SMU
SINGAPORE MANAGEMENT UNIVERSITY

Lee Kong Chian School of Business

QF620 - Stochastic Modelling in Finance

Project Report

Prepared By:

WANG LI

SEAH YAN DE BRYAN

MINTON YAP LI HUI

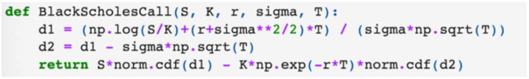
NICHOLAS WONG

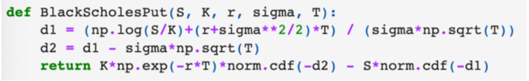
Part 1 (Analytical Option Formulae):

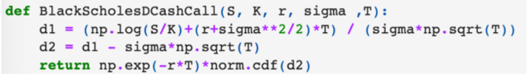
1. This section will do a side-by-side comparison between the various valuation formulae and its implementation in Python.

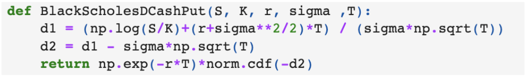
1.1 Black Scholes Model

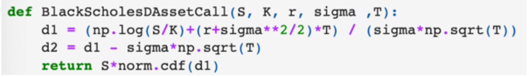
Option	Pricing Formula	Python Code

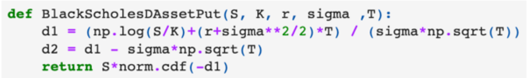
Vanilla Call	$S_0\Phi\left(\frac{\log\left(\frac{S_0}{K}\right)+\left(r+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)-Ke^{-rT}\Phi\left(\frac{\log\left(\frac{S_0}{K}\right)+\left(r-\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)$	
		`def BlackScholesCall(S, K, r, sigma, T):`
		`    d1 = (np.log(S/K)+(r+sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return S*norm.cdf(d1) - K*np.exp(-r*T)*norm.cdf(d2)`

Vanilla Put	$Ke^{-rT}\Phi\left(\frac{\log\left(\frac{K}{S_0}\right)-\left(r-\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)-S_0\Phi\left(\frac{\log\left(\frac{K}{S_0}\right)-\left(r+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)$	
		`def BlackScholesPut(S, K, r, sigma, T):`
		`    d1 = (np.log(S/K)+(r+sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return K*np.exp(-r*T)*norm.cdf(-d2) - S*norm.cdf(-d1)`

Digital
cash-or-nothing Call
	$e^{-rT}\Phi\left(\frac{\log\left(\frac{S_0}{K}\right)+\left(r-\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)$	
		`def BlackScholesDCashCall(S, K, r, sigma ,T):`
		`    d1 = (np.log(S/K)+(r+sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*norm.cdf(d2)`

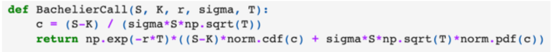
Digital
cash-or-nothing Put
	$e^{-rT}\Phi\left(\frac{\log\left(\frac{K}{S_0}\right)-\left(r-\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)$	
		`def BlackScholesDCashPut(S, K, r, sigma ,T):`
		`    d1 = (np.log(S/K)+(r+sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*norm.cdf(-d2)`

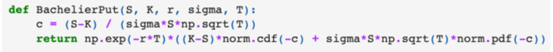
Digital
Asset-or-nothing Call
	$S_0\Phi\left(\frac{\log\left(\frac{S_0}{K}\right)+\left(r+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)$	
		`def BlackScholesDAssetCall(S, K, r, sigma ,T):`
		`    d1 = (np.log(S/K)+(r+sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return S*norm.cdf(d1)`

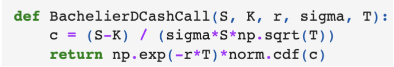
Digital
Asset-or-nothing Put
	$S_0\Phi\left(\frac{\log\left(\frac{K}{S_0}\right)-\left(r+\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)$	
		`def BlackScholesDAssetPut(S, K, r, sigma ,T):`
		`    d1 = (np.log(S/K)+(r+sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return S*norm.cdf(-d1)`

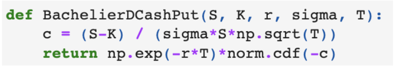
1.2 Bachelier Model

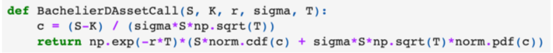
Option	Pricing Formula	Python Code

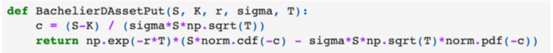
Vanilla Call	$(S_0-K)\Phi\left(\frac{S_0-K}{S_0\sigma\sqrt{T}}\right)+S_0\sigma\sqrt{T}\phi\left(\frac{S_0-K}{S_0\sigma\sqrt{T}}\right)$	
		`def BachelierCall(S, K, r, sigma, T):`
		`    c = (S-K) / (sigma*S*np.sqrt(T))`
		`    return np.exp(-r*T)*((S-K)*norm.cdf(c) + sigma*S*np.sqrt(T)*norm.pdf(c))`

Vanilla Put	$(K-S_0)\Phi\left(\frac{K-S_0}{S_0\sigma\sqrt{T}}\right)+S_0\sigma\sqrt{T}\phi\left(\frac{K-S_0}{S_0\sigma\sqrt{T}}\right)$	
		`def BachelierPut(S, K, r, sigma, T):`
		`    c = (S-K) / (sigma*S*np.sqrt(T))`
		`    return np.exp(-r*T)*((K-S)*norm.cdf(-c) + sigma*S*np.sqrt(T)*norm.pdf(-c))`

Digital
cash-or-nothing Call
	$\Phi\left(\frac{S_0-K}{S_0\sigma\sqrt{T}}\right)$	
		`def BachelierDCashCall(S, K, r, sigma, T):`
		`    c = (S-K) / (sigma*S*np.sqrt(T))`
		`    return np.exp(-r*T)*norm.cdf(c)`

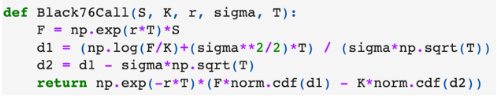
Digital
cash-or-nothing Put
	$\Phi\left(\frac{K-S_0}{S_0\sigma\sqrt{T}}\right)$	
		`def BachelierDCashPut(S, K, r, sigma, T):`
		`    c = (S-K) / (sigma*S*np.sqrt(T))`
		`    return np.exp(-r*T)*norm.cdf(-c)`

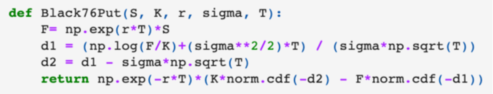
Digital
Asset-or-nothing Call
	$S_0\left[\Phi\left(\frac{S_0-K}{S_0\sigma\sqrt{T}}\right)+\sigma\sqrt{T}\phi\left(\frac{S_0-K}{S_0\sigma\sqrt{T}}\right)\right]$	
		`def BachelierDAssetCall(S, K, r, sigma, T):`
		`    c = (S-K) / (sigma*S*np.sqrt(T))`
		`    return np.exp(-r*T)*(S*norm.cdf(c) + sigma*S*np.sqrt(T)*norm.pdf(c))`

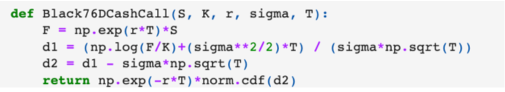
Digital
Asset-or-nothing Put
	$S_0\left[\Phi\left(\frac{K-S_0}{S_0\sigma\sqrt{T}}\right)-\sigma\sqrt{T}\phi\left(\frac{K-S_0}{S_0\sigma\sqrt{T}}\right)\right]$	
		`def BachelierDAssetPut(S, K, r, sigma, T):`
		`    c = (S-K) / (sigma*S*np.sqrt(T))`
		`    return np.exp(-r*T)*(S*norm.cdf(-c) - sigma*S*np.sqrt(T)*norm.pdf(-c))`

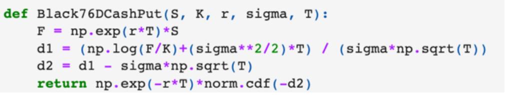
1.3 Black 76 Model

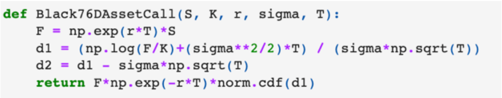
Option	Pricing Formula	Python Code

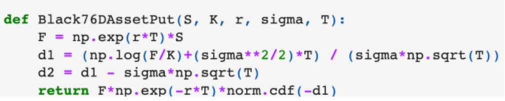
Call	$e^{-rT}\left[F_0\Phi\left(\frac{\log\left(\frac{F_0}{K}\right)+\frac{\sigma^2T}{2}}{\sigma\sqrt{T}}\right)-K\Phi\left(\frac{\log\left(\frac{F_0}{K}\right)-\frac{\sigma^2T}{2}}{\sigma\sqrt{T}}\right)\right]$	
		`def Black76Call(S, K, r, sigma, T):`
		`    F = np.exp(r*T)*S`
		`    d1 = (np.log(F/K)+(sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*(F*norm.cdf(d1) - K*norm.cdf(d2))`

Vanilla Put	$e^{-rT}\left[K\Phi\left(\frac{\log\left(\frac{K}{F_0}\right)+\frac{\sigma^2T}{2}}{\sigma\sqrt{T}}\right)-F_0\Phi\left(\frac{\log\left(\frac{K}{F_0}\right)-\frac{\sigma^2T}{2}}{\sigma\sqrt{T}}\right)\right]$	
		`def Black76Put(S, K, r, sigma, T):`
		`    F= np.exp(r*T)*S`
		`    d1 = (np.log(F/K)+(sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*(K*norm.cdf(-d2) - F*norm.cdf(-d1))`

Digital
cash-or-nothing Call
	$e^{-rT}\Phi\left(\frac{\log\left(\frac{F_0}{K}\right)-\frac{\sigma^2}{2}T}{\sigma\sqrt{T}}\right)$	
		`def Black76DCashCall(S, K, r, sigma, T):`
		`    F = np.exp(r*T)*S`
		`    d1 = (np.log(F/K)+(sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*norm.cdf(d2)`

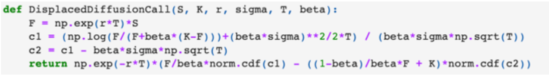
Digital
cash-or-nothing Put
	$e^{-rT}\Phi\left(\frac{\log\left(\frac{K}{F_0}\right)+\frac{\sigma^2}{2}T}{\sigma\sqrt{T}}\right)$	
		`def Black76DCashPut(S, K, r, sigma, T):`
		`    F = np.exp(r*T)*S`
		`    d1 = (np.log(F/K)+(sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*norm.cdf(-d2)`

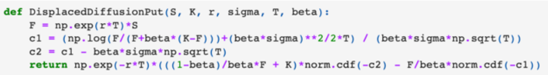
Digital
Asset-or-nothing Call
	$e^{-rT}F_0\Phi\left(\frac{\log\left(\frac{F_0}{K}\right)-\frac{\sigma^2}{2}T}{\sigma\sqrt{T}}\right)$	
		`def Black76DAssetCall(S, K, r, sigma, T):`
		`    F = np.exp(r*T)*S`
		`    d1 = (np.log(F/K)+(sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return F*np.exp(-r*T)*norm.cdf(d1)`

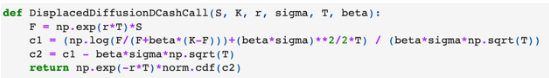
Digital
Asset-or-nothing Put
	$e^{-rT}F_0\Phi\left(\frac{\log\left(\frac{K}{F_0}\right)-\frac{\sigma^2}{2}T}{\sigma\sqrt{T}}\right)$	
		`def Black76DAssetPut(S, K, r, sigma, T):`
		`    F = np.exp(r*T)*S`
		`    d1 = (np.log(F/K)+(sigma**2/2)*T) / (sigma*np.sqrt(T))`
		`    d2 = d1 - sigma*np.sqrt(T)`
		`    return F*np.exp(-r*T)*norm.cdf(-d1)`

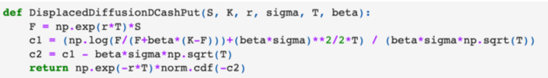
1.4 Displaced Diffusion Model

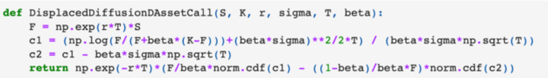
Option	Pricing Formula	Python Code

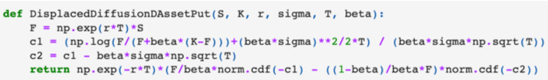
Call	$e^{-rT}\left[\frac{F_0}{\beta}\Phi\left(\frac{\log\left(\frac{F_0/\beta}{K+F_0((1-\beta)/\beta)}\right)+\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)-\left(K+F_0\frac{1-\beta}{\beta}\right)\Phi\left(\frac{\log\left(\frac{F_0/\beta}{K+F_0((1-\beta)/\beta)}-\frac{(\beta\sigma)^2T}{2}\right)}{\beta\sigma\sqrt{T}}\right)\right]$	
		`def DisplacedDiffusionCall(S, K, r, sigma, T, beta):`
		`    F = np.exp(r*T)*S`
		`    c1 = (np.log(F/(F+beta*(K-F)))+(beta*sigma)**2/2*T) / (beta*sigma*np.sqrt(T))`
		`    c2 = c1 - beta*sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*(F/beta*norm.cdf(c1) - ((1-beta)/beta*F + K)*norm.cdf(c2))`

Vanilla Put	$e^{-rT}\left[\left(K+F_0\frac{1-\beta}{\beta}\right)\Phi\left(\frac{\log\left(\frac{K+F_0((1-\beta)/\beta)}{F_0/\beta}\right)+\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)-\frac{F_0}{\beta}\Phi\left(\frac{\log\left(\frac{K+F_0((1-\beta)/\beta)}{F_0/\beta}\right)-\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)\right]$	
		`def DisplacedDiffusionPut(S, K, r, sigma, T, beta):`
		`    F = np.exp(r*T)*S`
		`    c1 = (np.log(F/(F+beta*(K-F)))+(beta*sigma)**2/2*T) / (beta*sigma*np.sqrt(T))`
		`    c2 = c1 - beta*sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*(((1-beta)/beta*F + K)*norm.cdf(-c2) - F/beta*norm.cdf(-c1))`

Digital
cash-or-nothing Call
	$e^{-rT}\Phi\left(\frac{\log\left(\frac{F_0/\beta}{K+F_0((1-\beta)/\beta)}\right)-\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)$	
		`def DisplacedDiffusionDCashCall(S, K, r, sigma, T, beta):`
		`    F = np.exp(r*T)*S`
		`    c1 = (np.log(F/(F+beta*(K-F)))+(beta*sigma)**2/2*T) / (beta*sigma*np.sqrt(T))`
		`    c2 = c1 - beta*sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*norm.cdf(c2)`

Digital
cash-or-nothing Put
	$e^{-rT}\Phi\left(\frac{\log\left(\frac{K+F_0((1-\beta)/\beta)}{F_0/\beta}\right)+\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)$	
		`def DisplacedDiffusionDCashPut(S, K, r, sigma, T, beta):`
		`    F = np.exp(r*T)*S`
		`    c1 = (np.log(F/(F+beta*(K-F)))+(beta*sigma)**2/2*T) / (beta*sigma*np.sqrt(T))`
		`    c2 = c1 - beta*sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*norm.cdf(-c2)`

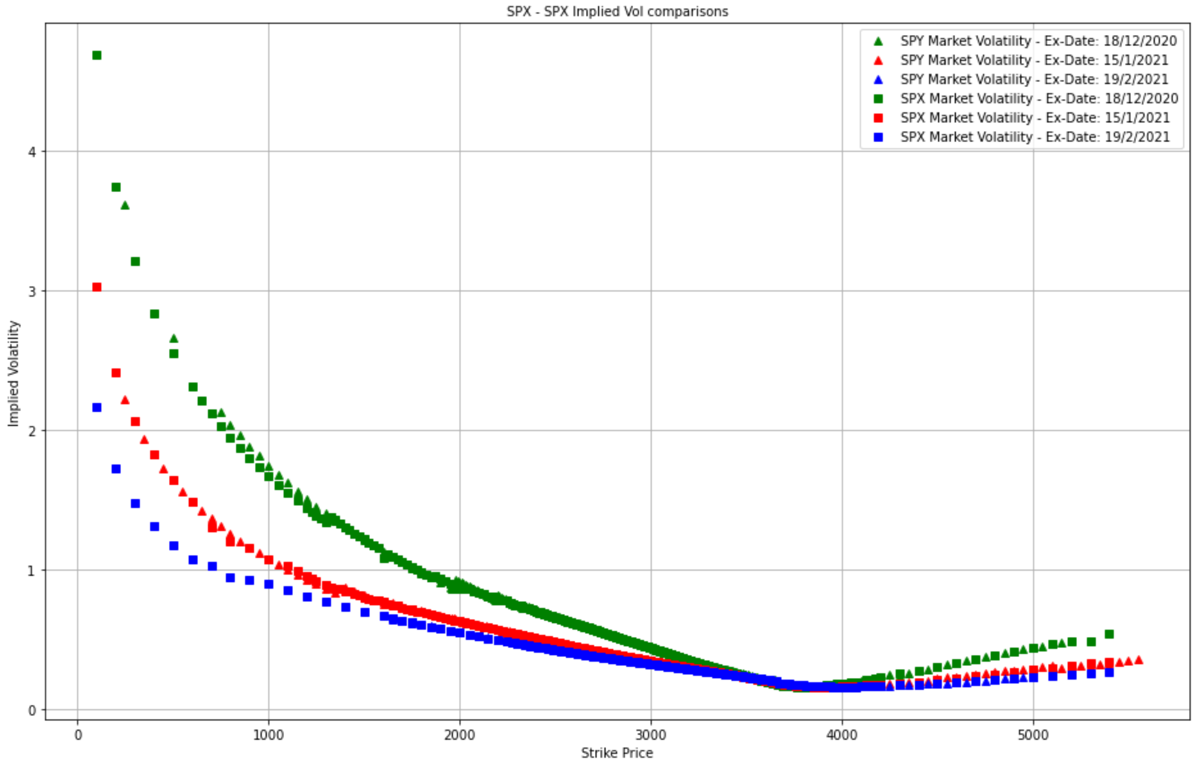
Digital
Asset-or-nothing Call
	$e^{-rT}\frac{F_0}{\beta}\Phi\left(\frac{\log\left(\frac{F_0/\beta}{K+F_0((1-\beta)/\beta)}\right)+\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)$	
		`def DisplacedDiffusionDAssetCall(S, K, r, sigma, T, beta):`
		`    F = np.exp(r*T)*S`
		`    c1 = (np.log(F/(F+beta*(K-F)))+(beta*sigma)**2/2*T) / (beta*sigma*np.sqrt(T))`
		`    c2 = c1 - beta*sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*(F/beta*norm.cdf(c1) - ((1-beta)/beta*F)*norm.cdf(c2))`

Digital
Asset-or-nothing Put
	$e^{-rT}\frac{F_0}{\beta}\Phi\left(\frac{\log\left(\frac{K+F_0((1-\beta)/\beta)}{F_0/\beta}\right)-\frac{(\beta\sigma)^2T}{2}}{\beta\sigma\sqrt{T}}\right)$	
		`def DisplacedDiffusionDAssetPut(S, K, r, sigma, T, beta):`
		`    F = np.exp(r*T)*S`
		`    c1 = (np.log(F/(F+beta*(K-F)))+(beta*sigma)**2/2*T) / (beta*sigma*np.sqrt(T))`
		`    c2 = c1 - beta*sigma*np.sqrt(T)`
		`    return np.exp(-r*T)*(F/beta*norm.cdf(-c1) - ((1-beta)/beta*F)*norm.cdf(-c2))`

Part 2 (Model Calibration):

2.1 Market Implied Volatility (Black Scholes Model): SPX - SPY Comparison

2. The implied volatilities for the various strikes were calculated using the Black-Scholes Pricing Model. A comparison of the implied volatilities from the SPX data and SPY data can be seen in the figure below.

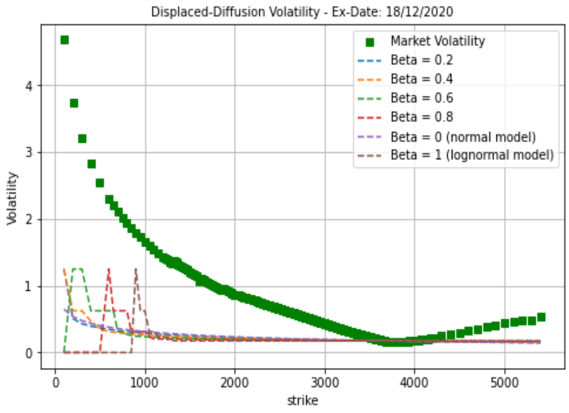


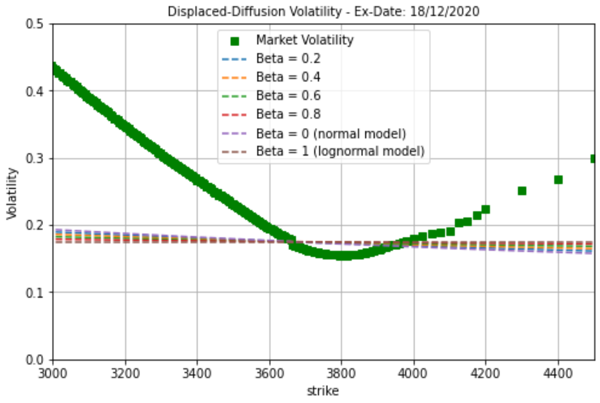
3. Visually, it appears that the implied volatilities are very similar between the SPX and SPY options - in particular around the spot price. We note that the pricing diverges slightly as it moves away from the spot price (3662.45) and further out of the money. Due to their similarities, we will subsequently be using the SPX data in our models - however any further insights can also be generalised to the SPY data.

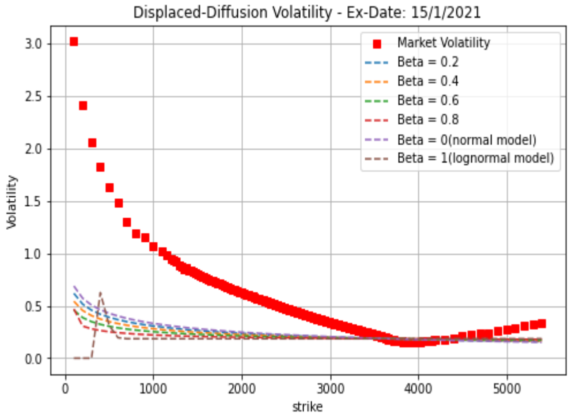
2.2 Model Calibration using Displaced-Diffusion ("DD") Model

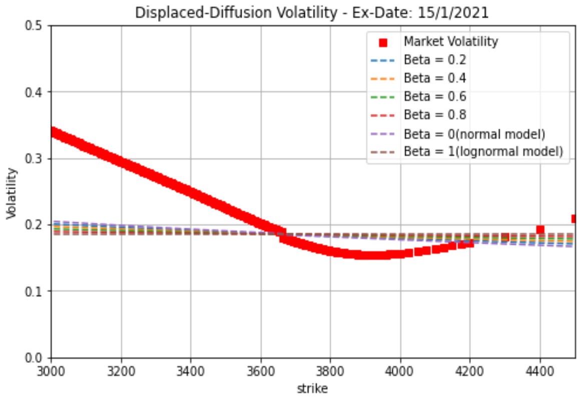
4. The DD model introduces Beta - used to scale between the normal and log normal processes. To calibrate to market volatility, we first calculate the ATM volatility for each of the 3 expiry dates.

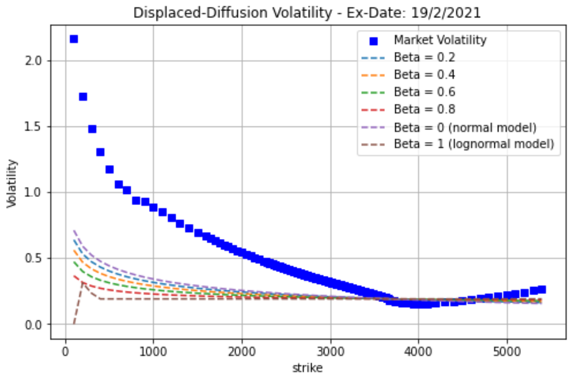
5. As can be seen in the 3 graphs below, we varied the Beta values between 0 and 1 to see which would most closely match the implied volatility surface from the SPX data.

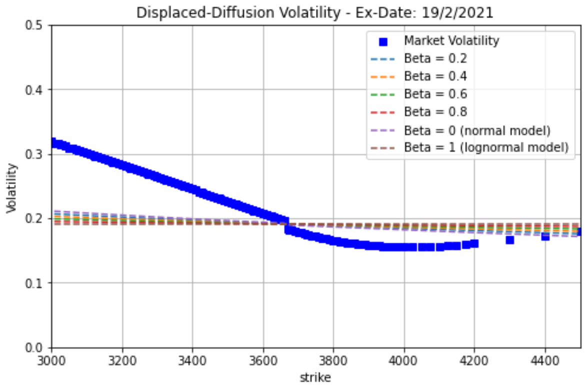






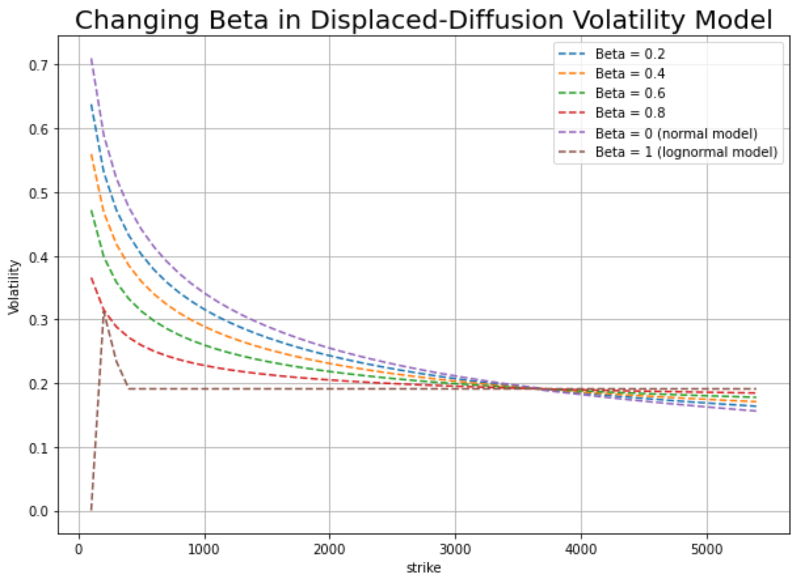






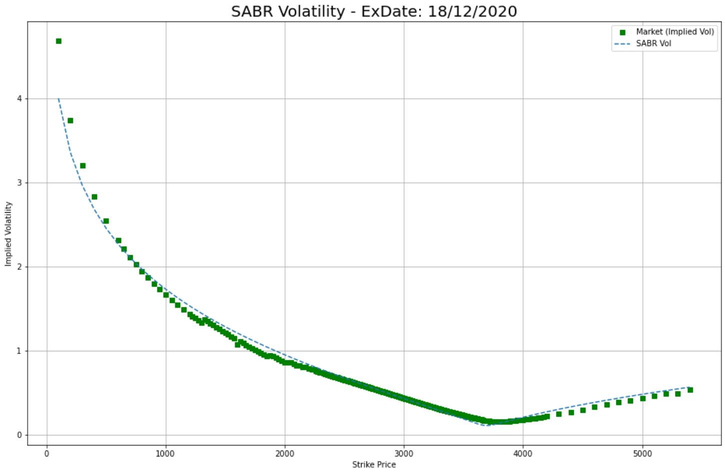
6. We observe that across the 3 expiry dates,the DD model volatilities are closest to the empirical SPX volatilities around the spot price, and diverge significantly further out of the money. This applies across all values of Beta. The Beta = 0 model was the closest to the implied SPX volatility. Due to the large divergences at low and high strikes, none of the DD models are a good fit for the market data, irrespective of the Beta values tested.

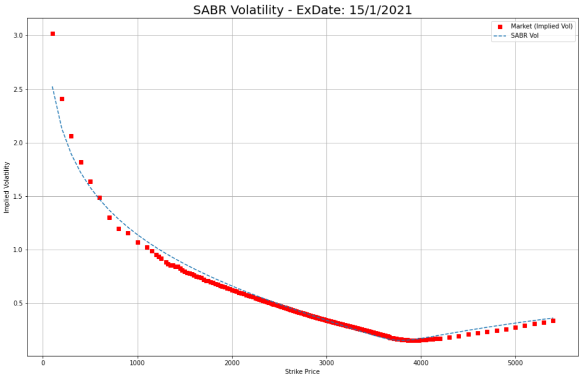
7. For completeness, we note that varying the Beta value in the DD model shifts the curve between the normal and lognormal models - as seen in the graph below. As the Beta increases, the curve flattens - ultimately becoming flat at Beta = 1.

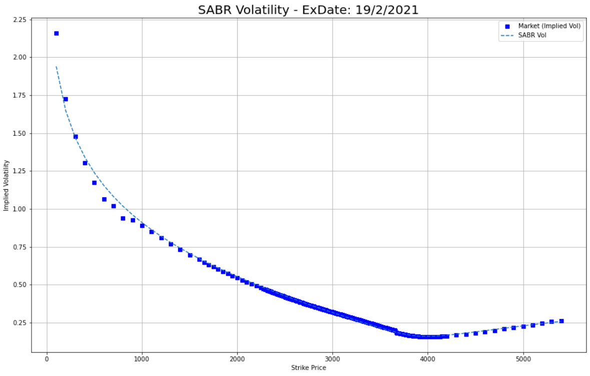


2.3 Model Calibration using SABR Model

8. The SABR model has more tunable parameters - Alpha, Beta, Rho and Nu. Fixing Beta at constant level (0.7), we use the least squares method to fit the SABR model to the implied SPX volatility surface. Our results fitting the SABR models were much better.







9. As can be seen in the three graphs above - the SABR models were able to fit the shape of the SPX volatility curve very closely across all 3 expiration dates. The calibration results for Alpha, Beta, Rho and Nu for the 3 expiries are as follows:

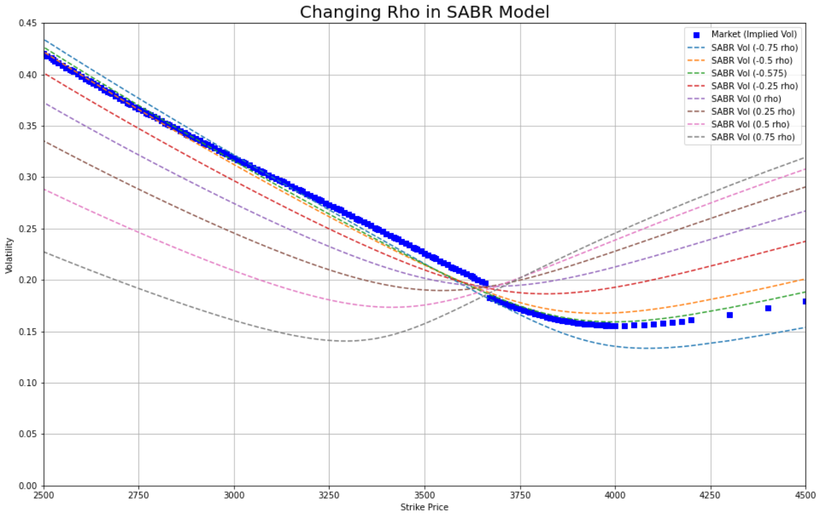
   SABR ExDate 18/12/2020 Calibration: alpha = 1.212, beta = 0.7, rho = -0.301, nu = 5.460

   SABR ExDate 15/1/2021 Calibration: alpha = 1.817, beta = 0.7, rho = -0.404, nu = 2.790

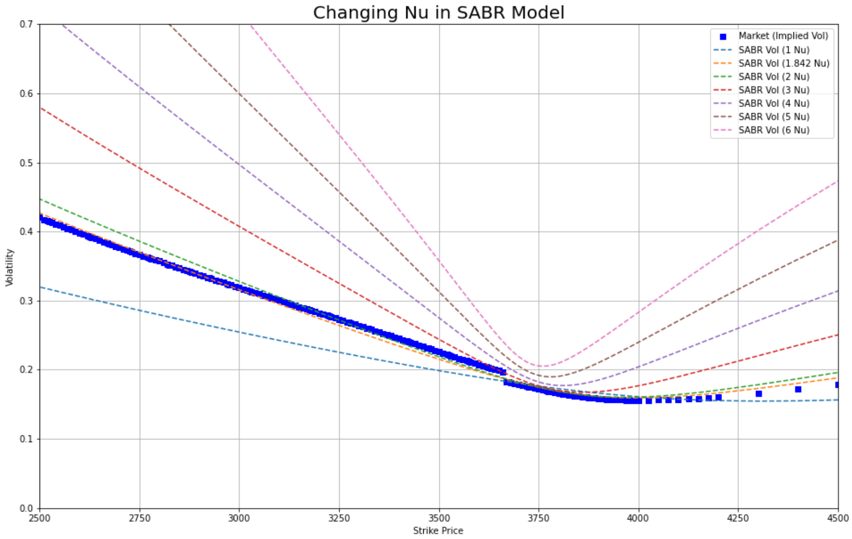
   SABR ExDate 19/2/2021 Calibration: alpha = 2.140, beta = 0.7, rho = -0.575, nu = 1.842

2.4 Varying SABR Model Parameters

10. Varying the Rho and Nu of a SABR model affects the shape of the curve differently. When Rho - which represents the degree of correlation between the volatility and its underlying - goes up, the steepness of the volatility curve decreases. OTM puts get cheaper, and conversely OTM calls get more expensive. A graph depicting this is below.



11. When Nu - which represents the volatility of volatility - increases, both OTM calls and puts increase in price. This is depicted by the increased convexity of the volatility smile in the graph below.



Part 3 (Static Replication):

3.1 Black-Scholes & Bachelier Models

12. Evaluating exotic European derivative with:

- Payoff Function: $S_T^{1/3} + 1.5 \times \log(S_T) + 10.0$
- “Model-free” Integrated Variance: $\sigma^2_{\text{MF}} T = \mathbb{E}\left[\int_0^T \sigma_t^2 \, dt\right]$

13. We are able to derive the price and implied variance using the following models fitted to the 15/1/2021 expiry of the option.

	Black-Scholes Model	**Bachelier Model**
Formula	$V_0 = e^{-rT}\left[S_0^{1/3} e^{\frac{rT}{3} - \frac{\sigma^2 T}{9}} + \frac{3}{2}\log(S_0) + \frac{3}{2}\left(r - \frac{\sigma^2}{2}\right)T + 10\right]$	$V_0 = e^{-rT}\left[S_0^{1/3} + 1.5\log(S_0) - \frac{3\sigma^2 T}{4S_0^2} + 10\right]$
Implied Variance	0.004236501	0.004263876
Derivative Pricing	37.7144	37.7136

14. Black-Scholes and Bachelier models both assume implied volatility is constant across all strike prices. Hence, at-the-money sigma would be most appropriate for the calculation of option prices as it provides the highest liquidity and best-fit representation of sigma.

3.1 Static- Replication of European Payoff

Formula:

$$e^{-rT} h(F) + \int_0^F h''(K) P(K)\, dK + \int_F^\infty h''(K) C(K)\, dK$$

$$h(S_T) = S_T^{1/3} + 1.5\log(S_T) + 10 \qquad h'(S_T) = \frac{1}{3S_T^{\frac{2}{3}}} + \frac{3}{2S_T} \qquad h''(S_T) = -\frac{2}{9S_T^{\frac{5}{3}}} - \frac{3}{2S_T^2}$$

15. Using the values in paragraph 9 above for the SABR model fit to the 15/1/2021 expiry, we are able to calculate the implied variance: 0.006337325.

16. Plugging this value into the Bachelier pricing model and Black Scholes pricing model, we calculate the price of the derivative as 37.7136 (Bachelier) and 37.7092 (BlackScholes). With reference to the prices calculated in section 3.1 above, this result is identical in the case of the Bachelier model, and within 2 decimal places for the Black Scholes.

Part 4 (Dynamic Hedging):

4.1 Brownian Motion

$$W_n(t) = \sqrt{t}\left(\frac{\sum_{i=1}^{n} X_i}{\sqrt{n}}\right)$$

4.2 Stock Price Process

$$S_T = S_0 \exp\left[\left(r - \frac{1}{2}\,\sigma^2\right)T + \,\sigma W_T\right]$$

4.3 Dynamic Hedging/Trading Strategy

$$C_t = \phi_t S_t - \psi_t B_t$$

where $\phi_t = \dfrac{\partial C}{\partial S} = \Phi\left(\dfrac{\log\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}\right)$ and $\psi_t B_t = -K\exp\left(-r(T-t)\right)\Phi\left(\dfrac{\log\left(\frac{S_t}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}\right)$

4.4 Hedging Error

17. The hedging error of the dynamic hedging strategy is generated by $\phi_t S_t + \psi_t \exp\left(\dfrac{rT}{N}\right) - \max\{0, S_t - K\}$

4.4 Dynamic Hedging Simulation Results

18. One key insight from the Black-Scholes model is the possibility of replicating an option by continuously rebalancing a portfolio consisting of the underlying stock and a risk-free asset. However, constant rebalancing of the hedge is impractical. A more workable solution is to rebalance by hedging in discrete time intervals. This iterative process will naturally have a tracking error and may result in gains/losses in excess of the replicated option payoff.

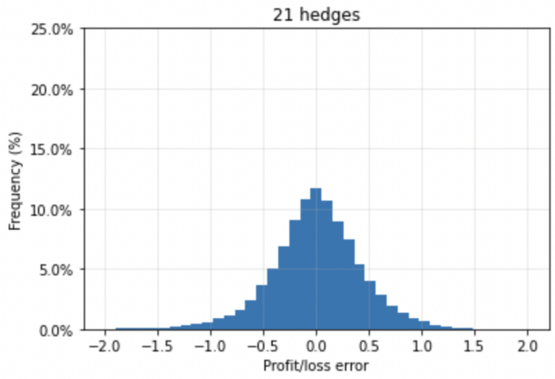
19. To illustrate this point, we can conduct a Monte Carlo simulation where we attempt to dynamically replicate an option payoff across 50,000 randomly generated paths - simulating the movement of stock prices. Setting the rebalancing frequency as N=21 and N=84, we can observe the differences in outcome.

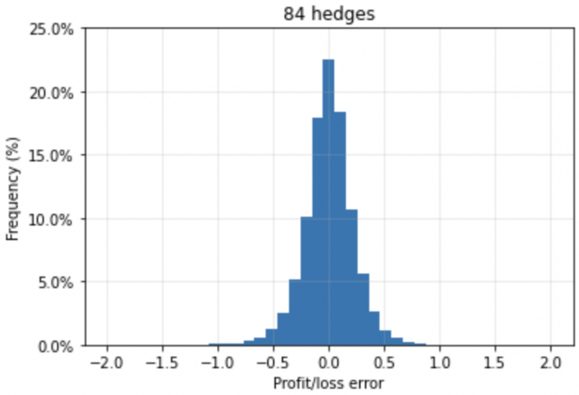
20. The Monte Carlo paths follow a 5% growth rate with a volatility σ=0.2. In both scenarios, we use the Black-Scholes hedge ratio and simulate N-number of hedging trades spaced equally in time across the tenure of the option. The initial stock price is set at S_0 = 100 and the option to be replicated is a vanilla call option with strike K = $100 and a time to expiry of T = 1/12 year.

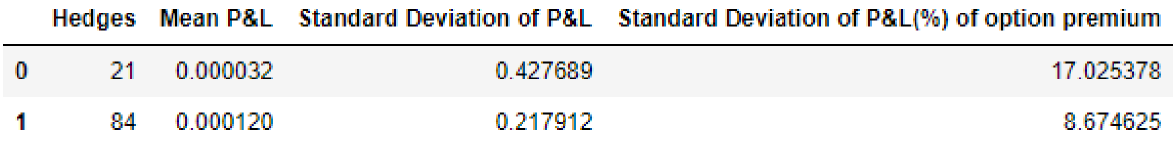
21. Using the Black-Scholes model and Monte Carlo simulation, we determined the following:
    - Fair value of the call option is 2.512
    - Mean P&L of the hedging is approximately 0
    - Standard Deviations of P&L were 0.4266 (N=21) and 0.2179 (N=84)

- Standard Deviations as a percentage of option premium were 17.03% (N=21) and 8.67% (N=84)

22. The following two graphs are the histograms of the hedging error computed for period 21 and 84:







	Hedges	Mean P&L	Standard Deviation of P&L	Standard Deviation of P&L(%) of option premium
0	21	0.000032	0.427689	17.025378
1	84	0.000120	0.217912	8.674625

23. We can make the following observations from the analysis of the Monte Carlo Simulation:
    - As the frequency of the hedging trades performed increases, the standard deviation of the P&L resulting from the hedging error declines. We note that in this case a 4x increase in N resulted in a halving of the standard deviation.
    - The distribution of hedging error follows the normal distribution, which suggests that the use of standard deviation as a risk measure is logical.
    - We note that in our simulation, trading costs were not factored in. Ultimately in the real money trading environment, the benefit of lower standard deviation and the necessarily higher trading costs must be weighed against each other in determining the frequency of hedging trades.

References

- Asset or Nothing options formula: https://quantpie.co.uk/bsm_bin_a_formula/bs_bin_a_summary.php
- Cash or Nothing options formula: https://quantpie.co.uk/bsm_bin_c_formula/bs_bin_c_summary.php
- Black Scholes formula: https://optionsformulas.com/pages/black-scholes-model.html
- Bachelier formula: https://optionsformulas.com/pages/bachelier-model.html
- Black-76 formula: https://optionsformulas.com/pages/black-76-model.html