Advanced_Derivatives_Bloomberg

December 23, 2021

Advanced Derivatives Final Assignment - Bloomberg excersie

Team Members:

Marcel Santos de Carvalho, id 79803

Alex Palacios, id 73713

Loris Baudry, id 79794

Question 8 - Bloomberg Pricing - One Touch Option

For this question, we are going to value a One Touch option on the VIXY ETF that replicates the CBOE Volatility Index on the S&P500. The option we price was valued at December 1st 2021 when the spot price on the VIXY was $19.15. The parameters we took for our option are the following:

1. It is an Up and In, Pay on Hit option (which mean that if the VIXY price goes above the stablished barrier, it will pay a prestlabished payoff)
2. The direction of the option is receiver
3. The barrier value was set to $22.00
4. The maturity of the option is 2 years
5. The prestablished payoff is $1.00 per option
6. We based the pricing on the Black & Scholes model, using the implied volatility provided by Bloomberg
7. The discount curve is based on the 3 month Libor curve
8. There is an all in Dividend Yield & Management fee rate of 0.850%

Except from the barrier level and the maturity of the option we used the inputs suggested by Bloomberg. To explain the motivation behind the barrier level, you can see below a graph of the historical price on the VIXY. As we can see, our proposed option would have resulted in a positive net payoff as the price went above the barrier in the following days. This type of option has the potential to partially hedge the downside risk on the market exposure as normally the VIX level goes up whenever there is a sharp fall in the S&P 500.

[2]:
```
import pandas as pd
import numpy as np
import scipy.stats as si
import matplotlib.pyplot as plt
import math
```

```python
from financepy.utils import *
from financepy.products.equity import *
```

```
####################################################################
# FINANCEPY BETA Version 0.204 - This build:  02 Nov 2021 at 10:17 #
#      This software is distributed FREE & WITHOUT ANY WARRANTY    #
#  Report bugs as issues at https://github.com/domokane/FinancePy  #
####################################################################
```

[5]:
```python
# Here we upload the documents we will use

from PIL import Image
img_sel_curve = Image.open('One_Touch/Curve_Selection.png')
img_curve = Image.open('One_Touch/Curve.png')
img_Pricing = Image.open('One_Touch/Pricing.png')
img_vol_g = Image.open('One_Touch/Volatility_Grid.png')
img_price = Image.open('One_Touch/VIXY_price.jpeg')

curve = pd.read_excel('IRS Zero_Discount_Curve One Touch.xlsx', index_col=0,␣
 ↪header=0, na_filter=True)
zeros = curve['Zero Rate']
discount = curve['Discount']
market_r = curve['Market Rate']
```

[6]:
```python
img_price
```

[6]:

Below, you can see the Bloomber pricing Dashboard using the OVME option for equity securities.

We specified the type of option to be a One Touch option with the parameters described above.

[7]:
```
img_Pricing
```

[7]:

Once on the dashboard it is possible to get information regarding the discount curve used to price the option and the implied volatility grid used to do so. Below, you can see both screenshots.

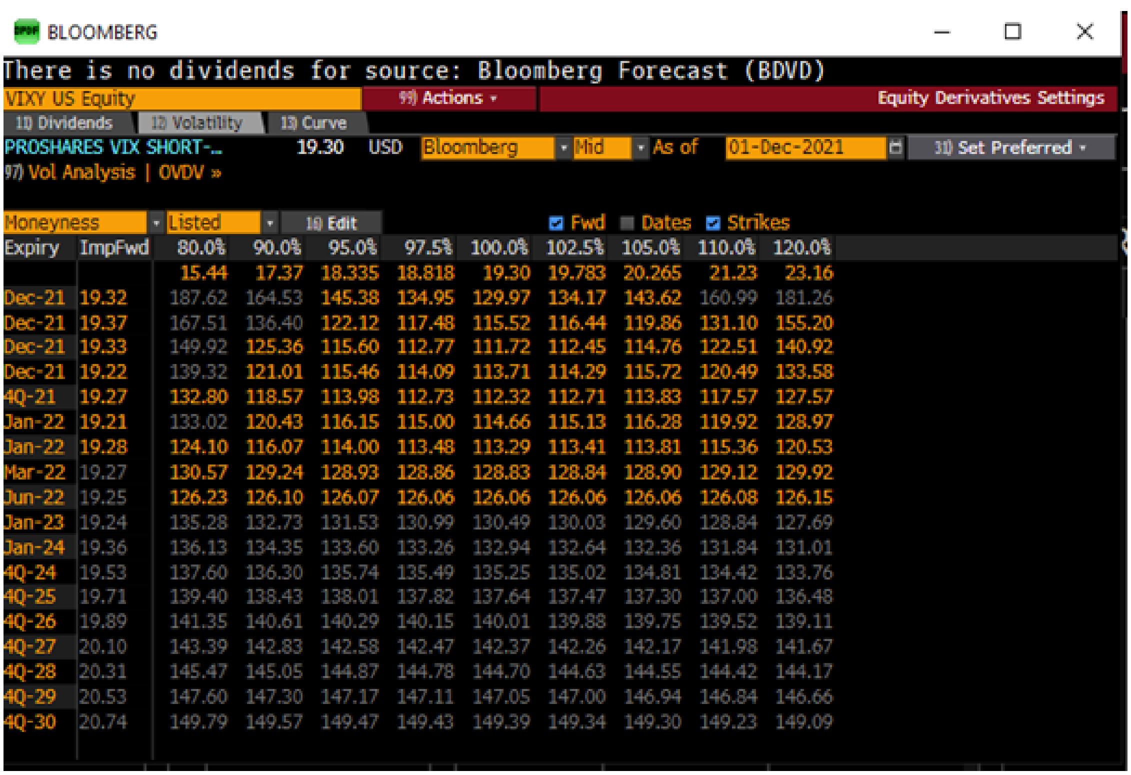
[8]:
```
img_sel_curve
```

[8]:

S-Curve		I-Curve	
S23	USD (30/360, S/A vs. 3M LIBOR)	I25	US Treasury Actives
S42	USD OIS	I26	US Agencies
S47	USD (ACT360, Ann vs. 3M LIBOR)	I27	Supranational Global
S168	USD CDSW Fixing	I39	US Sovereign Strips
S171	USD Qrtrly tenors (vs. 3M LIBOR)	I52	US Dollar Swap Rates
S260	USD ISDA Standard Rates	I111	US On/Off The Run Sov
S349	USD (vs. 12M LIBOR)	I168	US Fannie Mae Bench
S357	USD MUNI (vs. % LIBOR)	I169	UST Inflation Index
S478	Onsh. CNY USD Deposit	I197	US Freddie Mac Notes
S490	USD SOFR (vs. FIXED RATE)	I199	FHLB ISSUES
S528	USD (vs. 3M BSBY)	I205	US Swap Rates Act/360
S548	USD (vs. 1M BSBY)	I211	USD Brazil Sovereign
		I213	USD Mexico Sovereign
		I214	USD Russia Sovereign
		I248	USD Colombia Sov
		I249	USD Turkey Sovereign
		I252	BMA-FNMA Benchmark

[9]:
```
img_vol_g
```

[9]:

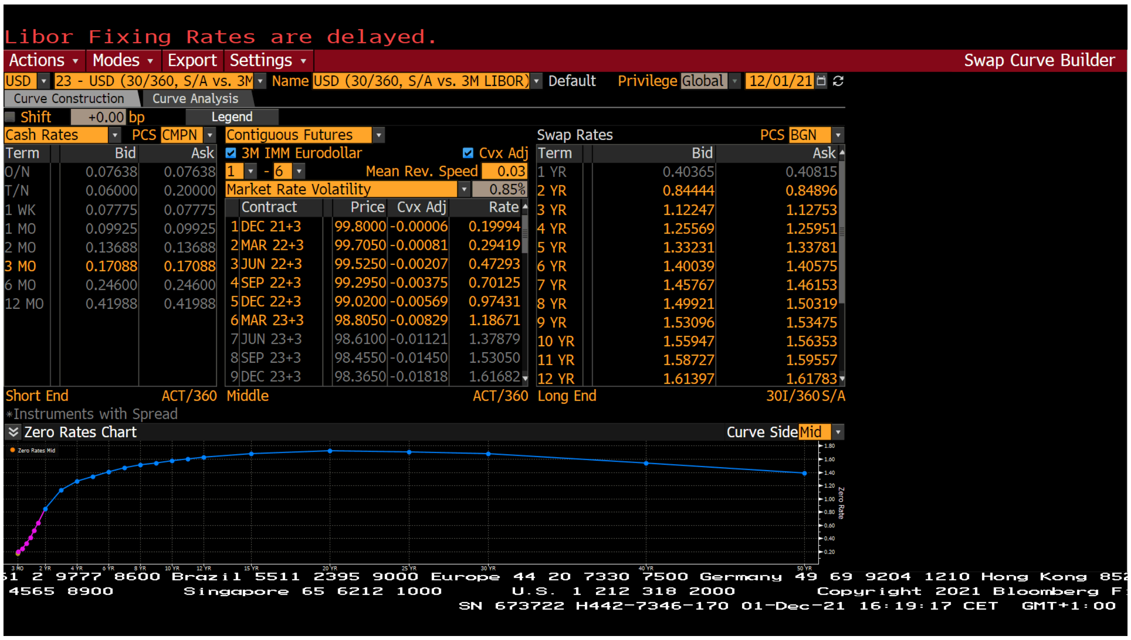


Expiry	ImpFwd	80.0%	90.0%	95.0%	97.5%	100.0%	102.5%	105.0%	110.0%	120.0%
		15.44	17.37	18.335	18.818	19.30	19.783	20.265	21.23	23.16
Dec-21	19.32	187.62	164.53	145.38	134.95	129.97	134.17	143.62	160.99	181.26
Dec-21	19.37	167.51	136.40	122.12	117.48	115.52	116.44	119.86	131.10	155.20
Dec-21	19.33	149.92	125.36	115.60	112.77	111.72	112.45	114.76	122.51	140.92
Dec-21	19.22	139.32	121.01	115.46	114.09	113.71	114.29	115.72	120.49	133.58
4Q-21	19.27	132.80	118.57	113.98	112.73	112.32	112.71	113.83	117.57	127.57
Jan-22	19.21	133.02	120.43	116.15	115.00	114.66	115.13	116.28	119.92	128.97
Jan-22	19.28	124.10	116.07	114.00	113.48	113.29	113.41	113.81	115.36	120.53
Mar-22	19.27	130.57	129.24	128.93	128.86	128.83	128.84	128.90	129.12	129.92
Jun-22	19.25	126.23	126.10	126.07	126.06	126.06	126.06	126.06	126.08	126.15
Jan-23	19.24	135.28	132.73	131.53	130.99	130.49	130.03	129.60	128.84	127.69
Jan-24	19.36	136.13	134.35	133.60	133.26	132.94	132.64	132.36	131.84	131.01
4Q-24	19.53	137.60	136.30	135.74	135.49	135.25	135.02	134.81	134.42	133.76
4Q-25	19.71	139.40	138.43	138.01	137.82	137.64	137.47	137.30	137.00	136.48
4Q-26	19.89	141.35	140.61	140.29	140.15	140.01	139.88	139.75	139.52	139.11
4Q-27	20.10	143.39	142.83	142.58	142.47	142.37	142.26	142.17	141.98	141.67
4Q-28	20.31	145.47	145.05	144.87	144.78	144.70	144.63	144.55	144.42	144.17
4Q-29	20.53	147.60	147.30	147.17	147.11	147.05	147.00	146.94	146.84	146.66
4Q-30	20.74	149.79	149.57	149.47	149.43	149.39	149.34	149.30	149.23	149.09

Finally, we look for a more complete information about the discount curve that we identified that was used to price the option. Below, you can see both the Bloomberg Curve dashboard and a table with the corresponding values.

[10]:
```
img_curve
```

[10]:



[11]:
```
curve
```

[11]:

Maturity Date	Market Rate	Shift (bp)	Shifted Rate	Zero Rate	Discount	\
03/03/2022	0.170880	0	0.170880	0.173216	0.999573	
03/16/2022	0.199938	0	0.199938	0.199237	0.999438	
06/15/2022	0.294224	0	0.294224	0.245657	0.998695	
09/21/2022	0.478013	0	0.478013	0.325762	0.997397	
12/21/2022	0.701390	0	0.701390	0.417175	0.995632	
03/15/2023	0.979530	0	0.979530	0.520570	0.993362	
06/21/2023	1.187029	0	1.187029	0.638691	0.990162	
12/04/2023	0.847250	0	0.847250	0.847811	0.983164	
12/03/2024	1.125300	0	1.125300	1.126475	0.966741	
12/03/2025	1.257500	0	1.257500	1.259775	0.950825	
12/03/2026	1.335300	0	1.335300	1.338288	0.935241	
12/03/2027	1.404500	0	1.404500	1.408619	0.918920	
12/04/2028	1.460700	0	1.460700	1.465431	0.902397	
12/03/2029	1.502680	0	1.502680	1.508413	0.886250	

```
12/03/2030       1.535400         0      1.535400   1.541962  0.870349
12/03/2031       1.564650         0      1.564650   1.572142  0.854447
12/03/2032       1.593270         0      1.593270   1.601563  0.838363
12/05/2033       1.618100         0      1.618100   1.627564  0.822398
12/03/2036       1.669500         0      1.669500   1.681130  0.776970
12/03/2041       1.712600         0      1.712600   1.725645  0.707962
12/03/2046       1.702300         0      1.702300   1.709015  0.652115
12/04/2051       1.679400         0      1.679400   1.677971  0.604255
12/05/2061       1.574790         0      1.574790   1.542322  0.539326
12/03/2071       1.452710         0      1.452710   1.385793  0.499898

                      Source
Maturity Date
03/03/2022              CASH
03/16/2022            FUTURE
06/15/2022            FUTURE
09/21/2022            FUTURE
12/21/2022            FUTURE
03/15/2023            FUTURE
06/21/2023            FUTURE
12/04/2023      DETAILED_SWAP
12/03/2024      DETAILED_SWAP
12/03/2025      DETAILED_SWAP
12/03/2026      DETAILED_SWAP
12/03/2027      DETAILED_SWAP
12/04/2028      DETAILED_SWAP
12/03/2029      DETAILED_SWAP
12/03/2030      DETAILED_SWAP
12/03/2031      DETAILED_SWAP
12/03/2032      DETAILED_SWAP
12/05/2033      DETAILED_SWAP
12/03/2036      DETAILED_SWAP
12/03/2041      DETAILED_SWAP
12/03/2046      DETAILED_SWAP
12/04/2051      DETAILED_SWAP
12/05/2061      DETAILED_SWAP
12/03/2071      DETAILED_SWAP
```

From the table above, we can see that Bloomber already constructs a Zero coupon bond rate derived from the market rates. This is very useful when pricing cashflows in the future, as each cashflow has its corresponding discount rate. To price the chosen option in financpy, we will use both the zero discount curve and a flat curve to analyse the differences.

[12]:
```
# We will import the missing classes from financepy to value an One Touch Option

from financepy.market.curves.discount_curve_flat import DiscountCurveFlat
```

[13]:
```
# We define the parameters for the valuation

valuation_date = Date(1, 12, 2021)
expiry_date = Date(1, 12, 2023)
interest_rate = 0.00828
dividend_yield = 0.00850
discount_curve = DiscountCurveFlat(valuation_date, interest_rate)
dividend_curve = DiscountCurveFlat(valuation_date, dividend_yield)
volatility = 1.32609
barrier_level = 22.0
model = BlackScholes(volatility)
stock_price = 19.15
payment_size = 1.0
```

[14]:
```
# Here, we create the option object and value the option assuming a flat␣
↪discount curve

downTypes = [FinTouchOptionPayoffTypes.UP_AND_IN_CASH_AT_HIT]
print("%60s %12s %12s" % ("Option Type", "Analytical", "Monte Carlo"))
for downType in downTypes:
    option =␣
↪EquityOneTouchOption(expiry_date,downType,barrier_level,payment_size)
    v = option.
↪value(valuation_date,stock_price,discount_curve,dividend_curve,model)
    v_mc = option.
↪value_mc(valuation_date,stock_price,discount_curve,dividend_curve,model)
    print("%60s %12.5f %12.5f" % (downType, v, v_mc))
```

```
                                                 Option Type   Analytical  Monte
Carlo
             FinTouchOptionPayoffTypes.UP_AND_IN_CASH_AT_HIT      0.85681
0.81258
```

Now, we will repeat the valuation but using the actual discount curve. To do so, we will construct the discount curve starting from the observed market deposit and swap rates.

[15]:
```
from financepy.products.rates import *
```

[16]:
```
# We use market convention T+2 for settlement dates
spot_days = 2
settlement_date = valuation_date.add_days(spot_days)
```

Ibor Deposits

[17]:
```
# We input the deposit rates we obtained from Bloomberg

dcType = DayCountTypes.ACT_360
depo1 = IborDeposit(settlement_date, '3M', market_r[1]/100, dcType)
```

```
depo2 = IborDeposit(settlement_date, '6M', market_r[2]/100, dcType)
depo3 = IborDeposit(settlement_date, '9M', market_r[3]/100, dcType)
depo4 = IborDeposit(settlement_date, '12M', market_r[4]/100, dcType)
depo5 = IborDeposit(settlement_date, '15M', market_r[5]/100, dcType)
depo6 = IborDeposit(settlement_date, '18M', market_r[6]/100, dcType)
depos = [depo1,depo2,depo3,depo4,depo5,depo6]
dcType = DayCountTypes.THIRTY_E_360_ISDA
fixedFreq = FrequencyTypes.SEMI_ANNUAL
```

Interest Rate Swaps

[18]:
```
swapType = SwapTypes.PAY
swap1 = IborSwap(settlement_date,"2Y",swapType,market_r[7]/100,fixedFreq,dcType)
swap2 = IborSwap(settlement_date,"3Y",swapType,market_r[8]/100,fixedFreq,dcType)
swap3 = IborSwap(settlement_date,"4Y",swapType,market_r[9]/100,fixedFreq,dcType)
swap4 = IborSwap(settlement_date,"5Y",swapType,market_r[10]/
 ↪100,fixedFreq,dcType)
swap5 = IborSwap(settlement_date,"6Y",swapType,market_r[11]/
 ↪100,fixedFreq,dcType)
swap6 = IborSwap(settlement_date,"7Y",swapType,market_r[12]/
 ↪100,fixedFreq,dcType)
swap7 = IborSwap(settlement_date,"8Y",swapType,market_r[13]/
 ↪100,fixedFreq,dcType)
swap8 = IborSwap(settlement_date,"9Y",swapType,market_r[14]/
 ↪100,fixedFreq,dcType)
swap9 = IborSwap(settlement_date,"10Y",swapType,market_r[15]/
 ↪100,fixedFreq,dcType)
swap10 = IborSwap(settlement_date,"11Y",swapType,market_r[16]/
 ↪100,fixedFreq,dcType)
swap11 = IborSwap(settlement_date,"12Y",swapType,market_r[17]/
 ↪100,fixedFreq,dcType)
swap12 = IborSwap(settlement_date,"15Y",swapType,market_r[18]/
 ↪100,fixedFreq,dcType)
swap13 = IborSwap(settlement_date,"20Y",swapType,market_r[19]/
 ↪100,fixedFreq,dcType)
swap14 = IborSwap(settlement_date,"25Y",swapType,market_r[20]/
 ↪100,fixedFreq,dcType)
swap15 = IborSwap(settlement_date,"30Y",swapType,market_r[21]/
 ↪100,fixedFreq,dcType)
swap16 = IborSwap(settlement_date,"40Y",swapType,market_r[22]/
 ↪100,fixedFreq,dcType)
swap17 = IborSwap(settlement_date,"50Y",swapType,market_r[23]/
 ↪100,fixedFreq,dcType)
```

[19]:
```
swaps = [swap1,swap2,swap3,swap4,swap5,swap6,swap7,
         swap8,swap9,swap10,swap11,swap12,swap13,swap14,swap15,swap16,swap17]
```

[20]:
```
# From the rates specified above, we derive the implict FRAs
# We prepare the list to store the values

fras = []
```

Bootstrapping The Curve

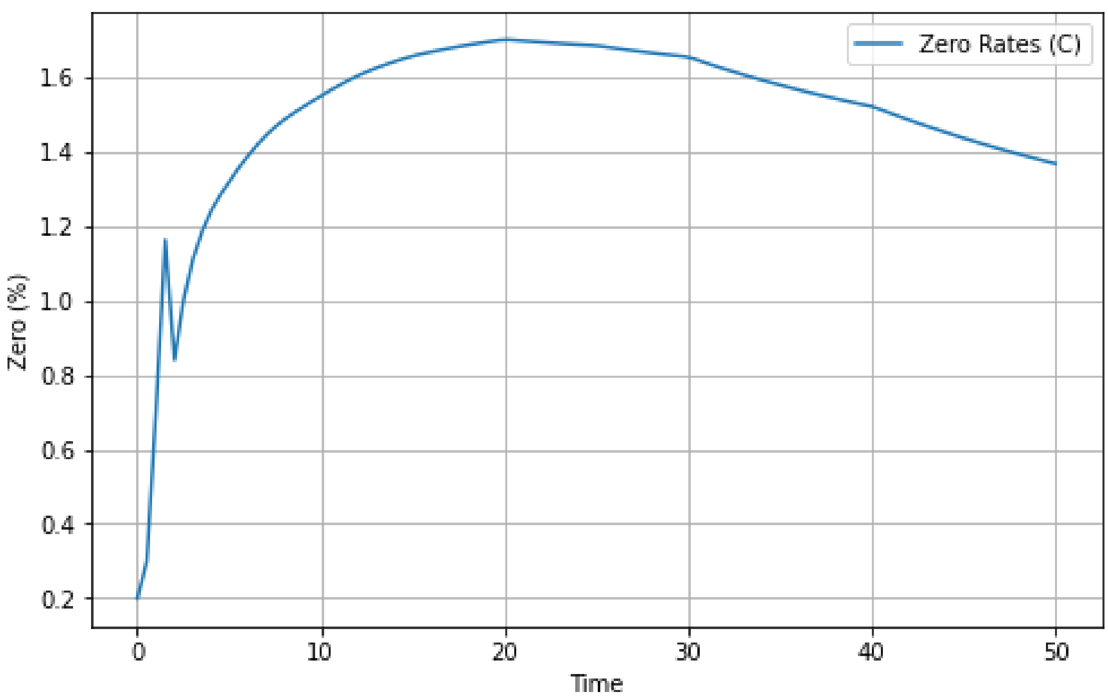
[21]:
```
libor_curve = IborSingleCurve(valuation_date, depos, fras, swaps)
```

Extracting the Zero Rate Curves

We choose a range of zero rate frequencies.

[22]:
```
years = np.linspace(1/365,50,100)
dates = settlement_date.add_years(years)
zerosC = libor_curve.zero_rate(dates, FrequencyTypes.CONTINUOUS)
```

[23]:
```
plt.figure(figsize=(8,5))
plt.plot(years,zerosC*100, label="Zero Rates (C)")
plt.xlabel("Time")
plt.ylabel("Zero (%)")
plt.legend()
plt.grid()
```



```
[24]: zeros_D_F = libor_curve.df(dates,dcType)
discount_curve_2 = DiscountCurve(valuation_date, dates, zeros_D_F)
```

```
[25]: downTypes = [FinTouchOptionPayoffTypes.UP_AND_IN_CASH_AT_HIT]
print("%60s %12s %12s" % ("Option Type", "Analytical", "Monte Carlo"))
for downType in downTypes:
    option2 =␣
 ↪EquityOneTouchOption(expiry_date,downType,barrier_level,payment_size)
    v2 = option2.
 ↪value(valuation_date,stock_price,discount_curve_2,dividend_curve,model)
    v2_mc = option2.
 ↪value_mc(valuation_date,stock_price,discount_curve_2,dividend_curve,model)
    print("%60s %12.5f %12.5f" % (downType, v2, v2_mc))
```

```
                                                 Option Type   Analytical  Monte
Carlo
             FinTouchOptionPayoffTypes.UP_AND_IN_CASH_AT_HIT      0.85682
0.81259
```

The analytical model produced the following sensitivities

```
[26]: option2.delta(valuation_date, stock_price, discount_curve_2, dividend_curve,␣
 ↪model)
```

```
[26]: 0.05018453356098185
```

```
[27]: option2.gamma(valuation_date, stock_price, discount_curve, dividend_curve,␣
 ↪model)
```

```
[27]: 4.9671378121729504e-05
```

```
[28]: option2.theta(valuation_date, stock_price, discount_curve, dividend_curve,␣
 ↪model)
```

```
[28]: -0.008710353621571021
```

```
[29]: option2.vega(valuation_date, stock_price, discount_curve, dividend_curve, model)
```

```
[29]: 0.027342039207178814
```

```
[30]: option2.rho(valuation_date, stock_price, discount_curve, dividend_curve, model)
```

```
[30]: 0.024778375415923648
```

```
[31]: Results = pd.DataFrame(index=[valuation_date],columns=["Bloomberg Price", "Flat␣
 ↪Curve Price", "Fitted Curve Price"])
Results.iloc[0,0]= 0.85760
Results.iloc[0,1]= np.round(v,5)
Results.iloc[0,2]=np.round(v2,5)
```

```
Results
```

[31]:

	Bloomberg Price	Flat Curve Price	Fitted Curve Price
01-DEC-2021	0.8576	0.85681	0.85682

As we can see, the prices of both the flat curve and the fitted curve are pretty close to that obtained from Bloomberg. This makes sense as we took the sames inputs. Nonetheless, there might be some structural differences, specially in the way that dividends are being discounted. We use a flat curve to discount the dividends assuming that the dividend rate provided in the bloomberg pricing page was the continous dividend rate. Both assumptions might be having an impact regarding our pricing using the financepy in-built functions.