In [ ]:

```
import pandas as pd
import numpy as np
from scipy.stats import norm
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.optimize import minimize
boeing=pd.read_csv("Boeing.csv")
```

Question 1

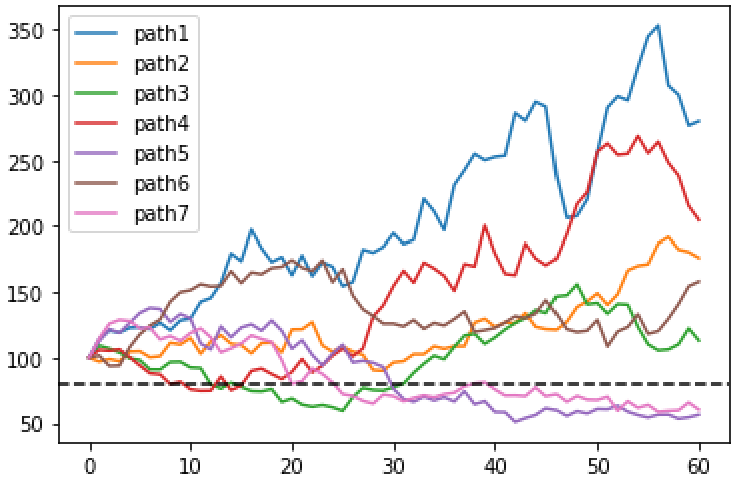
1(a)

In [3]:

```
v0 = 100
mu = 0.06
sigma = 0.25
T = 5
dt = 1/12
N = 80
step = np.arange(0,dt+T,dt)
```

In [4]:

```
np.random.seed(200)
v = np.zeros((7,len(step)))

for j in range(7):
    vt = np.zeros(len(step))
    vt[0] = v0
    wt = np.sqrt(dt) * sigma * np.random.normal(0,1,len(step))

    for i in range(1,len(step)):
        vt[i] = vt[i-1] + dt * mu * vt[i-1] + wt[i] * vt[i-1]
    v[j] = vt
    plt.plot(vt,label='path' + str(j+1))
plt.legend()
plt.axhline(y = N, color = 'black', linestyle = '--')
plt.show()
```



Path 5 and 7 results in default according to the definition of default, because they are below the dash line at year 5

1(b)

In [5]:

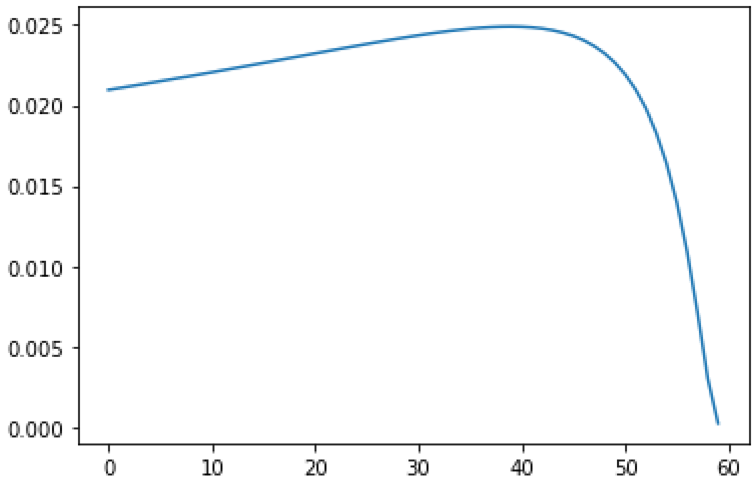
```
r = 0.03
ttm = T-step
```

In [6]:

```
def BMS(S,K,T,r,div,sigma,d_sign):
    d1 = (np.log(S/K) + (r -div + 0.5*sigma **2)*T)/(sigma *np.sqrt(T))
    d2 = d1 -  sigma*np.sqrt(T)
    delta = d_sign*norm.cdf(d_sign*d1)
    price = np.exp(-div*T)*S*delta - d_sign*np.exp(-r*T)*K*norm.cdf(d_sign*d2)
    return price
```

In [9]:

```
spreadt1 = np.zeros(len(step)-1)
for i in range(len(step)-1):
    dt = vt[0] - BMS(vt[0],N,ttm[i],r,0,sigma,1)
    yt = (1/ttm[i])*np.log(N/dt)
    spreadt1[i] = yt-r
plt.plot(spreadt1)
plt.show()
```



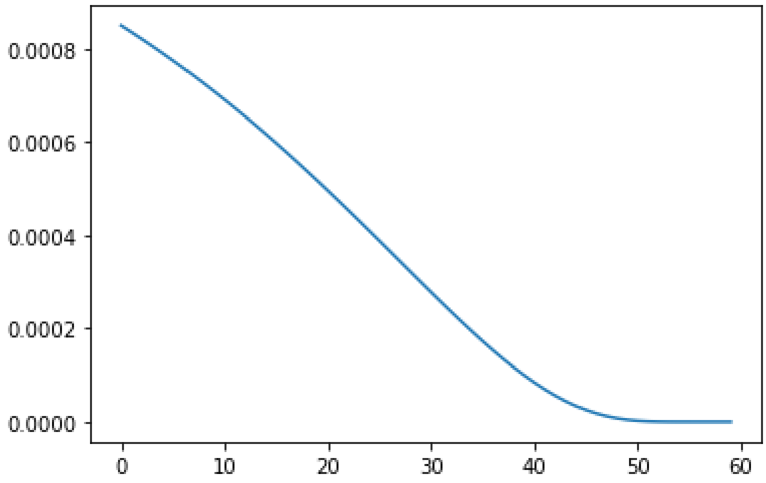
1(c)

In [10]:

```
sigma1 = 0.15
N1 = 60
```

In [11]:

```
spreadt = np.zeros(len(step)-1)
for i in range(len(step)-1):
    #dt = N1*np.exp(-r*ttm[i])-BMS(vt[0],N1,ttm[i],r,0,sigma1,-1)
    dt = vt[0] - BMS(vt[0],N1,ttm[i],r,0,sigma1,1)
    yt = (1/ttm[i])*np.log(N1/dt)
    spreadt[i] = yt-r
plt.plot(spreadt)
plt.show()
```



Compare to the graph from part b, while both credit spreads converge to zero, we can see the term structure of credit spread decrease for lower leverage and volatility.Thus, with a firm improves in credit quality, the the shape of credit spread will be flatter.

1(d)

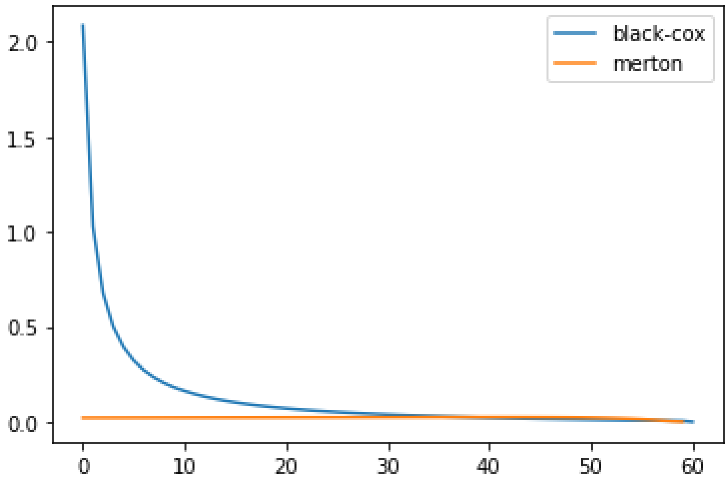
In [12]:

```
dt = 1/12
np.random.seed(200)
path = 1000
rho = 0.9
bankrupt = rho*N
v = np.zeros((path,len(step)))
price = np.zeros(path)

for j in range(path):
    vt = np.zeros(len(step))
    vt[0] = v0
    wt = np.sqrt(dt) * sigma * np.random.normal(0,1,len(step))

    for i in range(1,len(step)):
        vt[i] = vt[i-1] + dt * mu * vt[i-1] + wt[i] * vt[i-1]

    #if default present value of the bankruptcy payoff;
    if (vt<=bankrupt).sum() != 0:
        tau = step[np.where(vt<=bankrupt)[0][0]]
        vtau = vt[np.where(vt<=bankrupt)[0][0]]
        price[j] = np.exp(-r*tau)*vtau
    #no default comparing the asset value at maturity with the face value of debt.
    else:
        price[j] = np.exp(-r*step[-1])*min(vt[-1],N)
    v[j] = vt
print('The expected value of this bond is '+str(np.mean(price)))
```

```
The expected value of this bond is 67.0614175544994
```

In [13]:

```
spreadt = np.zeros(len(step))
for i in range(1,len(step)):
    #dt = np.exp(-r*ttm[i])*min(np.mean(price),N)
    yt = (1/step[i])*np.log(N/np.mean(price))
    spreadt[i-1] = yt-r
plt.plot(spreadt,label='black-cox')
plt.plot(spreadt1,label='merton' )
plt.legend()
plt.show()
```



Both of them converge to zero at maturity. The term structure generated through 1000 simulation seems to be way higher than the one from Merton model

1(e)

In [14]:

```
dt = 1/120
np.random.seed(200)
path = 5000
rho = 0.9
bankrupt = rho*N
step = np.arange(0,dt+T,dt)
v = np.zeros((path,len(step)))
price = np.zeros(path)

for j in range(path):
    vt = np.zeros(len(step))
    vt[0] = v0
    wt = np.sqrt(dt) * sigma * np.random.normal(0,1,len(step))

    for i in range(1,len(step)):
        vt[i] = vt[i-1] + dt * mu * vt[i-1] + wt[i] * vt[i-1]

    #if default present value of the bankruptcy payoff;
    if (vt<=bankrupt).sum() != 0:
        tau = step[np.where(vt<=bankrupt)[0][0]]
        vtau = vt[np.where(vt<=bankrupt)[0][0]]
        price[j] = np.exp(-r*tau)*vtau
    #no default comparing the asset value at maturity with the face value of debt.
    else:
        price[j] = np.exp(-r*step[-1])*min(vt[-1],N)
    v[j] = vt
print('The expected value of this bond is '+str(np.mean(price)))
```

```
The expected value of this bond is 68.099563826346
```

Question 2

In [15]:

```
df = pd.DataFrame({'Date': ['2021-01-29','2021-02-26','2021-03-31','2021-04-30','2021-05-28','20
                           '2021-08-31','2021-09-30','2021-10-29','2021-11-30','2021-12-31'],
      'Share price':[30.08,30.57,26.41,24.37,25.64,23.75,20.89,23.87,21.12,22.24,15.64,15.29],
      'Number of shares':[49175,49175,49574,49574,49667,49667,49667,49892,49892,49892,48880,48880]
      'Return':[-0.07132,0.01629,-0.13608,-0.07724,0.052113,-0.07371,-0.12042,0.142652,-0.11521,0
```

2(a)

In [16]:
```
sigma_e = df['Return'].std()
print('The monthly stock return volatility is ' + str(round(sigma_e,3)))
```

```
The monthly stock return volatility is 0.114
```

2(b)

In [17]:
```
rf = 0.004
N = (28.6 + 0.5 *  757.8)
df['Daily market capitalization'] = df['Share price'] * df['Number of shares'] * 10**-3
sigma = [sigma_e]
T = 1
t = np.arange(0,12/12,1/12)
E_real = df['Daily market capitalization']
```

In [18]:
```
def bs(V):

    d1 = (np.log(V/N) + (rf + 0.5*(sigma_star**2))*(T - t_star))/(sigma_star*np.sqrt((T - t_sta
    d2 = d1 - sigma_star * np.sqrt((T - t_star))
    E = norm.cdf(d1) * V - norm.cdf(d2) * N * np.exp(-rf*(T - t_star))
    diff = np.abs(E - E_real_star)

    return diff
```

In [19]:
```
V_iter = []
sigma_star = sigma[0]

for i in range(12):
    t_star = t[i]
    E_real_star = E_real[i]
    res = minimize(bs,1000)
    V_iter.append(res.x)

V_iter = np.array(V_iter).reshape(12)
df['Temp_V'] = V_iter
sigma.append(np.std(np.log(df['Temp_V']/df['Temp_V'].shift(1))))
```

In [20]:
```
print(df[['Temp_V']])
print('The monthly asset volatility volatility is ' + str(round(sigma[-1],3)))
```

```
         Temp_V
0   1885.057256
1   1909.288319
2   1715.393268
3   1614.397712
4   1679.876661
5   1586.141525
6   1444.229444
7   1597.743439
8   1460.676069
9   1516.690784
10  1171.711624
11  1154.739389
The monthly asset volatility volatility is 0.092
```

2(c)

In [21]:

```
mean = np.mean(np.log(df['Temp_V']/df['Temp_V'].shift(1)))
dd = (1/(sigma[1]*np.sqrt(12)))*(np.log(df['Temp_V'].iloc[-1]/N) + (mean*12 + 0.5*(sigma[1]*np.
PD = 1 - norm.cdf(dd)
```

In [22]:

```
print('The distance to default is ' + str(round(dd,4)))
print('The physical probability of the company is ' + str(round(PD,4)))
```

```
The distance to default is 1.7461
The physical probability of the company is 0.0404
```

Question 3

In [22]:

```
lmda=0.2
R=0.6
T=5
L_bar=0.70
S_0=boeing['S'][0]
```

In [23]:

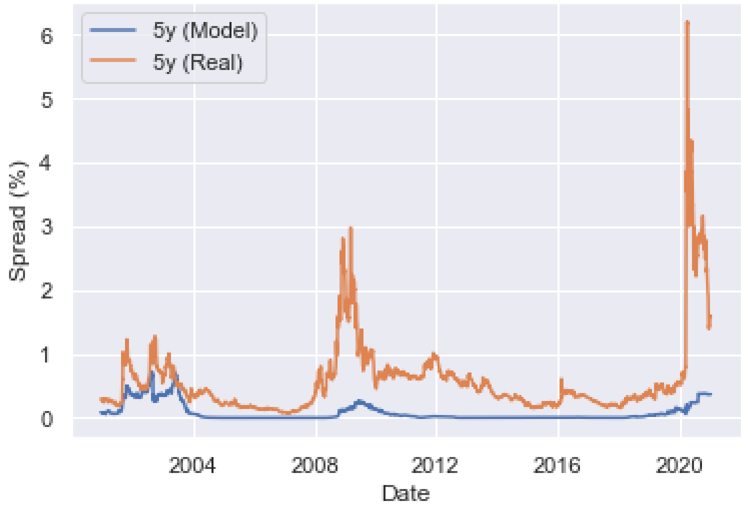
```
boeing['date']=pd.to_datetime(boeing['date'])
boeing['t']=(boeing['date']-boeing['date'][0])
boeing['t']=boeing['t'].dt.days/365
boeing['d']=(S_0+L_bar*boeing['D'])/(L_bar*boeing['D'])*np.exp(lmda**2)
boeing['At']=((boeing['sigma_stock']*boeing['S']/(boeing['S']+L_bar*boeing['D']))**2*boeing['t']
boeing['PS_t']=norm.cdf(-boeing['At']/2+np.log(boeing['d'])/boeing['At'])\
                -boeing['d']*norm.cdf(-boeing['At']/2-np.log(boeing['d'])/boeing['At'])
```

In [24]:

```
PS_0=boeing['PS_t'][0]
def G(row):
    zeta=lmda**2/row['sigma_stock']**2
    z=(1/4+2*row['r']/row['sigma_stock'])**0.5
    g_zeta=row['d']**((z+1)/2)*norm.cdf(-np.log(row['d'])/(row['sigma_stock']*zeta**0.5)-z*(row
            +row['d']**(-(z+1)/2)*norm.cdf(-np.log(row['d'])/(row['sigma_stock']*zeta**0.5)+z*
    g_tzeta=row['d']**((z+1)/2)*norm.cdf(-np.log(row['d'])/(row['sigma_stock']*(5+zeta)**0.5)-z
            +row['d']**(-(z+1)/2)*norm.cdf(-np.log(row['d'])/(row['sigma_stock']*(5+zeta)**0.5)
    diff=g_tzeta-g_zeta
    return  row['r']*(1-R)*(1-PS_0+np.exp(row['r']*zeta)*diff)/(PS_0-row['PS_t']*np.exp(-row['

boeing['ct']=boeing.apply(lambda row: G(row),axis=1)*100
```

In [25]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
plt.plot(boeing['date'], boeing['ct'],label="5y (Model)")
#plt.show()
plt.plot(boeing['date'], boeing['spread5y'],label="5y (Real)")
plt.legend()
plt.xlabel("Date")
plt.ylabel("Spread (%)")
plt.show()
```



Market spread is overpriced.

b

In [26]:

```
def regression(df):
    X = df["ct"]
    y = df["S"]

    model = sm.OLS(y, X).fit()
    sensitivity = model.params[0]
    return sensitivity

def RPV01(df_t):
    # These terms are fixed at 't':
    S_0, Lbar, D, Lambda, r, R = df_t[["S", "L", "D", "lambda", "r", "R"]]
    LD = Lbar * D
    d = (S_0 + Lbar * D) / LD * np.exp(Lambda**2)
    T, Sigma_S = df_t[["T", "sigma_stock"]]
    A_0 = np.sqrt(Sigma_S * (S_0 / (S_0 + Lbar * D))**2 * 0 + Lambda**2)
    A_T = np.sqrt(Sigma_S * (S_0 / (S_0 + Lbar * D))**2 * T + Lambda**2)

    P_0 = norm.cdf(-A_0/2 + np.log(d)/A_0) - d * norm.cdf(-A_0/2 - np.log(d)/A_0)
    P_T = norm.cdf(-A_T/2 + np.log(d)/A_T) - d * norm.cdf(-A_T/2 - np.log(d)/A_T)

    Sigma_V = Sigma_S * S_0 / (S_0 + LD)
    z = np.sqrt(1/4 + 2*r / Sigma_V**2)

    Epsilon = Lambda**2 / Sigma_V**2

    G_TEpsilon = \
        d**(z+0.5) * \
        norm.cdf(-np.log(d) / (Sigma_V * np.sqrt(T+Epsilon)) - \
                 z * Sigma_V * np.sqrt(T+Epsilon)) + \
        d**(-z+0.5) * \
        norm.cdf(-np.log(d) / (Sigma_V * np.sqrt(T+Epsilon)) + \
                 z * Sigma_V * np.sqrt(T+Epsilon))
    G_Epsilon = \
        d**(z+0.5) * \
        norm.cdf(-np.log(d) / (Sigma_V * np.sqrt(Epsilon)) - \
                 z * Sigma_V * np.sqrt(Epsilon)) + \
        d**(-z+0.5) * \
        norm.cdf(-np.log(d) / (Sigma_V * np.sqrt(Epsilon)) + \
                 z * Sigma_V * np.sqrt(Epsilon))
    rpv01_t = \
        (P_0 - P_T * np.exp(-r*T) - np.exp(r*Epsilon) * (G_TEpsilon - G_Epsilon)) /\
        (r)
    return rpv01_t
```

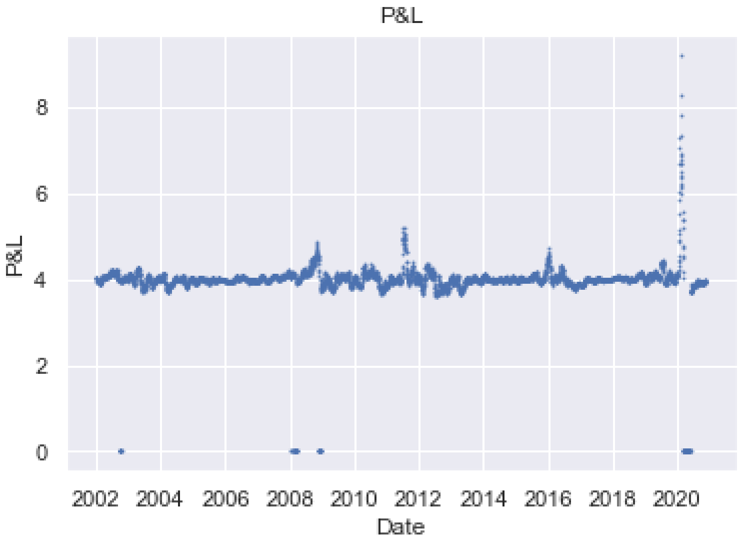
```
boeing_sensitivity = regression(boeing)
boeing["$hedge"] = boeing.apply(RPV01, axis=1) * boeing_sensitivity
print('hedge ratio:',boeing_sensitivity)
```

```
hedge ratio: 270.13424551022797
```

C

In [32]:

```
Boeing=boeing
alpha = 3.0
Boeing["1y_AvgDiff"] = (Boeing["ct"]-Boeing["spread5y"]).rolling(250).mean()
Boeing["signal"] = 0
Boeing["signal"] = np.where(Boeing["ct"]-Boeing["spread5y"] \
                            >= Boeing["1y_AvgDiff"]*alpha, 1, Boeing["signal"])
Boeing["signal"] = np.where(Boeing["ct"]-Boeing["spread5y"] \
                            >= Boeing["1y_AvgDiff"]/alpha, -1, Boeing["signal"])

c = 0
Boeing["CDS Price"] = (Boeing["spread5y"] - c) / (Boeing["r"] + (Boeing["spread5y"] / (1 - Boeir
                        (1 - np.exp(-(Boeing["r"] + (Boeing["S"] / (1 - Boeing["R"]))) * Boeing[
Position_Length = 30  # We hold each position for 1-month; Approx. 30 days
Boeing["P&L"] = 0.0
Boeing["P&L"] = np.where(Boeing["signal"] == 1,
                            (Boeing["CDS Price"].shift(-Position_Length)+Boeing["CDS Price"])/Bo
                            (Boeing["$hedge"].shift(-Position_Length)+Boeing["$hedge"])/Boeing["
                            Boeing["P&L"])
Boeing["P&L"] = np.where(Boeing["signal"] == -1,
                            (-Boeing["CDS Price"].shift(-Position_Length)-Boeing["CDS Price"])/-
                            (-Boeing["$hedge"].shift(-Position_Length)-Boeing["$hedge"])/-Boeing
                            Boeing["P&L"])

Boeing_plotted = Boeing.dropna()
plt.scatter(Boeing_plotted["date"], Boeing_plotted["P&L"], s=0.8, alpha=0.75)
plt.title("P&L")
plt.xlabel("Date")
plt.ylabel("P&L")
plt.show()


# Used for analytics below
avgMonthlyRet = np.mean(Boeing_plotted["P&L"])
avgAnnualRet = avgMonthlyRet*12
```



In [29]:

```
print("Average Return: ", avgMonthlyRet, "%")
```

```
Average Return:  3.940652763138677 %
```