In [84]:
```
#Import required libraries#
import pandas as pd
from dateutil.relativedelta import relativedelta
from dateutil import parser
from pandas.tseries.offsets import Day
from Cashflows_Generator import CashflowGenerator

import numpy as np
import matplotlib.pyplot as plt
import datetime
from math import exp
import os

from scipy.interpolate import interp1d
```

Create an instance of CashflowGenerator to generate the cashflows for a fixed leg and a floating leg of a swap contract. The generate_cashflows() method is called on each instance, and the resulting cashflows are stored in fixed_leg_df and float_leg_df variables. These variables are then printed to the console.

In [ ]:
```
##fixed_generator = CashflowGenerator('First Payment Date', 'Maturity', Month_freq, Do
fixed_generator = CashflowGenerator('2023-06-09', '2028-03-09', 3, 'act/360' , effecti
fixed_leg_df = fixed_generator.generate_cashflows()
print(fixed_leg_df)

float_generator = CashflowGenerator('2023-06-09', '2028-03-09', 3, 'act/360' , effecti
float_leg_df = float_generator.generate_cashflows()
print(float_leg_df)
```

In [ ]:
```
## DataFrame of zero rates for different maturities. This data is stored in df_zero.##
df_zero = pd.DataFrame({
    'date': ['2023-03-09', '2023-06-09', '2023-09-09', '2023-12-09',
             '2024-03-09', '2024-06-09', '2024-09-09', '2024-12-09',
             '2025-03-09', '2025-06-09', '2025-09-09', '2025-12-09',
             '2026-03-09', '2026-06-09', '2026-09-09', '2026-12-09',
             '2027-03-09', '2027-06-09', '2027-09-09', '2027-12-09',
             '2028-03-09', '2028-06-09', '2028-09-09', '2028-12-09'],
    'rate': [0.00147746193495074, 0.00144337757980778,
             0.00166389741542625, 0.00175294804717070, 0.00196071374597585,
             0.00224582504806747, 0.00264462838911974, 0.00328408008984121,
             0.00571530169527018, 0.00795496282359075, 0.00970003866673104,
             0.01113416387898720, 0.01229010329346910, 0.01320660291639990,
             0.01396222829363160, 0.01461391064905110, 0.01518876914165160,
             0.015673596204295
```

Define parameters of interest rate swap and interpolated zero rate curve

In [ ]:
```
# 2. Libor Swap Specification
#-------------------------

spot_date= pd.to_datetime('2023-03-09')   # spot date

no_amt = 1000000      # notional amount
fixed_rate = 0.07
```

```
spot_date_ymd = '2023-03-09'
spot_date = pd.to_datetime(spot_date_ymd).toordinal()

df_zero['date'] = pd.to_datetime(df_zero['date']).apply(lambda x: x.toordinal())
v_date = df_zero['date'].values
v_zero = df_zero['rate'].values
f_linear = np.interp(spot_date, v_date, v_zero)
v_date_inter = np.arange(spot_date, v_date[-1]+1)
v_zero_inter = np.interp(v_date_inter, v_date, v_zero)

# Convert ordinal dates to datetime objects
dates = [datetime.date.fromordinal(d) for d in v_date]
inter_dates = [datetime.date.fromordinal(d) for d in v_date_inter]

# Figures for zero curve
plt.figure(figsize=(6, 5))
plt.plot(dates, v_zero, 'bo-', markersize=8, label='market zero rate')
plt.plot(inter_dates, v_zero_inter, 'r-', linewidth=2.5, label='interpolated zero rate
plt.legend(loc='lower right')
plt.xlabel('date')
plt.ylabel('Rate')
plt.show()

#interpolated zero curve
interpolated = pd.DataFrame()
interpolated['date']  = inter_dates
interpolated['Rates']  = v_zero_inter
interpolated['date'] = pd.to_datetime(interpolated['date'])
```

Append rates matching payment dates to float and fixed leg

In [ ]:
```
#Get correct zero curve rates#
spot_date = pd.to_datetime(spot_date)
float_leg_df['date'] = pd.to_datetime(float_leg_df['date'])
float_leg = pd.merge(float_leg_df, interpolated, on='date', how='inner')

fixed_leg_df['Rates']= fixed_rate
fixed_leg =fixed_leg_df
```

In [ ]:
```
#Calculating Cashflows for legs#
float_leg['df'] = 1 / (1 + float_leg['Rates']) ** (float_leg.index + 1)
float_leg['cf'] = float_leg['Rates']*(no_amt)*float_leg['days']
float_leg['pv'] = float_leg['cf']*float_leg['df']

fixed_leg['df'] = 1 / (1 + fixed_leg['Rates']) ** (fixed_leg.index + 1)
fixed_leg['cf'] = fixed_leg['Rates']*(no_amt)*fixed_leg['days']
fixed_leg['pv'] = fixed_leg['cf']*fixed_leg['df']
```

In [ ]:
```
print(float_leg)
print(fixed_leg)

float_NPV=round(float_leg['pv'].sum(),6)
fixed_NPV=round(fixed_leg['pv'].sum(),6)
NPV=fixed_NPV-float_NPV
print(NPV)
```

```
print(float_NPV)
print(fixed_NPV)
```