In [1]:
```python
import numpy as np
import pandas as pd
from math import sqrt
```

In [2]:
```python
data = pd.read_excel('ZCB_yield.xlsx')
```

In [3]:
```python
data
```

Out[3]:

	years	yield
0	0.0833	0.0239
1	0.1667	0.0241
2	0.2500	0.0242
3	0.5000	0.0251
4	1.0000	0.0260
5	2.0000	0.0256
6	3.0000	0.0254
7	5.0000	0.0255
8	7.0000	0.0261
9	10.0000	0.0272
10	20.0000	0.0290
11	30.0000	0.0304

In [4]:
```python
def first_spline(x_):

    if np.sum(np.isnan(x_))==1:
        return 0
    else:
        x, y = map(np.asarray, (data['years'], data['yield']))
        # number of S_i(x)
        n = len(x) - 1
        a = np.empty((n,2))
        a[:,1] = np.diff(y)/np.diff(x)
        a[:,0] = y[:-1]-a[:,1]*x[:-1]

        index_ = np.where((x>x_)==True)[0][0]-1
        return a[index_,0]+x_*a[index_,1]
```

In [5]:
```python
def second_spline(x_):

    if np.sum(np.isnan(x_))==1:
        return 0
    else:
        x, y = map(np.asarray, (data['years'], data['yield']))
        # number of S_i(x)
        n = len(x) - 1
        a = np.empty((n,3))
        dydx = np.diff(y)/np.diff(x)

        matrix = np.eye(n+1, n, k=-1)
        matrix[:-1, :] = matrix[:-1, :]+np.eye(n)
```

```
        delta = np.dot(dydx,np.linalg.pinv(matrix))

        a[:,2] = np.diff(delta)/2/np.diff(x)
        a[:,1] = delta[:-1]
        a[:,0] = 1

        index_ = np.where((x>x_)==True)[0][0]-1

        return y[index_]*a[index_,0]+a[index_,1]*(x_-x[index_])+a[index_,2]*((x_-x
```

In [6]:
```
def cubic_interp1d(x0,x,y):

    x = np.asfarray(x)
    y = np.asfarray(y)

    if np.any(np.diff(x) < 0):
        indexes = np.argsort(x)
        x = x[indexes]
        y = y[indexes]

    size = len(x)

    xdiff = np.diff(x)
    ydiff = np.diff(y)

    # allocate buffer matrices
    Li = np.empty(size)
    Li_1 = np.empty(size-1)
    z = np.empty(size)

    # fill diagonals Li and Li-1 and solve [L][y] = [B]
    Li[0] = sqrt(2*xdiff[0])
    Li_1[0] = 0.0
    B0 = 0.0 # natural boundary
    z[0] = B0 / Li[0]

    for i in range(1, size-1, 1):
        Li_1[i] = xdiff[i-1] / Li[i-1]
        Li[i] = sqrt(2*(xdiff[i-1]+xdiff[i]) - Li_1[i-1] * Li_1[i-1])
        Bi = 6*(ydiff[i]/xdiff[i] - ydiff[i-1]/xdiff[i-1])
        z[i] = (Bi - Li_1[i-1]*z[i-1])/Li[i]

    i = size - 1
    Li_1[i-1] = xdiff[-1] / Li[i-1]
    Li[i] = sqrt(2*xdiff[-1] - Li_1[i-1] * Li_1[i-1])
    Bi = 0.0 # natural boundary
    z[i] = (Bi - Li_1[i-1]*z[i-1])/Li[i]

    # solve [L.T][x] = [y]
    i = size-1
    z[i] = z[i] / Li[i]
    for i in range(size-2, -1, -1):
        z[i] = (z[i] - Li_1[i-1]*z[i+1])/Li[i]

    # find index
    index = x.searchsorted(x0)
    np.clip(index, 1, size-1, index)

    xi1, xi0 = x[index], x[index-1]
    yi1, yi0 = y[index], y[index-1]
    zi1, zi0 = z[index], z[index-1]
    hi1 = xi1 - xi0
```

```python
    # calculate cubic
    f0 = zi0/(6*hi1)*(xi1-x0)**3 + \
         zi1/(6*hi1)*(x0-xi0)**3 + \
         (yi1/hi1 - zi1*hi1/6)*(x0-xi0) + \
         (yi0/hi1 - zi0*hi1/6)*(xi1-x0)
    return f0
```

In [7]:

```python
from pandas.tseries.offsets import DateOffset

start_date = '2019-04-15'
end_date = '2028-11-15'

date_range = pd.date_range(start=start_date, end=end_date, freq='6MS', inclusive='
coupondates = date_range+DateOffset(days=14)

bond_df = pd.DataFrame(3.125/2,index = coupondates,columns = ['coupon'])
bond_df.loc['2028-11-15'] = bond_df.loc['2028-11-15'] +100
bond_df['days_diff'] = bond_df.index.to_series()-pd.to_datetime('2019-01-19')
bond_df['years'] = bond_df['days_diff'].dt.days/365


bond_df['yield_1'] = bond_df['years'].apply(first_spline)
bond_df['pv_1'] = bond_df['coupon']/((1+bond_df['yield_1'])**bond_df['years'])


bond_df['yield_2'] = bond_df['years'].apply(second_spline)
bond_df['pv_2'] = bond_df['coupon']/((1+bond_df['yield_2'])**bond_df['years'])

bond_df['yield_3'] = cubic_interp1d(np.array(bond_df['years']), data['years'], data
bond_df['pv_3'] = bond_df['coupon']/((1+bond_df['yield_3'])**bond_df['years'])
bond_df
```

Out[7]:

	coupon	days_diff	years	yield_1	pv_1	yield_2	pv_2	yield_3	pv_
2019-05-15	1.5625	116 days	0.317808	0.024444	1.550554	0.024300	1.550623	0.024363	1.55059
2019-11-15	1.5625	300 days	0.821918	0.025679	1.530275	0.025543	1.530442	0.025850	1.53006
2020-05-15	1.5625	482 days	1.320548	0.025872	1.510675	0.025884	1.510652	0.026040	1.51034
2020-11-15	1.5625	666 days	1.824658	0.025670	1.491883	0.025801	1.491537	0.025729	1.49172
2021-05-15	1.5625	847 days	2.320548	0.025536	1.473696	0.025598	1.473489	0.025479	1.47388
2021-11-15	1.5625	1031 days	2.824658	0.025435	1.455487	0.025538	1.455075	0.025408	1.45559
2022-05-15	1.5625	1212 days	3.320548	0.025416	1.437559	0.025337	1.437927	0.025392	1.43767
2022-11-15	1.5625	1396 days	3.824658	0.025441	1.419352	0.025293	1.420138	0.025391	1.41961
2023-05-15	1.5625	1577 days	4.320548	0.025466	1.401632	0.025315	1.402527	0.025415	1.40193
2023-11-15	1.5625	1761 days	4.824658	0.025491	1.383812	0.025403	1.384384	0.025471	1.38394
2024-05-15	1.5625	1943 days	5.323288	0.025597	1.365802	0.025586	1.365879	0.025566	1.36602
2024-11-15	1.5625	2127 days	5.827397	0.025748	1.347352	0.025692	1.347786	0.025700	1.34772
2025-05-15	1.5625	2308 days	6.323288	0.025897	1.329253	0.025761	1.330371	0.025859	1.32956
2025-11-15	1.5625	2492 days	6.827397	0.026048	1.310911	0.025796	1.313113	0.026037	1.31100
2026-05-15	1.5625	2673 days	7.323288	0.026219	1.292729	0.026110	1.293734	0.026223	1.29268
2026-11-15	1.5625	2857 days	7.827397	0.026403	1.274175	0.026148	1.276654	0.026415	1.27406
2027-05-15	1.5625	3038 days	8.323288	0.026585	1.255961	0.026215	1.259735	0.026603	1.25577
2027-11-15	1.5625	3222 days	8.827397	0.026770	1.237490	0.026312	1.242374	0.026790	1.23727
2028-05-15	1.5625	3404 days	9.326027	0.026953	1.219269	0.026436	1.225003	0.026970	1.21907
2028-11-15	101.5625	3588 days	9.830137	0.027138	78.058804	0.026591	78.468221	0.027144	78.05432

In [8]:

```
print('Cubic:',sum(bond_df['pv_3']))
print('Second:',sum(bond_df['pv_2']))
print('First:',sum(bond_df['pv_1']))
```

```
Cubic: 104.34290452180281
Second: 104.77966233454511
First: 104.346670171326
```

In [10]:
```
bond_df.to_excel('data.xlsx')
```

In [ ]:
```

```