improve the stability of the strategy.

REFERENCES

[1] Li, Minqiang. “You Don't Have to Bother Newton for Implied Volatility”, November 2006. Available at SSRN: https://ssrn.com/abstract=952727

[2] Hagan, P., Kumar, D., Lesniewski, L, and D.E. Woodward (2002). “Managing Smile Risk”, Wilmott Magazine, September 2002, pp. 84-108

[3] Zhang, Mengfei. “On the Estimation of the SABR Model’s Beta Parameter: The Role of Hedging in Determining the Beta Parameter”, The Journal of Derivatives, August 2016