# MFE409 HW6

June 7, 2019

*Group Members: Sejal Bharati, Huanyu Liu, Tongsu Peng, Yong Jia Tan*

## 1 Question 1

Our group looked at Goldman Sachs annual reports for the years 2009 and 2018.

For 2009, Goldman Sachs calculated Tier 1 capital ratio, Tier 1 leverage ratio, Tier 1 common ratio, and tangible common shareholders' equity to risk-weighted assets ratio according to Basel I. Also, as of 2009 they were working to implement requirements set out for Basel II.

In 2018, Common Equity Tier 1 ratios were calculated with Basel III approach. Basel III Advanced ratios were applied to firm as of both December 2018 and December 2017. Risk-Weighted Assets were calculated in accordance to the Basel III Advanced Rules, for credit risk, market risk, operational risk.

## 2 Question 2

Reasoning for the problems are handwritten later. Code and results are below. Note: risk-free rate $r = 0$, so discounting is not considered.

### 2.1 1

```
In [1]: import numpy as np
        from scipy import optimize

        R = 0.6
        cds_3 = 0.005
        cds_5 = 0.006
        cds_10 = 0.01

        lambda_1 = cds_3 / (1-R)

        def at_5(lambda_2, lambda_1, R, cds_5):
            top = 1 - np.exp(-3*lambda_1-2*lambda_2)
            bottom = 1/lambda_1*(1-np.exp(-3*lambda_1)) + 1/lambda_2*np.exp(-3*lambda_1)*(1-np
            val = top / bottom * (1-R) - cds_5
            return val

        lambda_2 = optimize.fsolve(at_5, args=(lambda_1,R,cds_5), x0=1)[0]
```

```
def at_10(lambda_3, lambda_1, lambda_2, R, cds_10):
    top = 1 - np.exp(-3*lambda_1-2*lambda_2-5*lambda_3)
    bottom = 1/lambda_1*(1-np.exp(-3*lambda_1)) + 1/lambda_2*np.exp(-3*lambda_1)*(1-np
    val = top / bottom * (1-R) - cds_10
    return val

lambda_3 = optimize.fsolve(at_10, args=(lambda_1, lambda_2, R,cds_10), x0=1)[0]

print(lambda_1,lambda_2,lambda_3)
```

```
0.012499999999999999 0.018893851571742504 0.03640352563760203
```

## 2.2 2

In [2]:

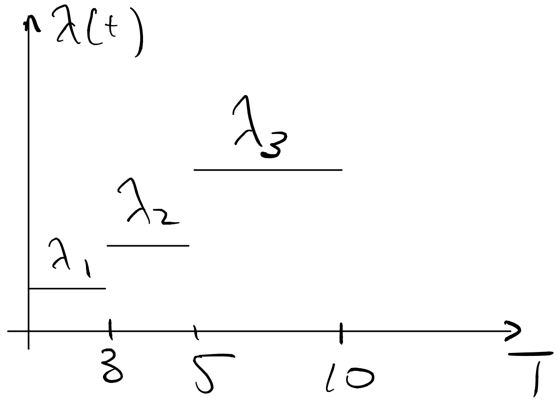
```
v = []
for i in range(6):
    v.append(np.exp(-lambda_1 * (i+1)*0.5))
for i in range(4):
    v.append(np.exp(-lambda_1 * 3 - lambda_2 * (i+1) * 0.5))
for i in range(10):
    v.append(np.exp(-lambda_1 * 3 - lambda_2 * 2 - lambda_3 * (i+1) * 0.5))

v = np.array(v)
q = 1 - v
cf = np.array([3,3,3,3,3,3,3,3,3,3,3,103])
v = v[:12]
q = q[:12]
print(np.sum(cf * v + cf * q * R))
```

```
131.09918088133094
```

In [ ]:

Question 2

1. Assume $\lambda$ is constant between each time period:




Using the exact formula for CDS Spread:

$$CDS(3) = (1-R)\frac{\int_0^3 \lambda_1 e^{-\lambda_1 \tau} d\tau}{\int_0^3 e^{-\lambda \tau} d\tau} = (1-R)\lambda_1$$

Using $CDS(3) = 50bps$, $R = 60\%$, we can solve for $\lambda_1$.

$$CDS(5) = (1-R)\frac{\int_0^3 \lambda_1 e^{-\lambda_1 \tau} d\tau + \int_3^5 \lambda_2 e^{-3\lambda_1 - \lambda_2(\tau-3)} d\tau}{\int_0^3 e^{-\lambda_1 \tau} d\tau + \int_3^5 e^{-3\lambda_1 - \lambda_2(\tau-3)} d\tau}$$

$$= \frac{1 - e^{-3\lambda_1 - 2\lambda_2}}{\frac{1}{\lambda_1}(1-e^{-3\lambda_1}) + \frac{1}{\lambda_2} e^{-3\lambda_1}(1-e^{-2\lambda_2})}$$

Using $CDS(5) = 60bps$, $R = 60\%$, and $\lambda_1$, solve for $\lambda_2$.

$$CDS(10) = (1-R)\frac{\int_0^3 \lambda_1 e^{-\lambda_1 \tau} d\tau + \int_3^5 \lambda_2 e^{-3\lambda_1 - \lambda_2(\tau-3)} d\tau + \int_5^{10} \lambda_3 e^{-3\lambda_1 - 2\lambda_2 - \lambda_3(\tau-5)} d\tau}{\int_0^3 e^{-\lambda \tau} d\tau + \int_3^5 e^{-3\lambda_1 - (\tau-3)\lambda_2} + \int_5^{10} e^{-3\lambda_1 - 2\lambda_2 - (\tau-5)\lambda_3} d\tau}$$

$$= \frac{1 - e^{-3\lambda_1 - 2\lambda_2 - 5\lambda_3}}{\frac{1}{\lambda_1}(1-e^{-3\lambda_1}) + \frac{1}{\lambda_2} e^{-3\lambda_1}(1-e^{-2\lambda_2}) + e^{-3\lambda_1 - 2\lambda_2}(1-e^{-5\lambda_3})}$$

Using $CDS(10) = 100bps$, $R = 60\%$, $\lambda_1, \lambda_2$, solve for $\lambda_3$.

$\Rightarrow$ Code and results in PDF.

2. Given $V(t) = e^{-\int_0^t \lambda(\tau) d\tau}$, $Q(t) = 1 - V(t)$, we can find the price of the coupon bond. Code and results in PDF.

**3. Dynamic credit model**

1.

In [1]:
```
import numpy as np
import scipy.linalg as lg
import pandas as pd
```

In [2]:
```
index = ['AAA', 'AA', 'A','BBB', 'BB', 'B', 'CCC','Default']
p0 = np.zeros(shape=(8,8))
for i in range(8):
    p0[i][i] = 1
p0_df = pd.DataFrame(p0,index=index, columns=index)
p1 = np.array(
    [[90.81,8.33,0.68,0.06,0.12,0,0,0],
     [0.7,90.65,7.79,0.64,0.06,0.14,0.02,0],
     [0.09,2.27,91.05,5.52,0.74,0.26,0.01,0.06],
     [0.02,0.33,5.95,86.93,5.3,1.17,1.12,0.18],
     [0.03,0.14,0.67,7.73,80.53,8.84,1,1.06],
     [0,0.11,0.24,0.43,6.48,83.46,4.07,5.2],
     [0.22,0,0.22,1.3,2.38,11.24,64.86,19.79],
     [0,0,0,0,0,0,0,100]]
) / 100
p1_df = pd.DataFrame(p1,index=index,columns=index)
print('P0')
print(p0_df)
print('\n\nP1')
print(p1_df)
```

```
P0
         AAA   AA    A  BBB   BB    B  CCC  Default
AAA      1.0  0.0  0.0  0.0  0.0  0.0  0.0      0.0
AA       0.0  1.0  0.0  0.0  0.0  0.0  0.0      0.0
A        0.0  0.0  1.0  0.0  0.0  0.0  0.0      0.0
BBB      0.0  0.0  0.0  1.0  0.0  0.0  0.0      0.0
BB       0.0  0.0  0.0  0.0  1.0  0.0  0.0      0.0
B        0.0  0.0  0.0  0.0  0.0  1.0  0.0      0.0
CCC      0.0  0.0  0.0  0.0  0.0  0.0  1.0      0.0
Default  0.0  0.0  0.0  0.0  0.0  0.0  0.0      1.0


P1
            AAA      AA       A     BBB      BB       B     CCC  Default
AAA      0.9081  0.0833  0.0068  0.0006  0.0012  0.0000  0.0000   0.0000
AA       0.0070  0.9065  0.0779  0.0064  0.0006  0.0014  0.0002   0.0000
A        0.0009  0.0227  0.9105  0.0552  0.0074  0.0026  0.0001   0.0006
BBB      0.0002  0.0033  0.0595  0.8693  0.0530  0.0117  0.0112   0.0018
BB       0.0003  0.0014  0.0067  0.0773  0.8053  0.0884  0.0100   0.0106
B        0.0000  0.0011  0.0024  0.0043  0.0648  0.8346  0.0407   0.0520
CCC      0.0022  0.0000  0.0022  0.0130  0.0238  0.1124  0.6486   0.1979
Default  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000   1.0000
```

2.

$$\Lambda dt = \frac{dP}{P}$$
$$\Lambda = \frac{1}{P}\frac{dP}{dt}$$

3.

$$\Lambda dt = \frac{dP}{P}$$
$$\int \Lambda dt = \int \frac{1}{P} dP$$
$$\Lambda t = ln(P_t) - ln(P_0)$$

Because $\Lambda$ is constant,

$$\Lambda = ln(P_1) - 0$$

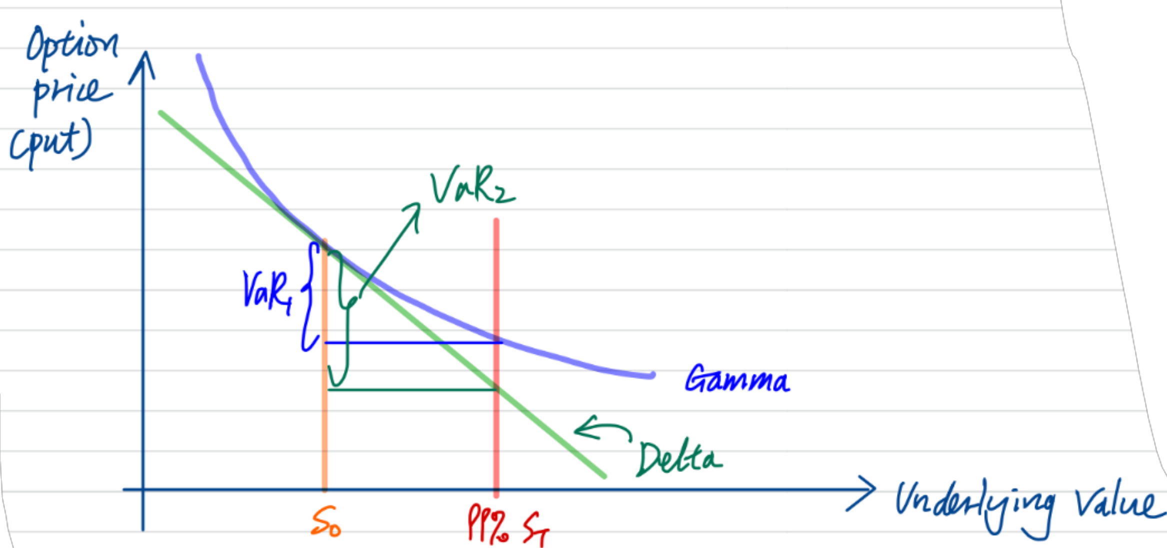
4.

In [3]:
```
lamb = lg.logm(p1)
lambda_df = pd.DataFrame(lamb,index=index, columns=index)
print(lambda_df)
```

```
               AAA        AA         A       BBB        BB         B       
CCC  \
AAA     -0.096756  0.091804  0.003540  0.000213  0.001366 -0.000149 -0.0
00017
AA       0.007679 -0.099596  0.085676  0.004529  0.000146  0.001445  0.0
00187
A        0.000889  0.024880 -0.096887  0.061815  0.006609  0.002275 -0.0
00450
BBB      0.000154  0.002828  0.066721 -0.145127  0.062726  0.009384  0.0
14248
BB       0.000323  0.001346  0.004537  0.092356 -0.223974  0.107065  0.0
10057
B       -0.000090  0.001186  0.002358  0.001003  0.078637 -0.188845  0.0
54935
CCC      0.002856 -0.000310  0.002023  0.015707  0.026174  0.151364 -0.4
37788
Default  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.0
00000

              Default
AAA     -8.424479e-07
AA      -8.649716e-05
A        5.484132e-04
BBB     -1.834970e-04
BB       7.803713e-03
B        5.070411e-02
CCC      2.400228e-01
Default  0.000000e+00
```

**1. “If you use the Delta approach for a positive Gamma option, you will overestimate the VaR”. Is it true or false? Explain.**

True. As shown in the graph.




Example (using put)

$VaR_1 < VaR_2$

**2. Assume you are the CEO of a bank and feel that the risk regulations are too constraining. What are ways you could use to take more risk while still respecting the regulation:**

**(a) Under Basel I**

One main weakness of Basel I is that for the on-balance sheet exposure, its differentiation of the four brackets is not very detailed nor through. For example, all claims in OECD governments, banks or public sector entities receive a very low risk weight of 0% or 20%. However, out of 38 OECD countries, many are still developing countries, which imply that they may have higher risks. Even if we ignore the developed or developing country distinction, each country has its own sovereign risk and idiosyncratic risks. Therefore, I could potentially load up government bonds from Mexico, Greece, which has a much higher yield and risk profile than US treasury bonds, but my risk-adjusted asset would remain very low.

Moreover, for the on-balance sheet, the risk weight does not differ with maturity, so I could load up on long-maturity investments, which have higher interest rate risk.

**(b) Under Basel II**

Basel II allows banks to provide its own estimates for assessing credit risks and to use its own internal models. Therefore, I could intentionally be overly optimistic and use models that led to lower risk assessment.

Basel II credit risk assessment replies on rating from rating agency. However, most rating agency reacts slowly to the market and the rating are done ex-post so I could find investment with more risks than the rating implies.

**3. Why is Basel II blamed for precipitating the financial crisis?**

First, Basel II allows banks to use their own internal models to assess risks, which would be used to determine the minimum capital requirement. This created incentives for banks to under-estimate their credit risks and being over-optimistic.

Second, banks' internal models are not necessarily superior in measuring risks and they could perform poorly.

Third, in the Basel II framework, the credit risk weight depends on rating agencies, which may have conflicts of interests. And as we know, prior to the financial crisis, the rating agencies did a terrible job rating the subprime CDOs, which greatly affects the credit risk assessment of the banks.

Fourth, the capital requirement could be pro-cyclical, which in turn amplify the market volatility.

Lastly, the average level of capital required is not adequate to capture