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 Λ 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
```

5.

In [4]:
```
default = np.zeros(shape=(7,7))
for i,v in enumerate([1,2,3,4,5,7,10]):
    p = lg.expm(lamb * v)
    default[i] = p[:7, 7]
default_df = pd.DataFrame(default.T,index=index[:-1],columns=[1,2,3,4,5,7
,10])
print(default_df)
```

```
                1         2         3         4         5         7
10
AAA  1.961059e-17  0.000018  0.000076  0.000196  0.000397  0.001122  0.0
03298
AA   1.539775e-17  0.000177  0.000547  0.001134  0.001965  0.004468  0.0
10656
A    6.000000e-04  0.001479  0.002821  0.004732  0.007268  0.014288  0.0
29495
BBB  1.800000e-03  0.006787  0.014100  0.023158  0.033554  0.057227  0.0
97138
BB   1.060000e-02  0.025855  0.044487  0.065421  0.087798  0.134367  0.2
03035
B    5.200000e-02  0.104150  0.154160  0.200946  0.244115  0.319724  0.4
10019
CCC  1.979000e-01  0.332380  0.425895  0.492711  0.541928  0.608815  0.6
69627
```

6.

There are some differences which maybe due to that the default table on the lecture notes is provided by S&P, while my results are based on Moody's data.

7.

In [5]:
```
bond_price = 0
for i in range(12):
    default_rate = lg.expm(lamb * (i + 1) * 0.5)[3][-1]
    bond_price += 3 * (1 - default_rate)
bond_price += 100 * (1 - default_rate) + 60 * default_rate
print(bond_price)
```

```
133.5262232035036
```

8.

In [6]:
```
recovery = 0.6
default3 = lg.expm(lamb * 3)[3][-1]
default5 = lg.expm(lamb * 5)[3][-1]
default10 = lg.expm(lamb * 10)[3][-1]
cds_spread3 = (1 - recovery) * default3 / (1 - default3)
cds_spread5 = (1 - recovery) * default5 / (1 - default5)
cds_spread10 = (1 - recovery) * default10 / (1 - default10)
print(cds_spread3)
print(cds_spread5)
print(cds_spread10)
```

```
0.005720729225183936
0.013887451926032468
0.04303544919660205
```