Option Pricing Applications

An interactive presentation using Python

manos.venardos@gmail.com

Agenda

- Put-Call Parity
- No-arbitrage interval
- Risk Neutral Distribution
- Black Scholes Model
- Implied Volatility & Parameterisation
- Implied vs Realised Volatility

Using many standard 3rd party libraries, and purpose-built analytics.

In [2]:
```
#Numerics & Analytics
```

```python
import datetime
import numpy as np
import pandas as pd
from pandas_datareader import data
from pandas.plotting import autocorrelation_plot
from scipy import stats, optimize
from scipy.stats import norm
from scipy.signal import savgol_filter

#Graphs & widgets
import matplotlib.pyplot as plt
from bokeh.plotting import figure
from bokeh.io import show, output_notebook, push_notebook
from bokeh.layouts import row, column, gridplot
from bokeh.models import ColumnDataSource, Span, HoverTool, CrosshairTool
from bokeh.models.glyphs import Patch
from bokeh.models.annotations import Title
from ipywidgets import interact, interactive, interactive_output, Button, HBox
, VBox, FloatSlider, Checkbox, link

#Option Pricing
import black_scholes as bs_model
import implied_vol as iv_model
```

Quiz 1: Forward Price

A very promising stock trades at $\$100$ and pays no dividends. Equity analysts expect the price to grow by 20% per annum. Interest rates are 5%.

"We commit to exchange the stock for $\$F$ in 1 year from today".

What value of F makes this contract have zero value today?

1. $\$100$
2. $\$120$
3. $\$105$
4. Information on volatility is needed

Models

We assume a financial market where there are no arbitrage opportunities and 2 underlying assets

- A riskless money-market account M, accruing the short rate of interest r
- A risky asset with price process S, possibly delivering yield, dividend and coupon streams C

In this setup, there exists a risk-neutral probability measure Q, equivalent to the real world measure P, under which the total financial gains G^S, deflated by M are a martingale

$$M_t^{-1}G_t^S = E_t^Q[M_T^{-1}G_T^S|I_t]$$

where I_t denotes the available information at time t.

Expected gains equal the riskless rate of return r, whereas unexpected gains are model-dependent.

We will frequently make simplifying assumptions, including

- Interest rates are zero or deterministic (or operating under a forward measure)
- There are are no interim payment streams associated with the risky asset (no yields / dividends / coupons), in which case $G^S = S$

Model complexity varies significantly and a basic taxonomy is as follows

Discrete Time & State	Continuous Time & Continuous Paths	Continuous Time & Discontinuous Paths
Binomial Tree	Black Scholes	Merton
Trinomial Tree	Local & Stochastic Volatility	Jump Diffusions
etc.	etc.	etc.

Many models focus on the dynamics of the log-price $X_t = \ln S_t$.

Derivative Contracts

A derivative security V is a contingent future payout that is linked to the price of the underlying security S.

There is a big universe of such payouts and we broadly classify them as follows

Terminal	Path Dependent
Cashflow	Barrier option

Forward	Asian option
Call/Put	Forward start option
Log-contract	Variance Swap
etc.	etc.

Since derivatives are financial assets, it follows that for contracts without interim payment streams

$$V_t = M_t E_t^Q [M_T^{-1} V_T | It]$$

By setting $V_T = 1$, one obtains the price of a fixed cashflow, which we denote by

$$PV_t(T) = M_t E_t^Q [M_T^{-1} | It]$$

Forward Contract

The Forward contract is the ***obligation*** at maturity T to exchange 1 unit of S for a pre-agreed price K. Forward contracts trade at inception at a zero price, by an appropriate choice K^*

$$M_t E_t^Q[M_T^{-1}(S_T - K^*)|I_t] = 0 \Rightarrow K^* = \frac{M_t E_t^Q[M_T^{-1} S_T | I_t]}{PV_t(T)}$$

a.k.a. the ***Forward Price*** of the asset, denoted by $F_t(T)$.

The forward price depends on the asset's interim payment streams. A few interesting special cases are

- When rates are zero or deterministic $K^* = E_t^Q[S_T|I_t]$
- When the asset makes no interim payments (rates can be arbitrary) $K^* = PV_t^{-1}(T) S_t$
- When rates and interim payments are zero $K^* = S_t$

and, therefore, the forward price is model-free in many simple cases.

European Option

The European Call (Put) contracts give the holder the ***right***, but not the obligation, to buy (sell) the underlying asset S at expiry T for a pre-agreed price K. Their payouts are

$$
\begin{aligned}
C(S_T, T) &= \max(S_T - K, 0) \\
P(S_T, T) &= \max(K - S_T, 0)
\end{aligned}
$$

and their prices are therefore

$$
\begin{aligned}
c(S_t, t) &= M_t E_t^Q [M_T^{-1} C(S_T, T)] \\
p(S_t, t) &= M_t E_t^Q [M_T^{-1} P(S_T, T)]
\end{aligned}
$$

Put-Call Parity

The ***Put-Call Parity*** suggests that for a given K there is a payoff relationship at expiry T, which then translates to a relationship between prices at t

$$
\begin{aligned}
C - P &= S_T - K \\
c - p &= PV_t(T)(F_t(T) - K)
\end{aligned}
$$

Put-call parity can be tested by fitting a linear in strike regression model using OLS

$$
c_i - p_i = \alpha + \beta K_i + \epsilon_i
$$

from which we can also imply the discount factor and the asset's forward price

$$PV_t(T) = -\hat{\beta}$$
$$F_t(T) = -\frac{\hat{\alpha}}{\hat{\beta}}$$

We now load a data set of SPX options expiring on 20/12/2019, as observed on 26/12/2018.

In [4]:
```
#Load data from a flat file
undl = '^GSPC'
as_of = datetime.datetime(2018, 12, 26).date()
expiry = datetime.datetime(2019, 12, 20).date()
filename = '_'.join([undl, as_of.strftime("%Y%m%d"), expiry.strftime("%Y%m%d"
)]) + '.csv'
print('Filename: ' + filename)
```

```
Filename: ^GSPC_20181226_20191220.csv
```

In [5]:
```
#Load all data, select subset for which both calls and puts are quoted
df = pd.read_csv(filename).sort_values('Strike')
```

```
c_strikes = df[df['Type'] == 'CALL']['Strike']
p_strikes = df[df['Type'] == 'PUT']['Strike']
strikes = np.sort(np.intersect1d(c_strikes, p_strikes))
df_common = df[df['Strike'].isin(strikes)].sort_values('Strike')

df_common[['Strike', 'Type', 'Midpoint', 'Bid', 'Ask', 'Volume']].head(8)
```

Out[5]:

	Strike	Type	Midpoint	Bid	Ask	Volume
0	100	CALL	2242.80	2239.20	2246.40	1.0
113	100	PUT	0.08	0.05	0.10	530.0
1	200	CALL	2145.65	2142.10	2149.20	100.0
114	200	PUT	0.08	0.05	0.10	420.0
2	300	CALL	2048.35	2044.80	2051.90	118.0
115	300	PUT	0.10	0.05	0.15	400.0
3	400	CALL	1951.15	1947.60	1954.70	10.0
116	400	PUT	0.13	0.05	0.20	10.0

We imply the SPX forward price and USD discount factor by running an OLS regression on put-call parity. We also test the stability of the estimates by expanding the sample outwards from the ATM region.

In [6]:

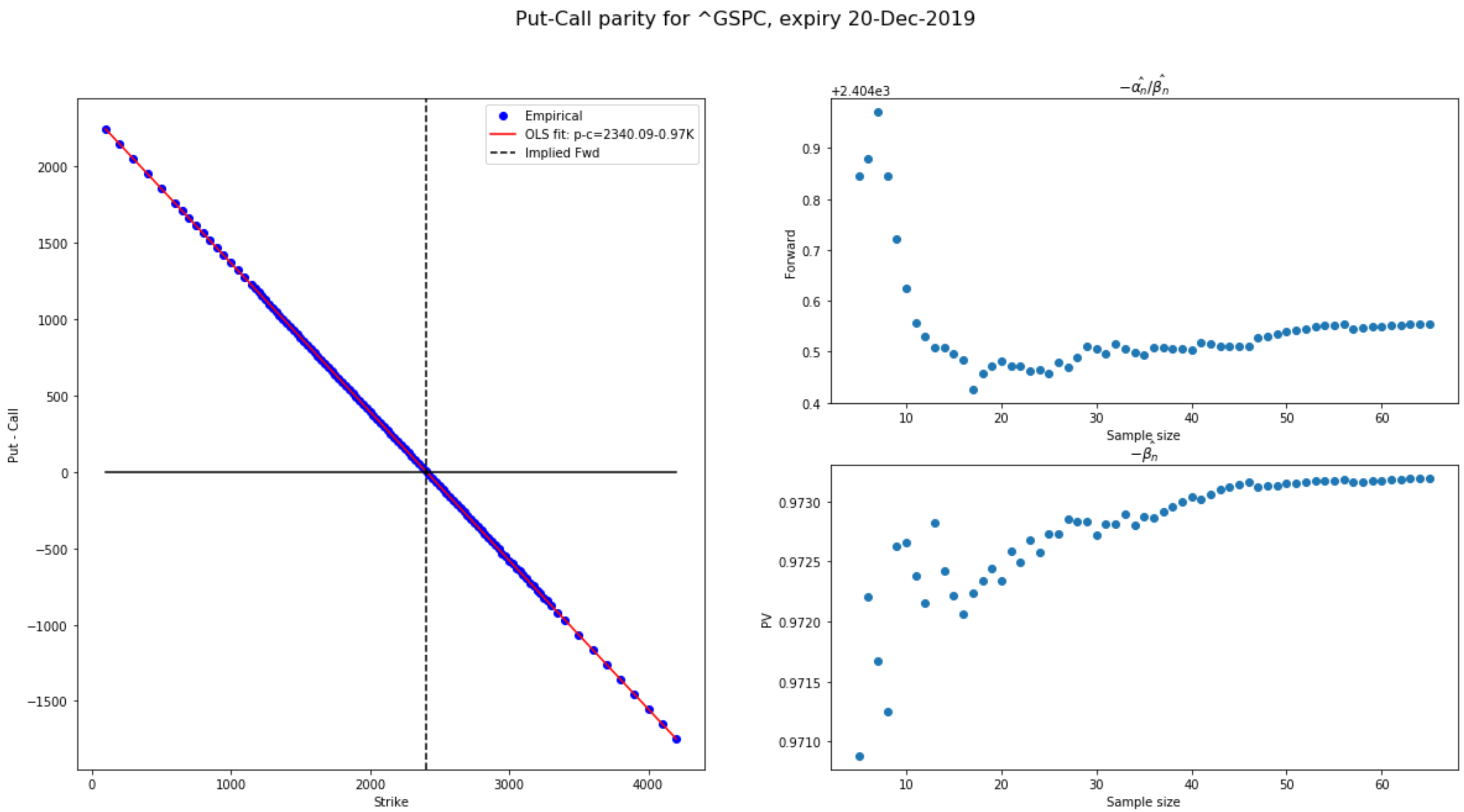
```
#Create the put-call parity relationship and fit with OLS a linear model in Strike
pc_parity = df_common[df_common['Type'] == 'CALL']['Midpoint'].values - df_common[df_common['Type'] == 'PUT']['Midpoint'].values
b, a, r_value, p_value, std_err = stats.linregress(strikes, pc_parity)
pv_hat = -b
fwd_hat = -a/b

#Also estimate coefficients starting from ATM and expanding outwards, to test the stability of estimation
atm_idx = np.abs(pc_parity - 0.0).argmin()
atm_strike = strikes[atm_idx]
intervals = np.arange(5, atm_idx)
alphas, betas = [], []
for i in intervals:
    bb, aa, rr, pp, ss = stats.linregress(strikes[atm_idx - i:atm_idx + i], pc_parity[atm_idx - i:atm_idx + i])
    alphas.append(aa)
    betas.append(bb)
```

```python
alphas = np.asarray(alphas)
betas = np.asarray(betas)

pvs_hat = -betas
fwds_hat = -alphas / betas
```

The plots suggest that put-call parity holds consistently across the strike chain.

In [8]:

```python
plot_put_call_parity(undl, expiry, strikes, pc_parity, a, b, fwd_hat, pv_hat,
intervals, fwds_hat, pvs_hat)
```



No-Arbitrage Interval

Unlike the Forward contract, there isn't a model-free price for a European option.

Instead, only a range of acceptable prices can be obtained.

For an asset without interim payments these are

$$\max(S_t - PV_t(T)K, 0) \leq c \leq S_t$$
$$\max(PV_t(T)K - S_t, 0) \leq p \leq PV_t(T)K$$

These are referred to as the ***no-arbitrage intervals*** for European options, and are actually very wide.

To derive these results, consider the two basic arguments

- Starting with put options, the payout at expiry is capped $P \leq K$, and therefore so is the price $p \leq PV_t(T)K$. Invoking put-call parity implies $C \leq S_T$ and therefore $c \leq S_t$
- Consider 2 portfolios at inception $\Pi_1 = c + PV_t(T)K$ and $\Pi_2 = S_t$. At expiry, $\Pi_1 = \max(S_T - K, 0) + K \geq S_T = \Pi_2$. Hence $c \geq \max(S_t - PV_t(T)K, 0)$

Note that the lower bounds differ from the intrinsic value of the options due to the PV term.

We now calculate the intrinsic value and no arbitrage intervals using the analytics libraries.

These methods cope with vector arguments so we calculate these across a range of spot prices.

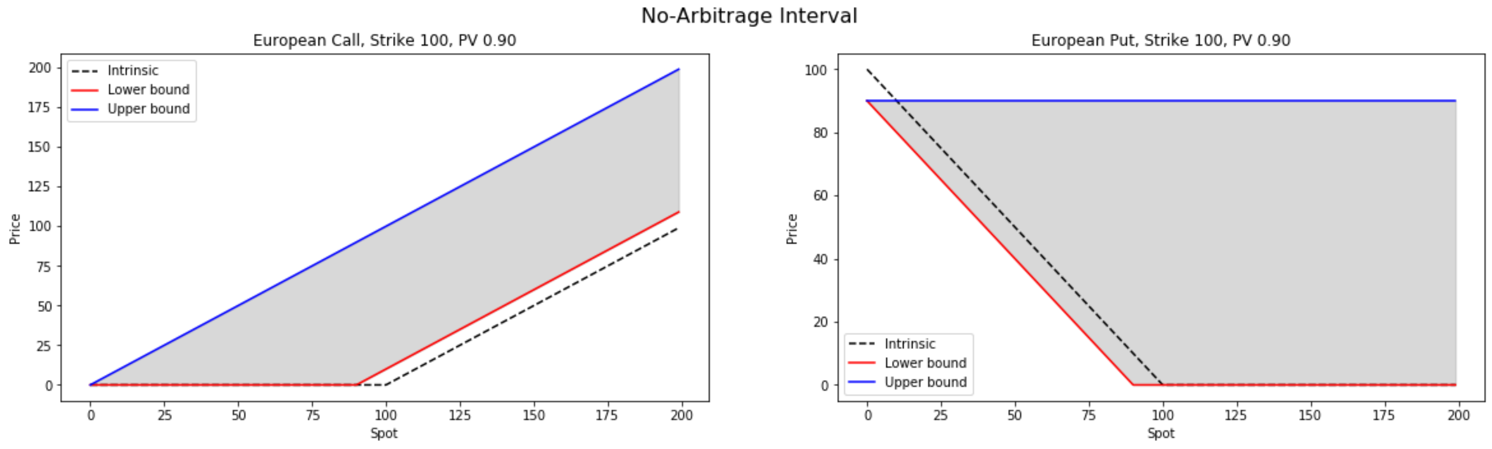
In [9]:

```
#Compute the no-arbitrage intervals for ATM call and put, across different spo
t prices
PV = 0.9
strike = 100.0
spots = np.arange(00.0, 200.0, 1.0)

c_intrinsic = bs_model.option_intrinsic_value(spots, strike, bs_model.CALL)
c_min, c_max = bs_model.option_price_interval(spots / PV, PV, strike, bs_model
.CALL)

p_intrinsic = bs_model.option_intrinsic_value(spots, strike, bs_model.PUT)
p_min, p_max = bs_model.option_price_interval(spots / PV, PV, strike, bs_model
.PUT)
```

The plots show the intrinsic value and no-arbitrage intervals for call and put options across spot prices.

In [11]:

```
plot_range_and_intrinsic(spots, strike, PV, c_intrinsic, c_min, c_max, p_intrinsic, p_min, p_max)
```



In the absence of interim payments

- European call prices always trade above their intrinsic value
- European put prices can trade below their intrinsic value

Option Strategies

In this section we discuss basic combinations of call and put options and conclude that they are also expected to trade within certain no-arbitrage intervals.

Hence, no-arbitrage implies not only conditions for each and every call and put option price individually, but also conditions across combinations of option prices.

Put Spread

The ***Put Spread*** involves long positions in a put with strike K_2 and equal and short positions in a put with strike K_1, all expiring at time T and with $K_1 < K_2$. For convenience we chose the position to be $(K_2 - K_1)^{-1}$, so the payoff becomes

$$PS(K_1, K_2) = \frac{P(K_2) - P(K_1)}{K_2 - K_1}$$

The no-arbitrage interval associated with this strategy is

$$\begin{aligned} 0 &\leq PS \leq \; < 1 \\ 0 \; &\leq ps \leq \; PV_t(T) \end{aligned}$$

for all strike pairs.

Call Spread

We define the ***Call Spread*** as

$$CS(K_1, K_2) = \frac{C(K_1) - C(K_2)}{K_2 - K_1}$$

which is also bound by the no-arbitrage interval

$$\begin{aligned} 0 &\le CS \le \; < 1 \\ 0 \; &\le \boldsymbol{cs} \le \; PV_t(T) \end{aligned}$$

for all strike pairs.

From these we deduce the spread parity relationship

$$\begin{aligned} CS + PS &= 1 \\ \boldsymbol{cs} + \boldsymbol{ps} &= PV_t(T) \end{aligned}$$

which follows due to the order of options in the numerator in the 2 spread positions.

Butterfly Spread

The ***Butterfly Spread*** involves equal and long positons in a put with strikes K_1, K_3 and twice as many short positions in a put with strike K_2, all expiring at time T and with $K_1 < K_2 < K_3$. Alternatively, it can be constructed with call options instead. For convenience we chose the position to be $((K_3 - K_2)(K_2 - K_1))^{-1}$, so the payoff becomes

$$BS(K_1, K_2, K_3) = \frac{P(K_1) - 2P(K_2) + P(K_3)}{(K_3 - K_2)(K_2 - K_1)} = \frac{C(K_1) - 2C(K_2) + C(}{(K_3 - K_2)(K_2 - K_1}$$

The no-arbitrage interval associated with this strategy is

$$\begin{aligned} 0 &\leq BS \\ 0 &\leq bs \end{aligned}$$

for all strike triplets.

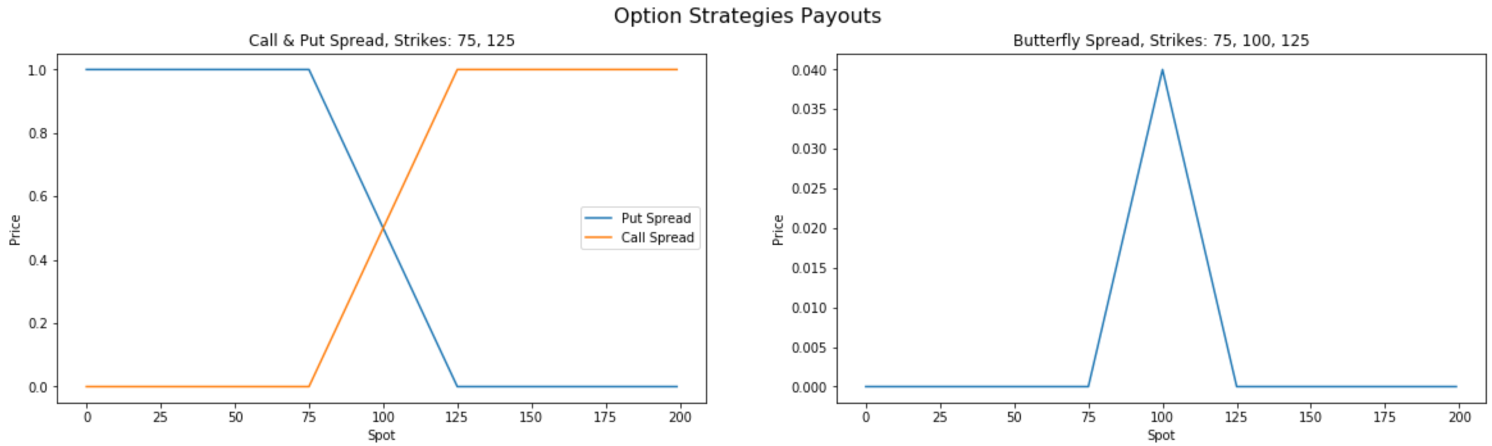
We now calculate the payoffs as a function of the spot price at expiry for the 3 option strategies.

In [12]:

```
#Calculate the payoff at maturity for a callspread, putspread and butterfly sp
read
strike1 = 75.0
strike2 = 100.0
strike3 = 125.0

c_spread = (bs_model.option_intrinsic_value(spots, strike1, bs_model.CALL) -
            bs_model.option_intrinsic_value(spots, strike3, bs_model.CALL)) /
(strike3 - strike1)

p_spread = (bs_model.option_intrinsic_value(spots, strike3, bs_model.PUT) -
            bs_model.option_intrinsic_value(spots, strike1, bs_model.PUT)) / (
strike3 - strike1)

b_spread = (bs_model.option_intrinsic_value(spots, strike1, bs_model.CALL) -
            2.0 * bs_model.option_intrinsic_value(spots, strike2, bs_model.CAL
L) +
            bs_model.option_intrinsic_value(spots, strike3, bs_model.CALL)) /
((strike3 - strike2)*(strike2 - strike1))
```

And plot their payoff as a function of the spot price at expiry.

In [14]:
```
plot_strategies_payoffs(spots, strike1, strike2, strike3, p_spread, c_spread,
b_spread)
```



Quiz 2: Arrow Securities

Prices in 1 year randomly take any value from a known set of values $S_1, \ldots, S_N$. The state space is discrete.

Consider the strike triplet $K_1 = S_{n-1} < K_2 = S_n < K_3 = S_{n+1}$ used to construct the call / put / buttefly spread strategies for some $n \in 2, . . , N-1$.

Select the incorrect statement

1. A butterfly-spread is conceptually similar to an Arrow security paying $\$1$ when $S = K_2$.
2. A call-spread is conceptually similar to the sum of Arrow securities paying $\$1$ when $S = S_i, i \leq n$.
3. A put-spread is conceptually similar to a sum of Arrow securities paying $\$1$ when $S = S_i, i \leq n$.
4. In continuous state space, the equivalent of an Arrow security is the Dirac delta function $\delta(S - K_2)$.

Risk Neutral Distribution

One can imply non-parametrically the risk neutral measure Q from market quotes directly. Assume one observes the continuum of call & put prices across strikes for a given expiry. Expressing the expected value as an integration operator over the risk neutral density q and assuming sufficient regularity, yields

$$
\begin{aligned}
p(K) &= PV\int_0^\infty \max(K-S,0)q(S)dS = \int_0^K (K-S)q(S)dS \\
\frac{\partial p}{\partial K} &= PV\int_0^K q(S)dS = PVQ(K) \\
\frac{\partial^2 p}{\partial K^2} &= PVq(K)
\end{aligned}
$$

where $Q(x) = \Pr^Q(S_T \le x)$ and $q(x) = Q'(x)$. Effectively, the first and second derivatives of put prices w.r.t. strike reveal the market's risk-neutral probability distribution and density functions respectively.

We use the quoted call & put prices to construct the call, put and buttefly spreads since

$$\lim_{K_1 \to K_2} ps(K_1, K_2) = \frac{\partial p}{\partial K}$$

$$\lim_{K_1 \to K_2} cs(K_1, K_2) = PV - \frac{\partial p}{\partial K}$$

$$\lim_{K_1 \to K_2 \to K_3} bs(K_1, K_2, K_3) = \frac{\partial^2 p}{\partial K^2}$$

It is typical to use OTM options as they are more liquid. Practical problems include

- Strike grids may be sparse and not uniform
- Chosing between Last, Bid, Ask and constructed Mid quotes
- Price flooring i.e. very OTM options trading for a minimum premium, and never at zero
- Concatenating puts with calls around ATM may result in discontinuities and kinks

We now apply this technique on the observed SPX option chain. We also employ a smoother.

In [15]:

```
#Select OTM options
```

```
otm_puts = df[(df['Type'] == 'PUT') & (df['Strike'] <= atm_strike)]
otm_calls = df[(df['Type'] == 'CALL') & (df['Strike'] >= atm_strike)]

#Compute implied probability using putspreads and callspreads
prob_put_strikes = ((otm_puts['Strike'] + otm_puts['Strike'].shift(1)) / 2.0)[
1:].values
prob_put = (1.0 / pv_hat)*(otm_puts['Midpoint'].diff() / otm_puts['Strike'].di
ff())[1:].values
prob_call_strikes = ((otm_calls['Strike'] + otm_calls['Strike'].shift(1)) / 2.
0)[1:].values
prob_call = 1.0 + (1.0 / pv_hat)*(otm_calls['Midpoint'].diff() / otm_calls['St
rike'].diff())[1:].values

prob_strikes = np.append(prob_put_strikes, prob_call_strikes)
prob = np.append(prob_put, prob_call)

#Also a smoothed version of probability
prob_hat = savgol_filter(prob, 51, 5) # window size 51, polynomial order 3

#Compute the implied density by butterfly spreads (or spreads of putspreads)
dens_strikes = (prob_strikes[1:] + prob_strikes[:-1]) / 2.0
dens = np.diff(prob) / np.diff(prob_strikes)
```

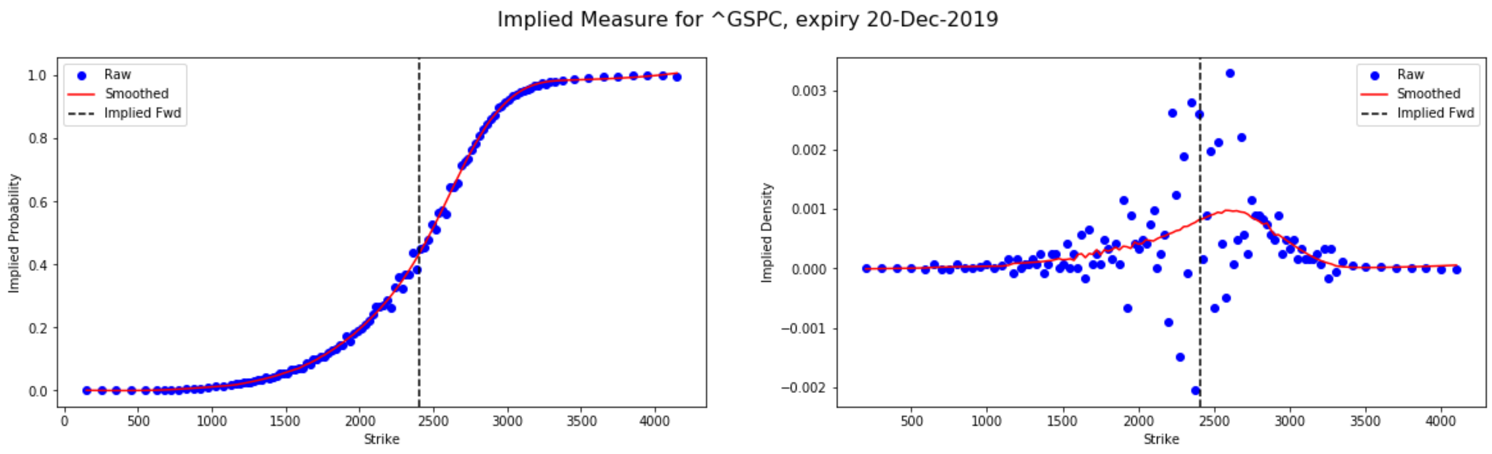
```
#And a smoothed version of the density
dens_hat = np.diff(prob_hat) / np.diff(prob_strikes)
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/scipy/signal/_array
tools.py:45: FutureWarning: Using a non-tuple sequence for multidimensiona
l indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In
the future this will be interpreted as an array index, `arr[np.array(seq)]
`, which will result either in an error or a different result.
  b = a[a_slice]
```

Numerical noise eventually creeps in. Employing a smoother seems inevitable for 2nd order derivatives.

In [17]:

```
plot_implied_distribution(undl, expiry, fwd_hat, prob_strikes, prob, prob_hat,
dens_strikes, dens, dens_hat)
```



The no-arbitrage intervals are linked to the existence of a well-defined risk neutral distribution and density

- Call prices are decreasing and convex in strike
- Put prices are increasing and convex in strike

Black Scholes Model

Log-returns are assumed to be normal i.i.d. and in the absence of interim payments and fixed rates, they are governed by the SDE

$$dX_t = \left(r - \frac{\sigma^2}{2}\right) dt + \sigma dW_t$$

where σ is the the annualised volatility of log-returns and W_t a standard Brownian motion. Measuring returns in discrete intervals of lenght δ, one obtains

$$X_t - X_{t-\delta} \sim N\left[\left(r - \frac{\sigma^2}{2}\right)\delta, \sqrt{\delta}\right]$$

While simplistic and unrealistic, it has become an industry standard, for reasons discussed below.

We dowload historic spot prices for SPX to test the normal i.i.d. assumption.

In [18]:

```
df_spot = data.DataReader(name=undl, data_source='yahoo', start='2000-01-01',
end = as_of)
df_spot.tail(5)
```

Out[18]:

	High	Low	Open	Close	Volume
Date					

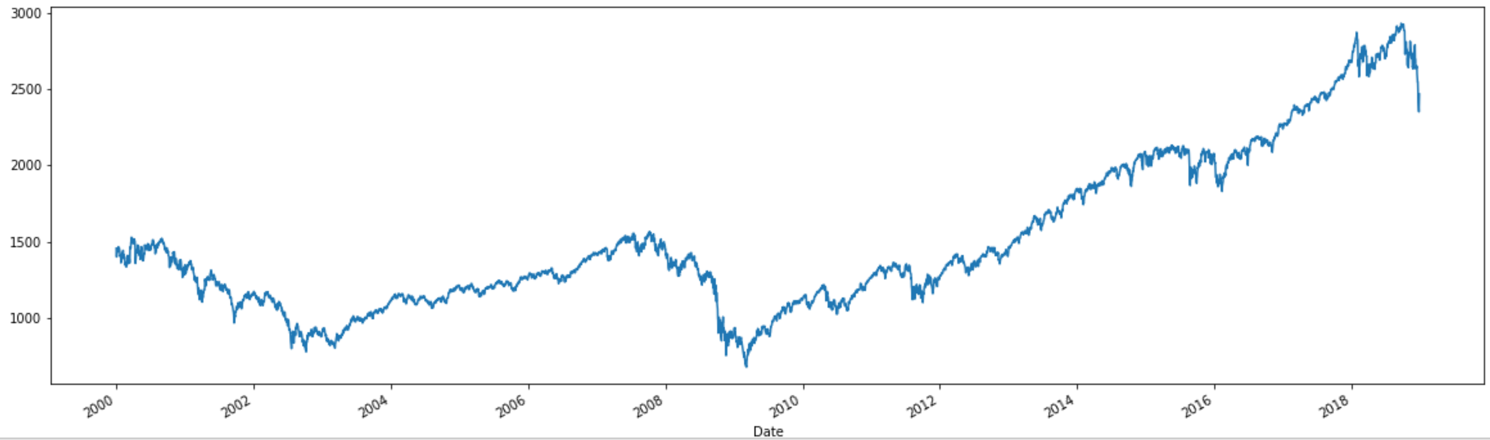
2018-12-19	2585.290039	2488.959961	2547.050049	2506.959961	5127940000	25
2018-12-20	2509.629883	2441.179932	2496.770020	2467.419922	5585780000	24
2018-12-21	2504.409912	2408.550049	2465.379883	2416.620117	7609010000	24
2018-12-24	2410.340088	2351.100098	2400.560059	2351.100098	2613930000	23
2018-12-26	2467.760010	2346.580078	2363.120117	2467.699951	4233990000	24

In [19]:

```
df_spot['Close'].plot(figsize = (20, 6))
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f49214d3208>
```



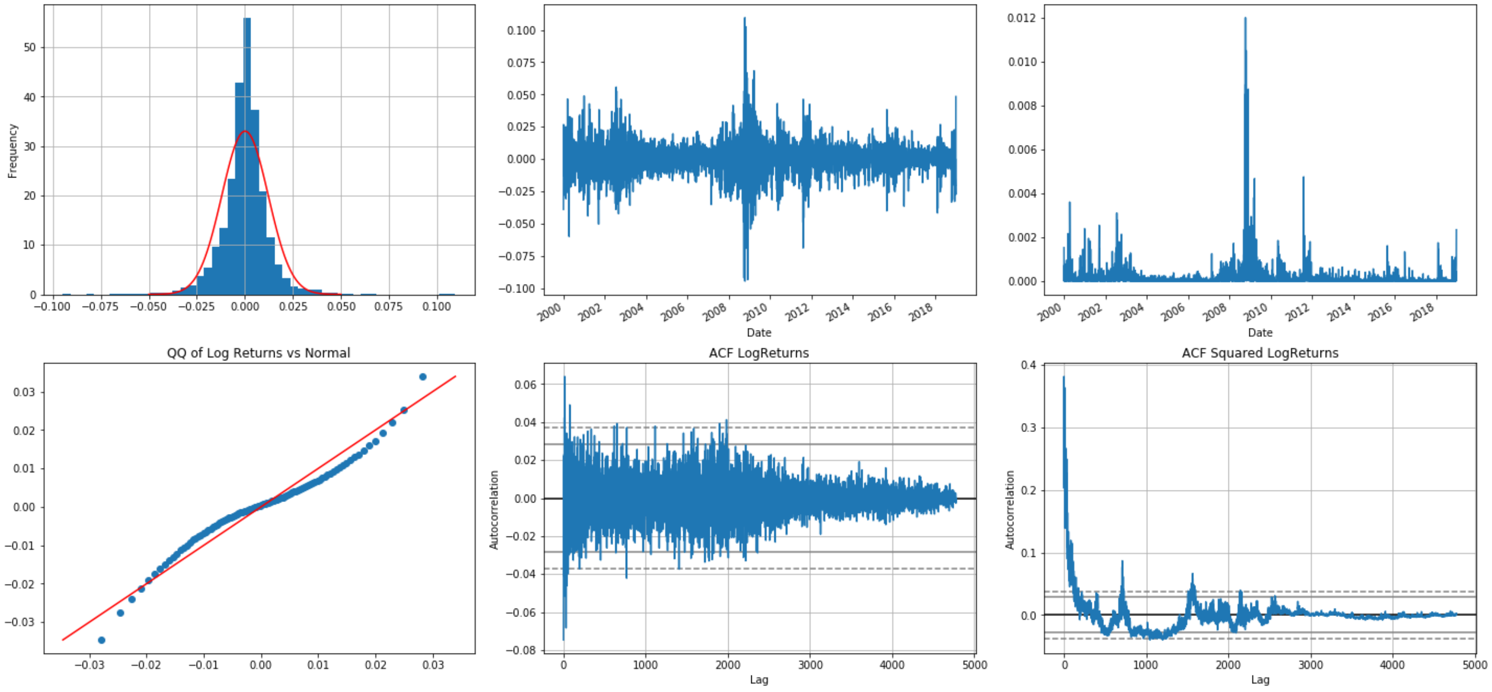
In [21]:
```
plot_log_returns_analysis(df_spot['Close'], undl, 1, '1D')
```

^GSPC 1D

Histogram of LogReturns of Normal

LogReturns

Squared LogReturns



Log-returns deviate significantly from the normal i.i.d. paradigm.

Marginal Distribution

- Negative samples have more probability mass than the normal density
- See histogram and QQ plots

Conditional Dependence

- While levels are serially uncorrelated, the squares are not, so independence is violated
- See ACF plots

Log-returns exhibit volatility clustering.

European Option

We begin our exploration with the closed form solution for the price of a European Call option

$$
\begin{aligned}
c(S_t, t) &= PV_t(T)\left[F_t(T)\Phi(d_1) - K\Phi(d_2)\right] \\
d_1 &= \frac{1}{\Sigma\sqrt{T-t}}\left(\ln\left(\frac{F_t(T)}{K}\right) + \frac{\Sigma^2(T-t)}{2}\right) \\
d_2 &= d_1 - \Sigma\sqrt{T-t}
\end{aligned}
$$

where $PV_t(T) = e^{-r(T-t)}$, $F_t(T) = PV_t^{-1}(T)S_t$ and Φ is the normal cumulative distribution function.

Quiz 3: ATM Option Price

Interest rates are zero and there are no interim payments.

Consider a call option with expiry T and strike $K = F_t(T)$.

What is approximately its premium as a $\%$ of the forward price F?

1. $\Sigma\sqrt{T-t}$
2. $\Sigma^2(T-t)$
3. Not that straightforward
4. $\frac{\Sigma\sqrt{T-t}}{\sqrt{2\pi}}$

No-Arbitrage Interval

The call option price c as a function of volatility σ

- Achieves the lower bound for a call option price since $\lim_{\sigma \to 0} c = \max(S_t - PV_t(T)K, 0)$
- Achieves the upper bound too since $\lim_{\sigma \to \infty} c = S_t$

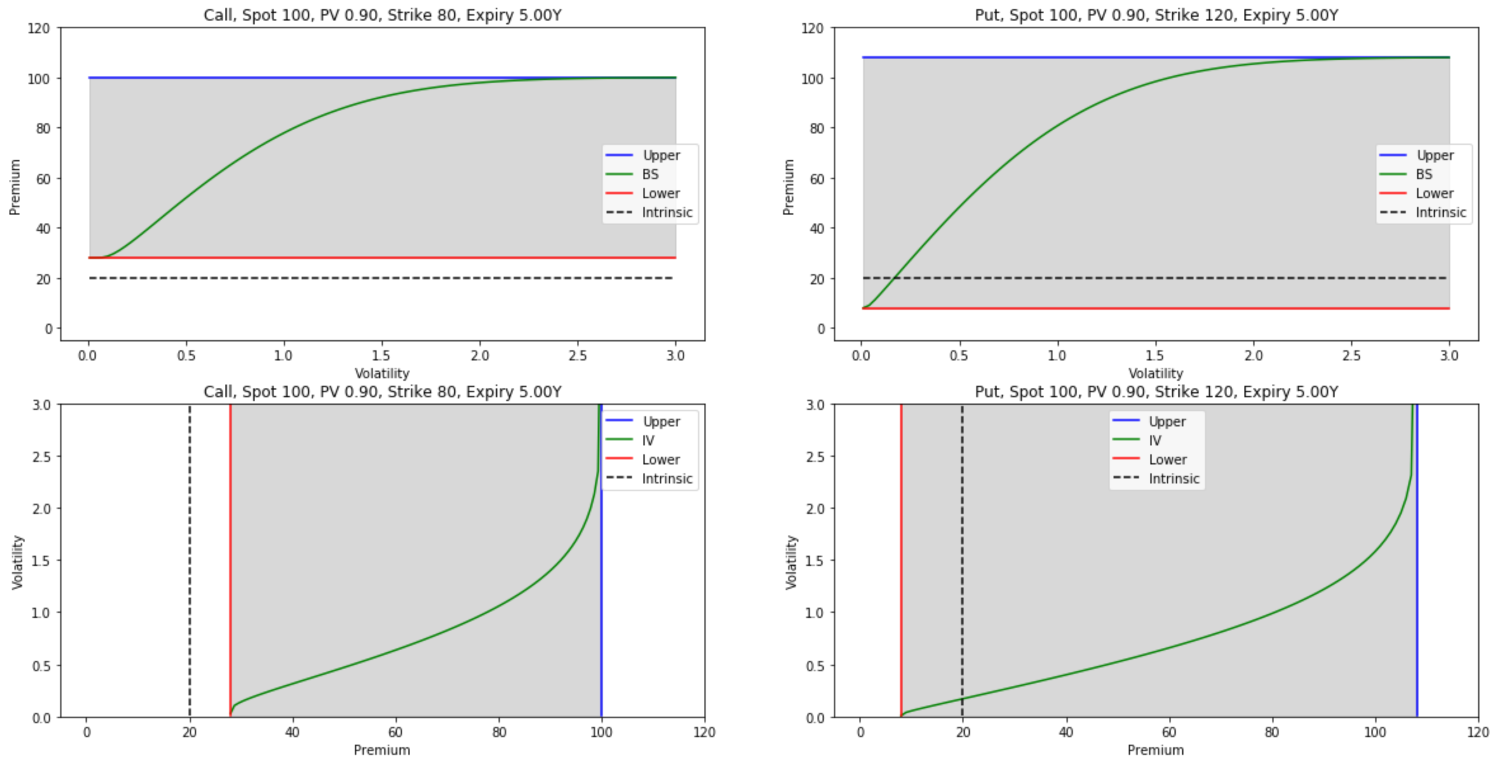
- Is continuous and increasing

Similarly, as volatility varies in $[0, \infty)$, the put price also spans the whole no-arbitrage interval $[\max(PV_t(T)K - S_t, 0), PV_t(T)K]$.

The monotonic and invertible price-volatility relationship is illustrated below.

In [24]:
```
plot_price_vol_mapping(S, PV, K1, K2, T, vols, c1_max, c1_price, c1_min, c1_intrinsic, p1_max, p1_price, p1_min, p1_intrinsic, c1_vols, c1_prices, p1_vols, p1_prices)
```

Option Price vs Volatily



For every market price V^M in the no-arbitrage interval, there exists a unique volatility σ^* that perfectly matches the price when using the BS formula - hence the term ***Implied Volatility***.

Since a) the model obeys the put-call parity by construction, b) implied volatility is a 1-1 mapping with prices and c) market quotes obey the put-call parity due to no-arbitrage, we conclude that call and put options have the same implied volatility for each K, T.

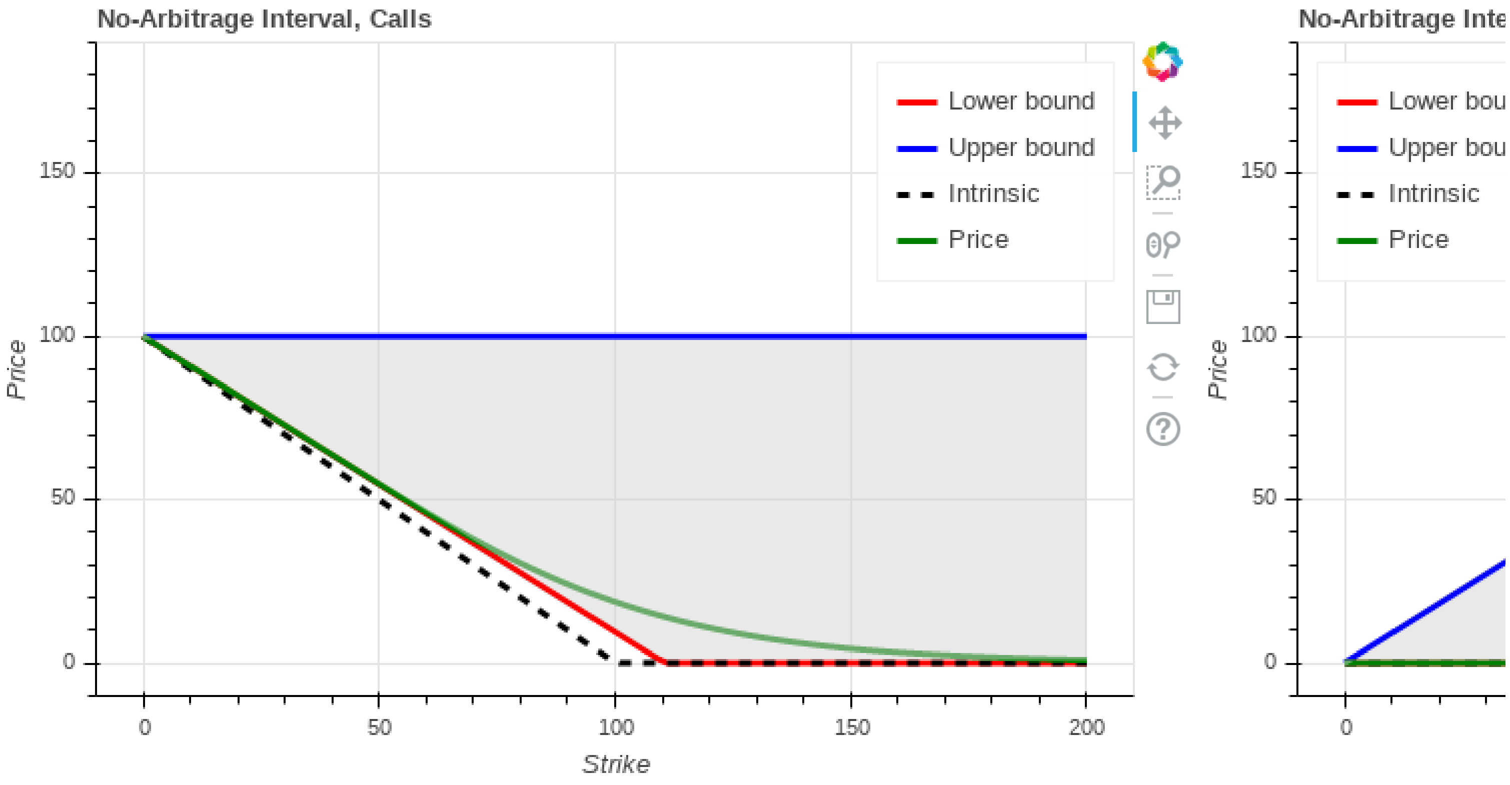
The existence and uniqueness of implied volatility is a major reason why the Black Scholes model has become an industry standard, despite all the empirical evidence against its assumptions.

Implied volatility is a summary statistic for option prices, comparable to the yield for bond prices.

Property	**Implied Volatility (IV) and Option Prices**	**Yield (Y) and Bond Prices**
No loss of information	IV is a 1-1 mapping with option prices	Y is a 1-1 mapping with bond prices
Normalises prices of different securities	Removes the effects of spot, strike, expiry etc.	Removes the effects of coupons, expiry etc.
Stationarity	IV is mean reverting, whereas option prices are not	Y is mean reverting, whereas bond prices are not

In [26]:
```
interactive_no_arb_interval(np.linspace(0.01, 200.0, 100), 100.0, 0.05, 0.25,
2.0, 1.0)
```

Loading BokehJS ...



We draw the following conclusions

- For any configuration of $S, r, T-t$, the BS call and put option prices across different strikes span their no-arbitrage intervals as volatility varies in $[0, \infty)$, and they do so simultaneously
- For any given configuration of $S, r, T-t$ and any given market price $V^M(K_1)$, the option prices for all other strikes are pinned down completely by the model. As a consequence there is no freedom to match any other market price $V^M(K_2)$ unless it happens to trade at an identical volatility

Implied Volatility Surface

Almost always, there is significant variation in the observed implied volatilities across strikes and expiries, and no single choise of volatility can match multiple market quotes.

Practitioners have opted for patching the Black Scholes model by using a different level of volatility for each K, T. We introduce the following terminology

- The collection of the implied volatilities across K, T is the ***Volatility Surface*** $\Sigma(K, T)$
- The cross-section across a fixed T is the ***Volatility Smile*** (or Skew)
- The cross-section across a fixed K is the ***Volatility Term Structure***

In order to compute the implied volatility for each observed quoted call & put price, we invert the Black Scholes formula for each K, T with $PV_t(T)$ and $F_t(T)$ set to their OLS fitted values.

We employ the analytic libraries to compute the implied volatility using a basic bisection algorithm.

In [27]:

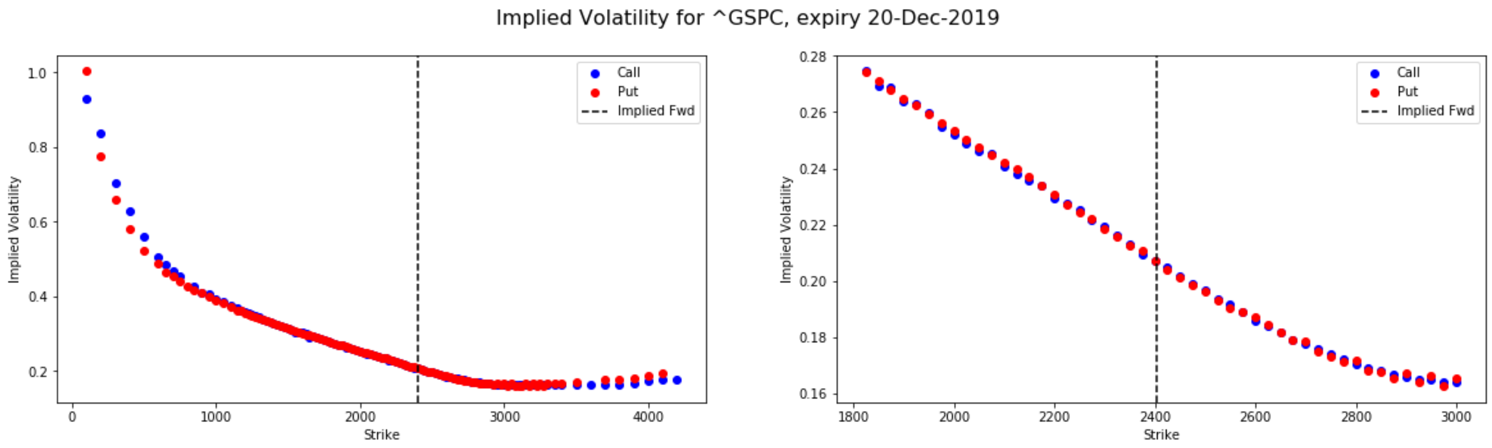
```
spot_close = 2400.00
tau = (expiry - as_of)/ datetime.timedelta(days=1)/365.25

implied_vols = [bs_model.option_vol(r['Midpoint'], fwd_hat, pv_hat, r['Strike'
], tau, r['Type']) for i, r in df_common.iterrows()]
df_common['IV'] = implied_vols

valid_calls = df_common[df_common['IV'].notnull() & (df_common['Type'] == 'CAL
L')]
valid_puts = df_common[df_common['IV'].notnull() & (df_common['Type'] == 'PUT'
```

```
)]

few_strikes = strikes[(strikes > 0.75 * fwd_hat) & (strikes < 1.25 * fwd_hat)]
valid_calls2 = valid_calls[valid_calls['Strike'].isin(few_strikes)]
valid_puts2 = valid_puts[valid_puts['Strike'].isin(few_strikes)]
```

We plot the SPX implied volatilities for the option chain.

In [29]:

```
plot_implied_volatility(undl, expiry, fwd_hat, pv_hat, valid_calls, valid_call
s2, valid_puts, valid_puts2)
```



This empirically validates that observed put & call options obey the put- call parity, and also share the same implied volatility for each K, T.

However, erroneous results are obtained when using the wrong $PV_t(T)$ or $F_t(T)$.

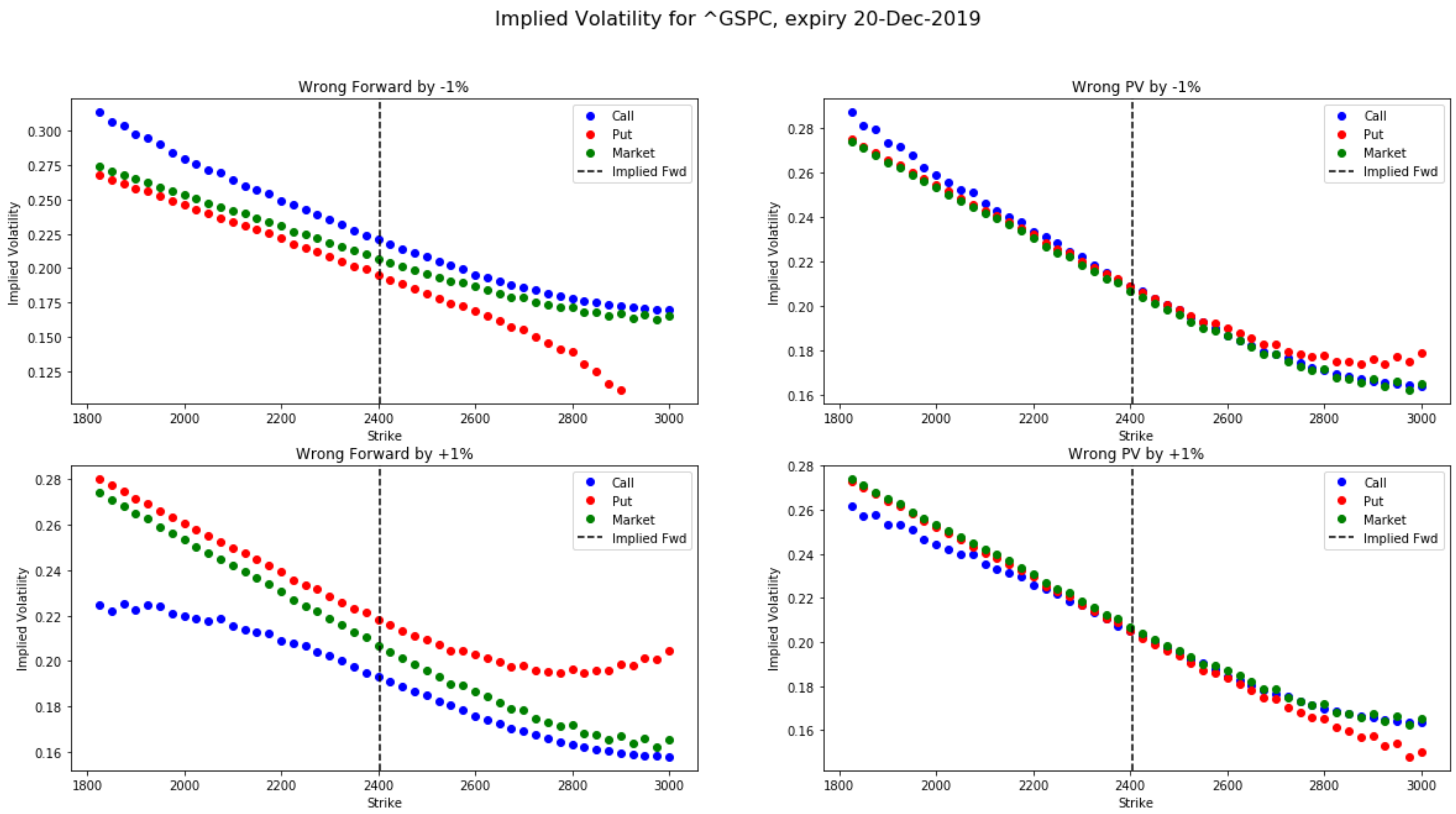
In [30]:
```
df_common_wrong = df_common.copy()

implied_vols_wrong_fwd_d = []
implied_vols_wrong_pv_d = []
implied_vols_wrong_fwd_u = []
implied_vols_wrong_pv_u = []
for i, r in df_common_wrong.iterrows():
    iv_wrong_fwd_d = bs_model.option_vol(r['Midpoint'], 0.99 * fwd_hat, pv_hat
, r['Strike'], tau, r['Type'])
    iv_wrong_fwd_u = bs_model.option_vol(r['Midpoint'], 1.01 * fwd_hat, pv_hat
, r['Strike'], tau, r['Type'])
    iv_wrong_pv_d = bs_model.option_vol(r['Midpoint'], fwd_hat, 0.99 * pv_hat,
r['Strike'], tau, r['Type'])
    iv_wrong_pv_u = bs_model.option_vol(r['Midpoint'], fwd_hat, 1.01 * pv_hat,
r['Strike'], tau, r['Type'])
    implied_vols_wrong_fwd_d.append(iv_wrong_fwd_d)
    implied_vols_wrong_fwd_u.append(iv_wrong_fwd_u)
```

```
    implied_vols_wrong_pv_d.append(iv_wrong_pv_d)
    implied_vols_wrong_pv_u.append(iv_wrong_pv_u)

df_common_wrong['IV Wrong Fwd Dn'] = implied_vols_wrong_fwd_d
df_common_wrong['IV Wrong Fwd Up'] = implied_vols_wrong_fwd_u
df_common_wrong['IV Wrong PV Dn'] = implied_vols_wrong_pv_d
df_common_wrong['IV Wrong PV Up'] = implied_vols_wrong_pv_u

calls_wrong = df_common_wrong[df_common_wrong['IV'].notnull() & (df_common_wro
ng['Type'] == 'CALL') & (df_common_wrong['Strike'].isin(few_strikes))]
puts_wrong = df_common_wrong[df_common_wrong['IV'].notnull() & (df_common_wron
g['Type'] == 'PUT') & (df_common_wrong['Strike'].isin(few_strikes))]
```

Put-call parity is broken due to the mis-specification of the implied forward and discount factor.

In [32]:

```
plot_wrong_implied_vols(undl, expiry, fwd_hat, pv_hat, calls_wrong, puts_wrong
)
```



The observed market prices obey put-call parity at the correct levels of $PV_t(T)$ and $F_t(T)$.

- **Forward mismatch**: at a lower F, but fixed PV, call prices *should* have been cheaper, and put prices more expensive - but they are not. In fact, call prices *seem* more expensive and put prices cheaper than they should be. Hence, volatility implied from calls is erroneously higher than implied from puts, with the real implied volatility in between. The opposite effect takes place when one overestimates F.

- **PV mismatch**: at a lower PV but fixed F, both call and put prices *should* have been cheaper - but they are not. In fact, both call and put prices *seem* more expensive. Hence, the implied volatilities of both calls and puts are greater than the real implied volatility. The opposite effect takes place when one overestimates PV.

For very deep out or in the money options, F or PV mismatch may result in an erroneous violation of the no-arbitrage interval, in which case an implied volatility will not exist.

Implied Volatility Stylised Facts

The shape of the volatility surface across K and T varies across underlyings and market conditions

- Equity indices tend to have downward sloping implied volatility across K i.e. the skew is the dominant effect. A major driver is the demand for broad insurance against a market drop; low strike puts are bid up, which then translates to higher implied volatilities. Also, equity markets tend to be more volatile during crashes, which is priced-in by the smile.
- Single stock volatility surfaces also tend to be downward sloping in K, but there may be idiosyncratic drivers to price-in a market rally, thus resulting in convexity. In extreme cases of positive expectations the smile may even be upward sloping e.g. M&A activity.
- FX implied volatility tends to be more symmetric in K as there is natural demand for both currencies.
- In low (high) volatility environments the volatility term structure is typically upward (downward) sloping as the term structure is pricing-in some reversion to a volatility average.

Implied Volatility Parameterisation

Only a discrete set of option prices are observable in the market, it is thus typical to parameterise volatiliy across (K, T) in order to interpolate and extrapolate to unbservable points.

The parameterisation should be rich and flexible enough to fit the stylised facts and adapt to varying market conditions.

Market Observables

Consider an equity option market that is liquid arount an ATM strike K^* and for which we observe the implied volatility for 3 strikes, and focus on

$$\begin{aligned} ATM &= \Sigma^M(100\% K^*) \\ skew &= \frac{1}{2}\left(\Sigma^M(90\% K^*) - \Sigma^M(110\% K^*)\right) \\ convx &= \frac{1}{2}\left(\Sigma^M(90\% K^*) - 2 \times \Sigma^M(100\% K^*) + \Sigma^M(110\% K^*)\right) \end{aligned}$$

In this setup, ***skew*** is a measure of the slope and ***convexity*** a measure of curvature around K^*. In fact, both are finite difference estimates of the first and second mathematical derivatives w.r.t. strike.

Quadratic in Moneyness

Consider now as a starting point the quadratic

$$q(x) = a + bx + \frac{1}{2}cx^2$$

We define the implied volatility as a quadratic in proportional forward moneyness

$$\Sigma_q(K) = q(pm(K))$$
$$pm(K) = \frac{K - K^*}{K^*}$$
$$K^* = PV^{-1}K^{ref}$$

We have chosen the ATM strike to be the forward price of a fixed reference strike K^{ref}, typically the spot price of the underlying when the volatility is fitted.

Implied volatility is a quadratic and we can link the 3 parameters to the observable points

$$
\begin{aligned}
a &= \Sigma_q(K^*) = ATM \\
b &= \Sigma_q'(K^*) \approx skew \times 10 \\
c &= \Sigma_q''(K^*) \approx convx \times 100
\end{aligned}
$$

where the multipliers of 10 and 100 are due to the market observables being only the numerators of the finite difference estimates.

The parameterisation is intuitive since all parameters can be easily linked to market observables and can achieve a variety of different shapes to match the various stylised facts and market environments.

However, once the ATM region is fit, there are no degrees of freedom left to fit the wings; these simply inherit the quadratic behaviour of the ATM region and can thus explode to $+\infty$.

Quadratic in Generalised Moneyness

Consider a monotonic function f, and generalise the volatility parameterisation to

$$\Sigma_{q \bullet f}(K) = q(f(pm(K)))$$

If f is such that $f(0) = 0$, $f'(0) = 1$ and $f''(0) = 0$, the quadratic is locally preserved

$$\begin{aligned}\Sigma_{q \bullet f}(K^*) &= \Sigma_q(K^*) \\ \Sigma'_{q \bullet f}(K^*) &= \Sigma'_q(K^*) \\ \Sigma''_{q \bullet f}(K^*) &= \Sigma''_q(K^*)\end{aligned}$$

Such an f impacts directly the behaviour of the wings.

$$f(x|w) = w \tanh\left(\frac{x}{w}\right), w > 0$$

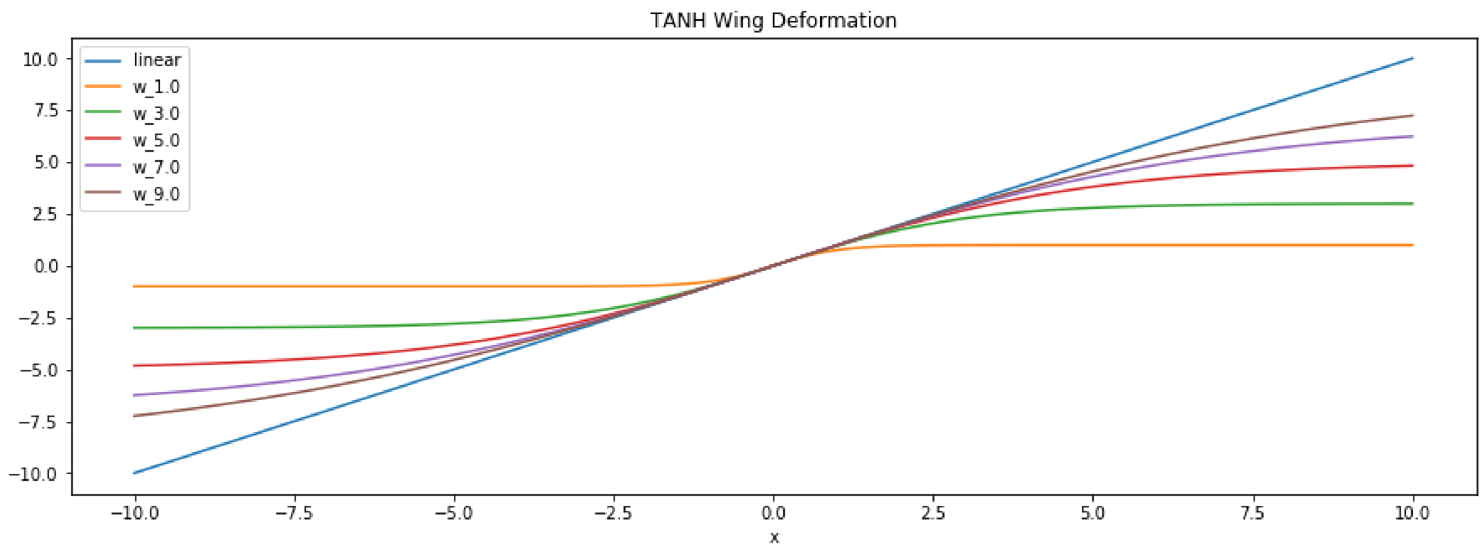
For high values of w, the deformation has minimal impact since $\lim_{w \to \infty} f(x|w) = x$.

For low values of w however, the smile devietes significantly from a quadratic at the wings.

In [34]:

```
xs = np.linspace(-10, 10, 101)
ws = np.arange(1.0, 11.0, 2.0)
```

```
plot_tanh(xs, ws)
```



Fitting & Calibration

In order to fit the volatility parameterisation to market quotes, use a typical objective function minimisation

$$\min_{a,b,c,w} \sum_{i=1}^{N} \left(\Sigma(K_i|a,b,c,w) - \Sigma^M(K_i)\right)^2$$

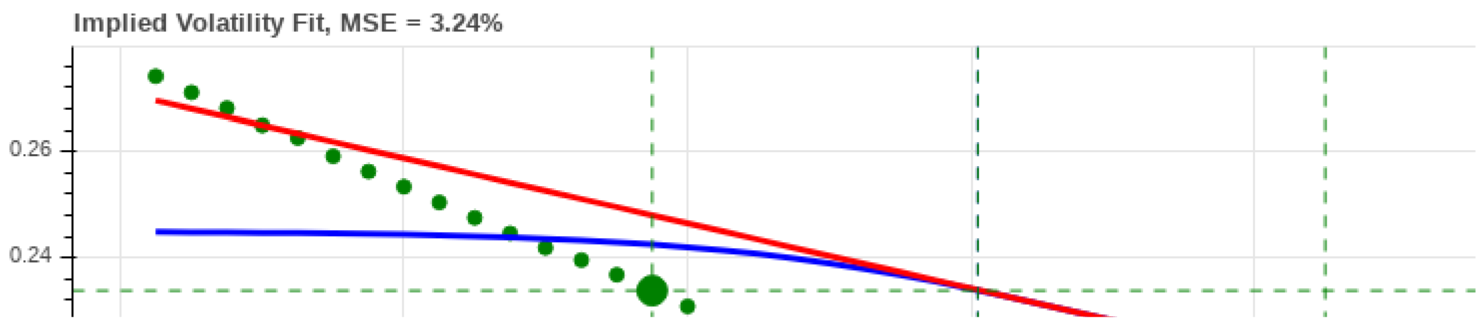
The optimal parameters provide the best fit to current market quotes and can be used to evaluate the volatility for any other strike.

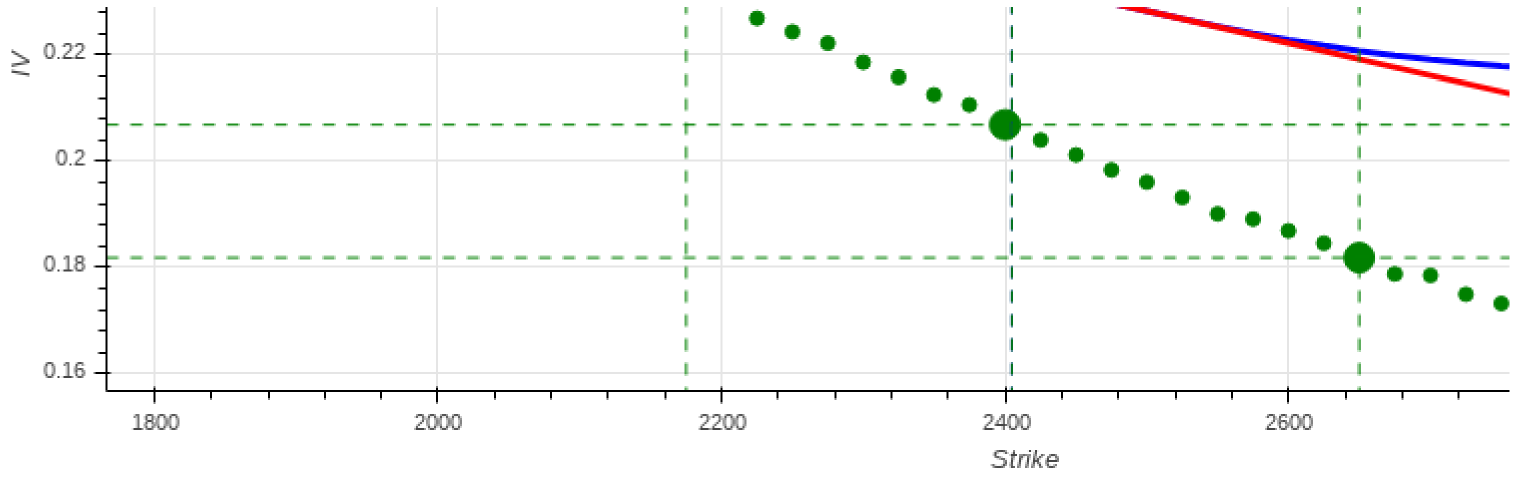
Fitting volatilities is preferrable to fitting prices, as the latter will place excessive focus on ITM prices and is unlikely to fit weel the liquid OTM contracts.

We now fit this functional form to SPX implied volatilities.

In [36]:
```
interactive_vol_fitting(fwd_hat, valid_puts2, fwd_hat, 0.23, -0.15, 0.01, 0.1)
```

Loading BokehJS ...





No-Arbitrage Revisited

A functional form for implied volatility is not a model. It is a description of market prices via the operational definition $p(K) = BS(K, \Sigma(K))$.

When rates are zero, the implied probability distribution is given by

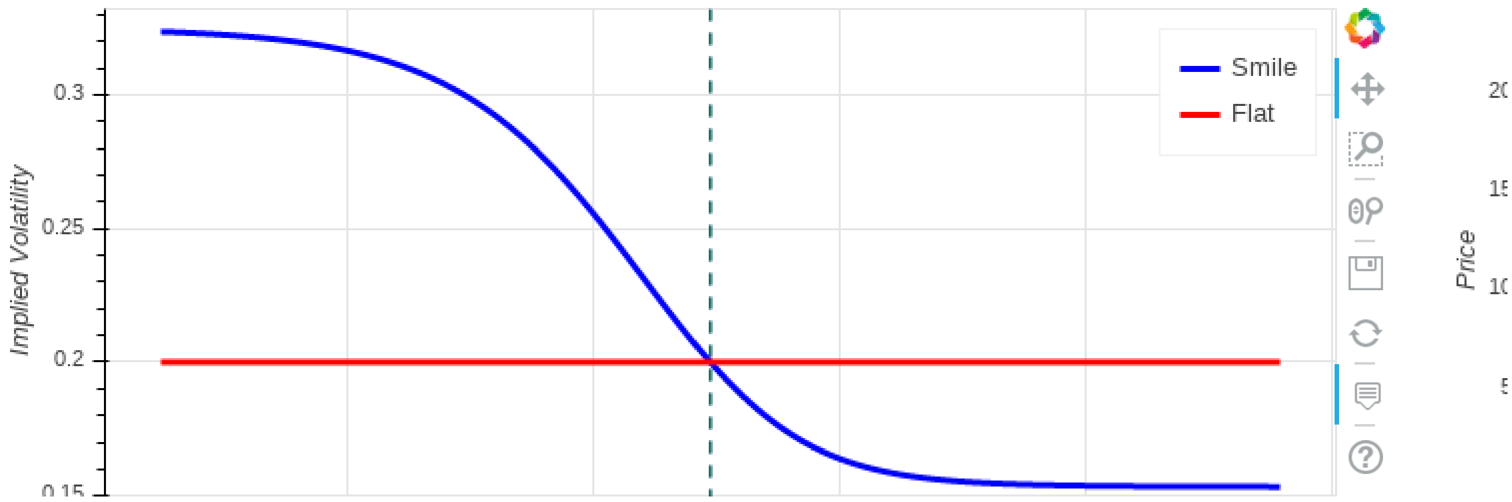
$$Q = \frac{dp}{dK} = \frac{\partial BS}{\partial K} + \frac{\partial BS}{\partial \Sigma}\frac{d\Sigma}{dK}$$

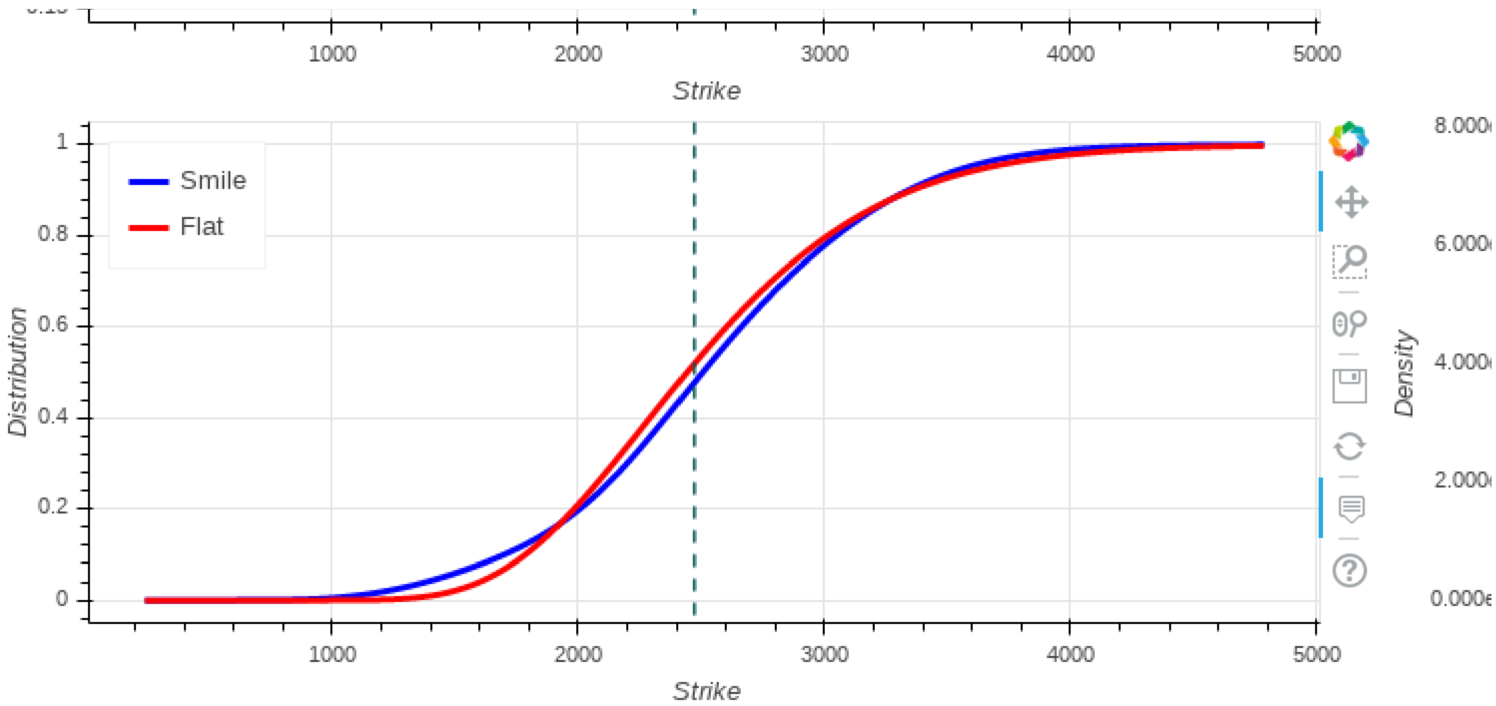
The first term is the model-based LogNormal distribution. The second term is the adjustment due to the volatility smile, and an ill-behaved functional form could result in $Q \notin [0,1]$. Extrapolating too much skew $\frac{d\Sigma}{dK}$, or oscillations when interpolating, can make the second term dominate.

Similar comments apply for the implied density when computing $q = \frac{d^2 p}{dK^2}$.

In [38]:
```
interactive_smile_density(np.arange(10.0, 200.0, 1.0) * fwd_hat / 100.0, 1.0,
fwd_hat, pv_hat, fwd_hat, 0.20, -0.26, 0.72, 0.33)
```

BokehJS 0.13.0 successfully loaded.





Implied Vs Realised Volatility

Assume the realised volatility of asset prices is σ so that $dS = \sigma S dW$.

Assume implied volatility is $\Sigma \neq \sigma$, so the option price $V(S,t)$ follows the PDE

$$V_t + \tfrac{1}{2} V_{SS} \Sigma^2 S^2 = 0$$

The dynamics of the option price are governed by the SDE, obtained via Ito's lemma

$$dV = V_t dt + V_S dS + \frac{1}{2} V_{SS} \sigma^2 S^2 dt$$

Thus a delta-hedged option portfolio $d\Pi = dV - V_S dS$ is governed by

$$d\Pi = \frac{1}{2} V_{SS} \left(\sigma^2 - \Sigma^2 \right) S^2 dt$$

Quiz 4: Monetising Implied Volatility

Implied volatility Σ exceeds realised volatility σ.

Select the correct statement.

1. Delta-hedging long calls is a profitable strategy.
2. Delta-hedging short call spreads results in profits when spot is high.

3. Delta-hedging short put options accrues a positive constant daily profit.
4. Delta-hedging short call options accrues a positive stochastic daily profit.

Thank you for your attention!