

UNIVERSITÀ DEGLI STUDI DI TORINO

M.Sc. in Quantitative Finance

Coursework



UNIVERSITÀ
DI TORINO

Quantitative Risk Management Coursework

Alex Dal Cin - 1088357

ACADEMIC YEAR 2023/2024

Contents

1	Basel credit Risk Model	3
1.1	First section	3
1.2	Merton - Vasicek	8
2	Loss Given Default Estimation	11
2.1	Beta Distribution Fitting	11
2.2	Example 35 in Roncalli's book	14
3	Default Probability	19
3.1	Merton's Model	19
3.2	CIR model	22
3.2.1	Survival Probability	22
3.2.2	Premium of CDS	23
3.3	Exponential, Gompertz and piecewise exponential	24
3.4	Credit migration matrix	28
3.4.1	Piecewise constant hazard function	28
3.4.2	(optional) Markov Generator	31
4	Default correlation	39
5	Counterparty Credit Risk	47

Chapter 1

Basel credit Risk Model

1.1 First section

In this coursework, these values are given:

- 100, which is the number of credits in our portfolio
- 1\$ mln, which is the EAD and represents the outstanding debt that we have at the time of default
- 50%, which is the Expected LGD, where LGD is the Loss given Default and it is the expected percentage of exposure at default that is lost if the debtor defaults
- 5%, which is the Probability of default (PD) and is the default risk of the debtor on 1 year of time horizon

Using these parameters, our goal is to reproduce Figure 3.21 of Roncalli. First of all, by looking at Figure 3.21 we can actually see that it is made of three graphs:

- Quantile function that changes as α changes.
- Cumulative distribution function (CDF) of the Loss.
- Probability density function (PDF) of the Loss.

All these three graphs plot two lines, that have two different values of ρ :

- 10%
- 30%

I achieved this result with this code on MATLAB:
We start from this formula of the conditional default probability:

$$p_i(X) = \Phi \left(\frac{B_i - \sqrt{\rho}X}{\sqrt{1 - \rho}} \right) \quad (1.1)$$

Which then becomes, after recovering the unconditional default probability:

$$p_i(X) = \Phi \left(\frac{\Phi^{-1}(p_i) - \sqrt{\rho}X}{\sqrt{1 - \rho}} \right) \quad (1.2)$$

Which, if we plug it, together with the derivation of X, in the formula of $F^{-1}(\alpha)$

$$F^{-1}(\alpha) = \sum_{i=1}^n \omega_i E[LGD_i] \Phi \left(\frac{\Phi^{-1}(p_i) - \sqrt{\rho}X}{\sqrt{1 - \rho}} \right) \quad (1.3)$$

The PD is estimated by the bank, and in this case is p_i , and we use it to obtain B on the IRB formula by inverting the standard normal, which is a barrier: if the asset value Z falls below this barrier, then the default occurs. ρ is the exposure to common risk factors, X is a standard normal that can be obtained by applying the inverse of the standard normal on $(1 - \alpha)$, which becomes the inverse of α with changed sign in our formula. LGD is given by the regulators, this does not happen everytime, in advanced RB we need to model it. We obtain, as a consequence $F_inverse$ on our code. F_l is equal to the numerical solution of $F_inverse=l$ and $f_l = \frac{1}{\partial_\alpha F^{-1}(F(l))}$.

I, then, plotted, all these 3 values for all the ρ in order to do a sensitivity analysis: analyze how the value changes as some parameter changes, in the first case is α and in the second and the third is Loss. In the third picture we can notice that the PDF in the case of $\rho = 0.3$ goes to 1 as the Loss goes to 0, whereas with $\rho = 0.1$, it goes to 0, then, $\rho = 0.1$ hits a maximum just to then going down and reach a similar value of $\rho = 0.3$ starting from Loss=5. $\rho = 0.3$ has always a negative slope.

Listing 1.1: Code that generates the values that we want to plot

```
% 1. Roncalli Figure 3.21

N=100;
EAD = 1;
E_LGD = 0.5;
p_i = 0.05;
alpha=linspace(0,0.99,100);
rho=[0.1;0.3];
```

```
l = linspace(0,10,1000);
F_inverse= N * EAD* E_LGD* normcdf((norminv(p_i)+sqrt(rho).*
    norminv(alpha))./(sqrt(1-rho)));
F_l = normcdf((sqrt(1-rho).*(norminv((l)/(N*EAD*E_LGD)))-
    norminv(p_i))./(sqrt(rho)));
f_l = 1./(N*EAD*E_LGD*sqrt((rho)./(1-rho)).*(1)./(normpdf(
    norminv(F_l))).*normpdf((norminv(p_i)+sqrt(rho).*norminv(
    F_l))./(sqrt(1-rho))));
```

Listing 1.2: Plot of Figure 3.29

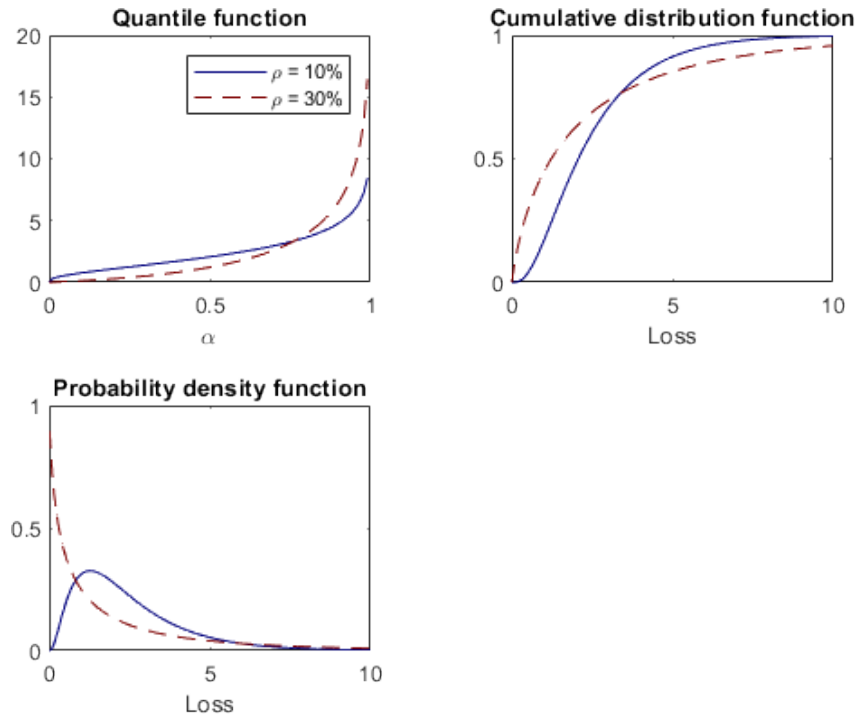
```
%plot

%figure;
bluecolor=[0, 0, 125/255];
redcolor=[126/255, 0, 0];
tiledlayout(2,2);
nexttile
plot(alpha,F_inverse(1,:), "Color",bluecolor);
hold on
plot(alpha,F_inverse(2,:), "Color",redcolor, "LineStyle", "--");
xlabel("\alpha")
title("Quantile function")
legend("\rho = 10%", "\rho = 30%")

nexttile
plot(l,F_l(1,:), "Color",bluecolor);
hold on
plot(l,F_l(2,:), "Color",redcolor, "LineStyle", "--");
xlabel("Loss")
title("Cumulative distribution function")

nexttile
plot(l,f_l(1,:), "Color",bluecolor);
hold on
plot(l,f_l(2,:), "Color",redcolor, "LineStyle", "--");
xlabel("Loss")
title("Probability density function")
```

This is the final plot



I also replicated the table 3.24 by applying this code

```
rho=0.1;
alpha=[0.10 0.25 0.50 0.75 0.90 0.95]
l=[0.1 1 2 3 4 5]
F_inverse= N * EAD* E_LGD* normcdf((norminv(p_i)+sqrt(rho).*
    norminv(alpha))./(sqrt(1-rho)));
F_l = normcdf((sqrt(1-rho).*(norminv((l)/(N*EAD*E_LGD)))-
    norminv(p_i))./(sqrt(rho)));
f_l = 1./(N*EAD*E_LGD*sqrt((rho)./(1-rho)).*(1)./(normpdf(
    norminv(F_l))).*normpdf((norminv(p_i)+sqrt(rho).*norminv(
    F_l))./(sqrt(1-rho))));
```

1

0.1 1 2 3 4 5

F(1)

0.03 16.86 47.98 70.44 83.8 91.26

f(1)

1.04	31.19	27.74	17.39	9.9	5.43
------	-------	-------	-------	-----	------

alpha

0.1	0.25	0.5	0.75	0.9	0.95
-----	------	-----	------	-----	------

F inverse

0.77	1.25	2.07	3.28	4.78	5.9
------	------	------	------	------	-----

1.2 Merton - Vasicek

In this coursework, these parameters are given:

- $\omega_i = 100$, which is the exposure at default
- $E[LG D_i] = 70\%$, which is the expected Loss Given Default
- $\rho = 0.2$
- $\alpha = 90\%$
- $p_i = 10\%$

We are requested to analyse the sensitivity of the risk contribution to parameter changes under Merton-Vasicek model, as in Figure 3.22 of Roncalli's book.

We are in the same framework as before: we study $RC = \omega_i E[LG D_i] p_i$, where p_i is the value that we found before. We repeat these calculation 4 times with these parameters changes:

- $E[LG D]$ equal to 70% or 30% and alpha equal to 90%
- alpha equal to 90%, 99.5% and 40% and ρ equal to 20%
- p_i equal to 5% or 10% and alpha equal to 90%
- p_i equal to 5% or 10% and alpha equal to 99.9%

I divided my code in 4 parts, one for each parameter that i had to study, each part is organized as follows:

- Mathematical part where i get the values, where i repeated the formula with the parameters
- Plot part where i plotted $RC(1,:)$ or $RC(2,:)$ or $RC(3,:)$, which is the value of RC with the different parameters that we want to study.

N.B. in the following code there will be variables like "bluecolor" or "redcolor" or "greencolor", i created them in order to use the same color, they don't add anything. The same story applies to "figure;", which is a command that i used to save the plots.


```
bluecolor=[0, 0, 125/255];
redcolor=[126/255, 0, 0];
greencolor=[0, 126/255,0];
w=100;
rho=0.2;
alpha=0.9;
p_i = linspace(0,1,100);

%figure;
tiledlayout(2,2);

nexttile
E_LGD = [0.7;0.3];
RC = w*E_LGD*normcdf((norminv(p_i)+sqrt(rho).*norminv(alpha))
    /(sqrt(1-rho)));
plot(p_i*100,RC(1,:), "Color",bluecolor);
hold on
plot(p_i*100,RC(2,:), "Color",redcolor, "LineStyle", "--");
legend("E[LGD]=70%", "E[LGD]=30%", "Location", "best")
title("\alpha = 90%")
xlabel("p_i (in %)")

nexttile
E_LGD=0.7;
alpha = [0.9;0.995;0.40];
RC = w*E_LGD*normcdf((norminv(p_i)+sqrt(rho).*norminv(alpha))
    /(sqrt(1-rho)));
plot(p_i*100,RC(1,:), "Color",bluecolor);
hold on
plot(p_i*100,RC(2,:), "Color",redcolor, "LineStyle", "--");
hold on
plot(p_i*100,RC(3,:), "Color",greencolor, "LineStyle", "-.");
legend("\alpha=90%", "\alpha=99.5%", "\alpha=40%", "Location", "
    best")
title("\rho = 20%")
xlabel("p_i (in %)")

nexttile
rho = linspace(0,1,100);
alpha=0.9;
p_i = [0.05;0.10];
RC = w*E_LGD*normcdf((norminv(p_i)+sqrt(rho).*norminv(alpha))
    ./ (sqrt(1-rho)));
plot(rho*100,RC(1,:), "Color",bluecolor);
hold on
plot(rho*100,RC(2,:), "Color",redcolor, "LineStyle", "--");
```

```

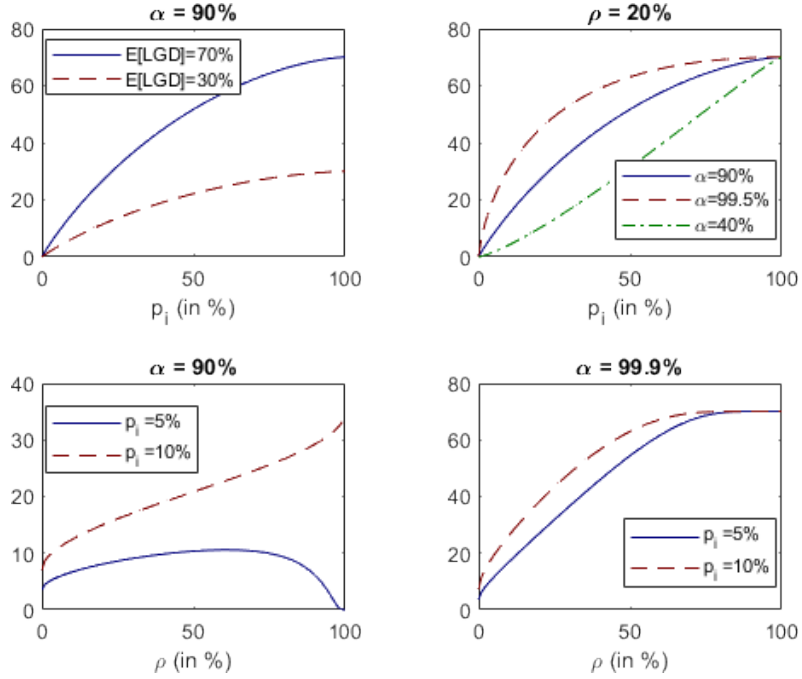
legend("p_i =5%", "p_i =10%", "Location", "best");
title("\alpha = 90%")
xlabel("\rho (in %)")

nexttile
alpha=0.999;
RC = w*E_LGD*normcdf((norminv(p_i)+sqrt(rho).*norminv(alpha))
    ./ (sqrt(1-rho)));
plot(rho*100, RC(1,:), "Color", bluecolor);
hold on
plot(rho*100, RC(2,:), "Color", redcolor, "LineStyle", "--");
legend("p_i =5%", "p_i =10%", "Location", "best");
title("\alpha = 99.9%")
xlabel("\rho (in %)")

```

This is the final plot, we can notice:

- RC is an increasing function of $E[LGD]$ and α (top graphs).
- In the bottom left, RC, in the case of $p_i = 10\%$ increases with ρ , reaching a value of 35, half of the nominal LGD. In the other case, where $p_i = 5\%$, RC increases and then decreases.
- Maximum reached at $\rho = 60$ more or less, when $\alpha = 99.9\%$, this effect vanishes.



Chapter 2

Loss Given Default Estimation

2.1 Beta Distribution Fitting

This coursework gives as an input a serie of LGD for a sample of homogeneous defaulted credit: 68%, 90%, 22%, 45%, 17%, 25%, 89%, 65%, 75%, 56%, 87%, 92%, 46%.

Beta distribution is a way of modelling Loss Given Default. It is a stochastic modelling type, as opposed to Economic modelling; with this type of approach we need to select a function which is the Beta function. Why do we choose this function? Because of its properties:

- Domain in $[0,1]$, so that we are sure that we stay within the range that we expect.
- First two moments look very easy, reasonably tractable, as we will also use it in the estimation of α and β in the method of moments
- It is very flexible, depending on the choice of parameters, you can obtain different shapes (skewness, symmetric, ecc)

Limitation: The standard deviation can not be freely chosen given the mean. Anyway, below we apply the method of moments, which matches the first two moments.

$$\begin{cases} \hat{\mu} = \frac{\hat{\alpha}}{\hat{\alpha} + \hat{\beta}} \\ \sigma^2 = \frac{\hat{\alpha}\hat{\beta}}{(\hat{\alpha} + \hat{\beta})^2(\hat{\alpha} + \hat{\beta} + 1)} \end{cases} \quad (2.1)$$

That, after being solved, gives the result that is written in the code.

The other method is Maximum Likelihood function, in this case i use a built-in function in order to obtain my parameters.

In both the methods, we will use the observed LGD to obtain the parameters alpha and beta

```
LGD = [0.68 0.90 0.22 0.45 0.17 0.25 0.89 0.65 0.75 0.56 0.87  
       0.92 0.46];  
LGD = sort(LGD);
```

```
% Method of moments
```

```
u=mean(LGD);  
sigma = std(LGD);  
alpha_mm = (u^2*(1-u))/(sigma)^2 - u;  
beta_mm = (u*(1-u)^2)/(sigma)^2 - (1-u);
```

```
% Maximum Likelihood Estimation
```

```
MLE = mle(LGD,"distribution","Beta");  
alpha_mle = MLE(1);  
beta_mle = MLE(2);
```

Now that I obtained the value of beta, both with MLE and MM, I am going to print their values:

```
fprintf('Beta MM value is %0f , Alpha MM value is %0f\n',  
       beta_mm,alpha_mm);  
fprintf('Beta MLE value is %0f , Alpha MLE value is %0f\n',  
       beta_mle,alpha_mle);
```

This is the output:

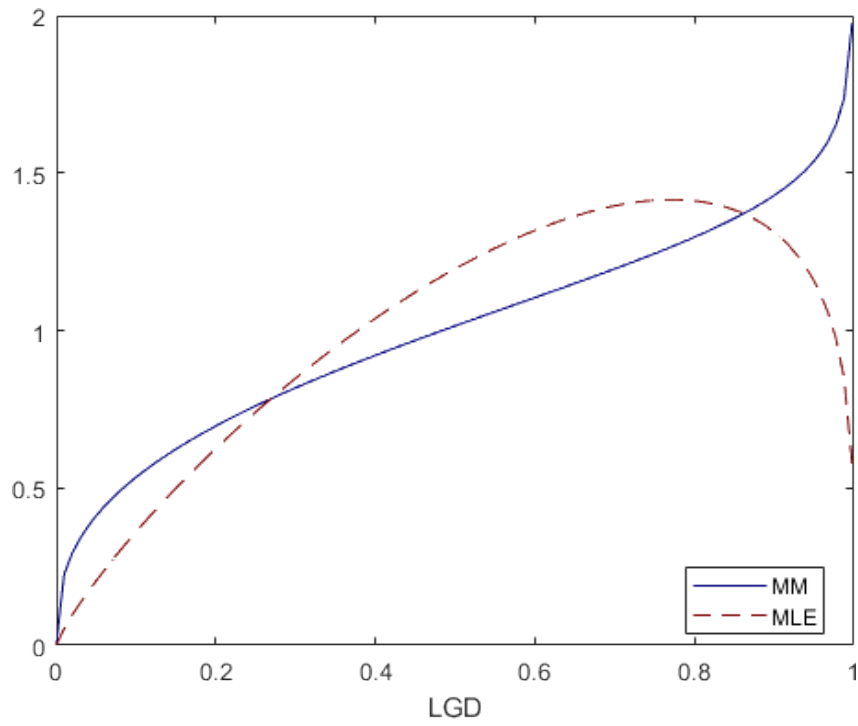
```
Beta MM value is 0.922513 , Alpha MM value is 1.370540  
Beta MLE value is 1.247769 , Alpha MLE value is 1.835553
```

Then, i plot all the values of the PDF as LGD changes:

```
% plot  
line = linspace(0,0.9975,100);  
%figure;  
PDF_mm = betapdf(line,alpha_mm,beta_mm);  
PDF_mle = betapdf(line,alpha_mle,beta_mle);  
plot(line,PDF_mm,"Color",bluecolor)  
hold on  
plot(line,PDF_mle,"Color",redcolor,"LineStyle","--");  
ax = gca;
```

```
ax.XTick = 0:0.2:1;  
ax.YTick = 0:0.5:2;  
xlabel("LGD");  
legend(ax,"MM","MLE","Location","best");
```

This is the final result, we can notice that they have different shapes:



2.2 Example 35 in Roncalli's book

LGD (in %)	0	10	20	25	30	40	50	60	70	75	80	90	100
\hat{p} (in %)	1	2	10	25	10	2	0	2	10	25	10	2	1

Table 2.1: Roncalli's data

First of all, I created the empirical distribution of LGD. I assumed a dimension of 100, so that i can easily convert \hat{p} equals the "quantity" of LGD in my array. For instance, $LGD = 0$ will appear only 1 time, $LGD = 10$ 2 times and so on.

```
LGD=[0 0.10 0.20 0.25 0.30 0.40 0.50 0.60 0.70 0.75 0.80 0.90
      1];
p=[0.01 0.02 0.10 0.25 0.10 0.02 0 0.02 0.10 0.25 0.10 0.02
    0.01]; %They sum up to 1
final_LGD=repelem(LGD,p*100);
```

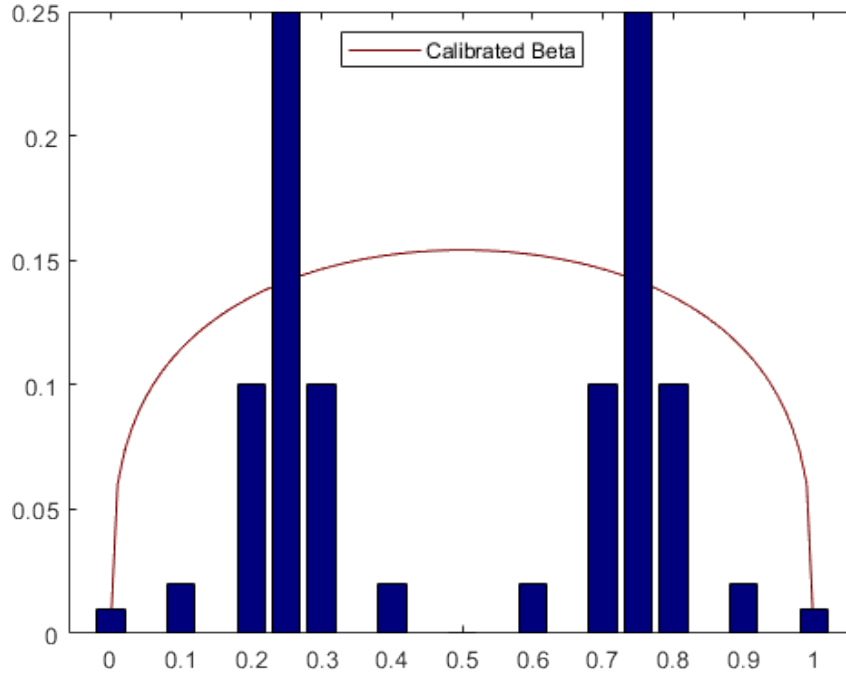
Then, i computed the PDF by using the method of moments, the same procedure that we have seen before:

```
u=mean(final_LGD);
sigma = std(final_LGD);
alpha_mm = (u^2*(1-u))/(sigma)^2 - u;
beta_mm = (u*(1-u)^2)/(sigma)^2 - (1-u);
line = linspace(0,1,100);
PDF_mm = betapdf(line,alpha_mm,beta_mm)*(1/2)*sigma;
```

And then, I plotted the Empirical (which is the LGD plotted with its frequencies) and the Calibrated Beta obtained through the method of moments. The Calibrated Beta is rescaled. Anyway, we can see that it is still very different from the Empirical one

```
plot(line,PDF_mm,"Color",redcolor)
hold on
bar(LGD,p,"FaceColor",bluecolor)
legend("Calibrated Beta","Location","north")
```

This is the final result



In the second part, we are given a portfolio of 10 loans, whose loss is equal to:

$$L = \sum_{i=1}^{10} EAD_i \cdot LGD_i \cdot 1_{\tau_i \leq T_i} \quad (2.2)$$

where $EAD_i = 1000\$$, $T_i = 5$ years for each i and τ_i is exponentially distributed with intensity parameter λ_i equal to the table below, which is the one of Example 36 of Roncalli. In order to solve this exercise, first of all,

i	1	2	3	4	5	6	7	8	9	10
λ_i (in bps)	10	10	25	25	50	100	250	500	500	1000

Table 2.2: Roncalli's data (Example 36)

we write down below the data:

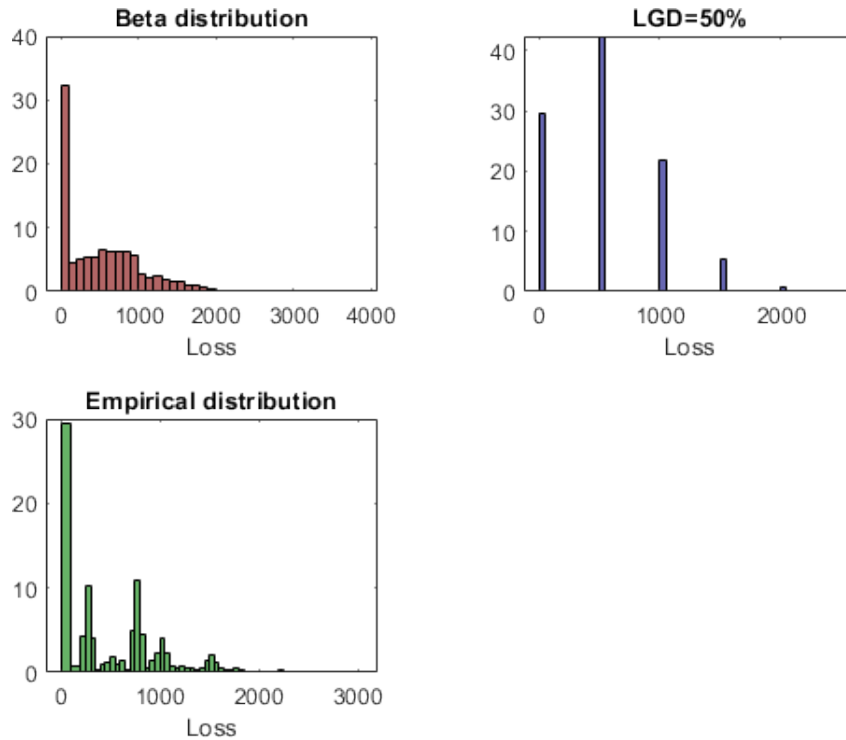
```
Q=10;
N=10000;
EAD=1000;
T=5;
LGD=[0 0.10 0.20 0.25 0.30 0.40 0.50 0.60 0.70 0.75 0.80 0.90
      1];
p=[0.01 0.02 0.10 0.25 0.10 0.02 0 0.02 0.10 0.25 0.10 0.02
    0.01]; %They sum up to 1
lambda=[0.0010 0.0010 0.0025 0.0025 0.0050 0.01 0.0250 0.05
         0.05 0.10];
```

Then, we build what we want to plot. As we can see above, we are performing a Monte Carlo simulation, as a consequence N needs to be very high, so i preferred to optimize my code in order to make it as fast as possible. As a consequence, i put all the computation in one line, which will give the simulated values that we want. Here, a brief explanation:

- Beta: EAD is fixed, we create a set of (N,Q) random betas, each with parameter α_{mm} and β_{mm} , then, again, I created a set (N,Q) of random exponential, where there are Q different lambdas as parameter repeated N times; then, I computed the values of the indicator function, which is 1 if $\tau \leq T$, which means that default occurs, so the "losses" are different from 0. I multiplied these 3 values and sum it $Q=10$ times, obtaining N values.
- Granular: EAD and $E[LGD]=0.5$ fixed, with the exponential and then indicator I did the same as before. Again, I multiplied and sum $Q=10$ times, obtaining N values.
- EAD is fixed, exponential and then indicator are obtained as before, I used the empirical LGD, this time not in a dimension of 100, but (N,Q) , then I mixed the matrix randomly and reshaped it, in order to match the matrix dimensions. Then I multiplied and sum $Q=10$ times, obtaining N values.

```
Beta=sum(EAD*betarnd(repmat(alpha_mm,N,Q),repmat(beta_mm,N,Q))
.*(exprnd(1./repmat(lambda,N,1))<=T),2);
Granular=sum(EAD*0.5*(exprnd(1./repmat(lambda,N,1))<=T),2);
final_LGD2=repelem(LGD,p*N*Q);
Empirical=sum(EAD.*reshape(final_LGD2(randperm(numel(
final_LGD2))),N,Q).*(exprnd(1./repmat(lambda,N,1))<=T),2);
```

And this is the final plot, some differences might occur because of the width dimension of the blocks.



These are the VaRs: (99%,99.5%,99.9%)

```
Var_99_empirical=prctile(Empirical,99)
Var_995_empirical=prctile(Empirical,99.5)
Var_999_empirical=prctile(Empirical,99.9)
Var_99_granular=prctile(Empirical,99)
Var_995_granular=prctile(Empirical,99.5)
Var_999_granular=prctile(Empirical,99.9)
Var_99_beta=prctile(Empirical,99)
Var_995_beta=prctile(Empirical,99.5)
Var_999_beta=prctile(Empirical,99.9)
fprintf("VaR Empirical 99 level is %0f\n",Var_99_empirical)
fprintf("VaR Empirical 99.5 level is %0f\n",Var_995_empirical)
fprintf("VaR Empirical 99.9 level is %0f\n",Var_999_empirical)
fprintf("VaR Granular 99 level is %0f\n",Var_99_granular)
fprintf("VaR Granular 99.5 level is %0f\n",Var_995_granular)
fprintf("VaR Granular 99.9 level is %0f\n",Var_999_granular)
fprintf("VaR Beta 99 level is %0f\n",Var_99_beta)
fprintf("VaR Beta 99.5 level is %0f\n",Var_995_beta)
fprintf("VaR Beta 99.9 level is %0f\n",Var_999_beta)
```

And this is the output:

VaR Empirical 99 level is 2050.000000

VaR Empirical 99.5 level is 2250.000000
VaR Empirical 99.9 level is 2475.000000
VaR Granular 99 level is 2050.000000
VaR Granular 99.5 level is 2250.000000
VaR Granular 99.9 level is 2475.000000
VaR Beta 99 level is 2050.000000
VaR Beta 99.5 level is 2250.000000
VaR Beta 99.9 level is 2475.000000

Chapter 3

Default Probability

3.1 Merton's Model

Given the parameters:

- S_0 :8086
- Average maturity of debt: 1 year
- Face Value of Debt: 3500
- Equity volatility:18%

We are talking about a framework where:

- A firm is financed by equity and a single issue of zero-coupon debt with face value F and maturity T .
- Frictionless market
- Discount factors: $P(t, T) = \exp(-r(T - t))$.
- Value of the firm $V(t) = D(t) + S(t)$, where D is debt and S is equity.
- $\mu = r - \delta$ under the risk neutral measure, with δ being the payout ratio.

τ is the default time and it occurs if the value of the firm at maturity $V(T) < F$. AT MATURITY has to be lower in order to occur, if it happens before but at maturity is still higher, then it is not considered default. We obtain these payoffs:

- Equity= $V(T)-F$ if $V(T) \geq F$ and 0 otherwise \rightarrow Call

- Bonds=F if $W(T) \geq F$ and $V(T)$ otherwise $\rightarrow F - \max(0, F - V_T) \rightarrow F$ (that will be discounted) - Put.

Calibrating our data, we face this problem:

$$\begin{cases} S(0) = V(0)\Phi(d_1) - F \exp^{-rt} \Phi(d_2) \\ \sigma_S S_0 = \Phi(d_1)\sigma_V V(0) \end{cases} \quad (3.1)$$

We have 2 incognitas: $V(0)$ and σ_V , it might seems straightforward to obtain them because it is a system with 2 equations and 2 variables to found, but d_1 and d_2 are tricky and hide other variables that you can see on the code below "d1" and "d2". As a consequence, in order to solve it, I used the built-in tool "mertonmodel", which is a function of MatLab that solves the system. We can also use a Newton-Rhapson algorithm or a function that minimizes the differences. The fact is that we can not do it analytically, we have to rely to numerical methods.

```
r=0.02;
S_0=8086;
T=1;
F=35000;
vol_eq=0.18;
[PD,DD,V_0,vol_v]=mertonmodel(S_0,vol_eq,F,r);

%   either you subtract S_0 from V_0, or we can use the its
%   formula from BS
D_0=V_0 - S_0;

%   for the sake of completeness, i compute it also with the
%   BS formula
d1=(log(V_0 / F)+(r + 0.5*(vol_v)^2)*T)/(vol_v *sqrt(T));
d2=d1-vol_v*sqrt(T);
D_0_2= F*exp(-r*T)*normcdf(d2) + V_0 * normcdf(-d1);
%   I keep this formula because i still need d1 and d2

l= (F*exp(-r*T))/V_0;
s_0_1= -(1/T) * log(normcdf(d2)+ (1/l) * normcdf(-d1));
RR = (V_0/F) * exp(r*T)*(normcdf(-d1)/normcdf(-d2));
```

Now, let's print them

```
fprintf("Debt Value at t=0 is %0f \n",D_0);
fprintf("1 Year Term Spread is %0f \n",s_0_1);
fprintf("The endogenous Recover is %0f \n",RR);
fprintf("Volatility of firm value is %0f \n",vol_v)
```

Debt Value at $t=0$ is 34306.953566

1 Year Term Spread is 0.000000 (PS: It is very low, $2.09077199997628e-12$)

The endogenous Recover is 0.994705

Volatility of firm value is 0.034333

3.2 CIR model

Intensity can, indeed, be stochastic: τ is the first jump of a doubly stochastic (Cox) process. This allow to capture:

- varying nature
- uncertainty in intensity
- volatility in credit spreads

Consider an intensity-based model, in which the intensity is stochastic and follows a CIR-like process with parameter $k = 0.3, \theta = 0.15, /sigma = 0.2$.

3.2.1 Survival Probability

In order to plot the survival probability, we start by plugging in the parameters:

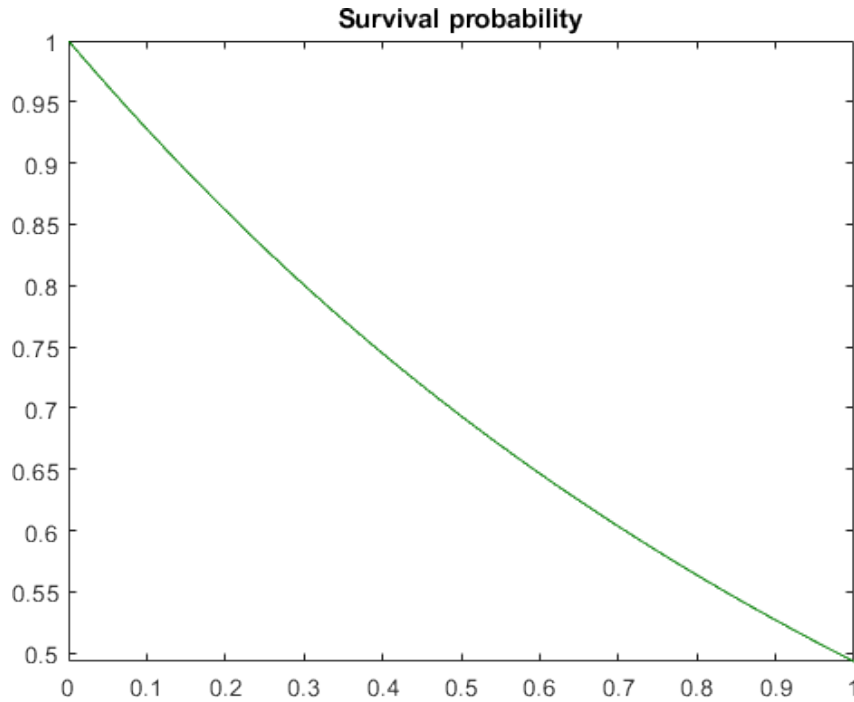
```
k=0.3;
theta=0.15;
sigma=0.2;
T=5;
N=100*T;
dt=T/N;
i=0:500;
```

Then, we apply the solution of a CIR stochastic process, which is the same that can be applied to bonds.

```
% We are going to use the formula of the Bond applied to our
case.

h=sqrt(k^2+2*sigma^2);
alpha=((2*h*exp((k+h)*dt*i/2))./(2*h+(k+h)*(exp(h*dt*i)-1)))
    .^(2*k*theta/sigma^2);
beta=(2*(exp(h*dt*i)-1))./(2*h+(k+h)*(exp(h*dt*i)-1));
S=alpha.*exp(-beta*theta);
figure;
plot(i/N,S,"Color",bluecolor)
title("Survival probability")
ylim([min(S), max(S)]);
```

And this is the final plot:



3.2.2 Premium of CDS

The fair premium of a CDS is annual, it offers a 5-years protection which is characterised by the above survival probabilities, $LGD = 5$ million, a constant $r = 3.5\%$. This is the code:

```
LGD=5000000;  
r=0.035;  
PremLeg=0;  
for j=1:T  
    PremLeg=PremLeg+sum([1*exp(-r*j)*S((j*100)+1)])  
end  
  
ts=0:dt:T  
ProtLeg=0;  
  
for l=1:N  
    ProtLeg=ProtLeg+((exp(-r*ts(l))+exp(-r*ts(l+1)))/2)*(S(l+1)-S(l))  
end  
ProtLeg=-LGD*ProtLeg  
R=ProtLeg/PremLeg  
fprintf('The value of the Fair Premium is %0f', R);
```

And this is the output:

The value of the Fair Premium is 781656.671373>>

3.3 Exponential, Gompertz and piecewise exponential

$\Delta t(inmonths)$	3	6	9	12	15	18	21	22
$n_D \Delta t$	2	5	9	12	16	20	25	29

Table 3.1: Roncalli's data (Example 37)

We are given these default frequencies by Roncalli, we aim to estimate the hazard function under:

- Exponential
- Gompertz
- Piecewise exponential

Survival function are used to characterize the probability of default and it is defined as

$$S(t) = Pr\tau > t = 1 - F(t) \quad (3.2)$$

As a consequence

$$f(t) = -\frac{\partial S(t)}{\partial t} \quad (3.3)$$

The hazard function is the instantaneous default rate given that the default has not occurred before t.

Starting from its definition, we obtain this formula

$$\lambda(t) = \frac{f(t)}{S(t)} = \frac{-\partial_t S(t)}{S(t)} = -\frac{\partial \ln S(t)}{\partial t} \quad (3.4)$$

Then

$$S(t) = e^{-\int_0^t \lambda(s) ds} \quad (3.5)$$

We can have many hazard and survival functions, in this coursework we are going to face, as already said, Gompertz, Exponential and piecewise exponential Plot them, together with their survival functions.


```
N=1000;
%delta_t=[3 6 9 12 15 18 21 22]/12; % i divide the months by
    the total
delta_t=[0.25 0.5 0.75 1 1.25 1.5 1.75 2];
% amount of the months in a year. I prefer to do this with
    Matlab, instead
%on writing manually the amount because of 22.
n_d=[2 5 9 12 16 20 25 29];
S=1-n_d/N; %n_d corresponds to the sum of the indicator
    functions
```

We start by putting in the data, i chose to write manually the data in months because, otherwise, the results didn't fit the ones given by the book. Then, with the method of least squares, i estimated the parameters λ for exponential and piecewise exponential (same value) and λ and γ for the Gompertz. These parameters where found in order to match the empirical S , which we previously found.

In order to do this, i created the functions that i needed and then, with the help of `lsqcurvefit()` i obtained the parameters.

```
Gom_lambda=@(t,gom_lambda,gom_gamma)gom_lambda*gom_gamma*exp(
    gom_gamma*t)
```

```
% Lambda with an Exponential
```

```
Exponential=@(lambda,delta_t)(exp(-lambda*delta_t));
Lambda_exp=lsqcurvefit(Exponential,0,delta_t,S);
```

```
% Piecewise exponential
```

```
Piecewise_exp=@(lambda,delta_t)(exp(-lambda*delta_t));
```

```
% Lambda and Gamma with Gompertz
```

```
Gompertz=@(lambda_gamma,delta_t)(exp(lambda_gamma(1)*(1-exp(
    lambda_gamma(2)*delta_t))));
lambda_gamma=lsqcurvefit(Gompertz,[0.02 0.3],delta_t,S);
Lambda_gom=lambda_gamma(1);
Gamma_gom=lambda_gamma(2);
```

Then, I plotted the hazard function as in figure 3.35 of Roncalli's book. These are just the formula of $\lambda(t)$ and $S(t)$ applied, the one that works differently is the piecewise exponential, where we had to use the different of the logs on each path divided by the differences between the `delta_t`, starting

from the `piecewise_lambda` value(1) that we had to calculate separately

```
%Piecewise constant
piecewise_lambda=zeros(1,8);
piecewise_lambda(1)=-log(S(1))/(0.25)*100;
for i=2:8
    piecewise_lambda(i)=(log(S(i-1))-log(S(i)))/(delta_t(i)-
        delta_t(i-1))*100;
end
t=linspace(0,2.50,100)
% Hazard function
figure;
plot(t,Gom_lambda(t,Lambda_gom,Gamma_gom)*100,"Color",redcolor
    ,"LineStyle","--")
hold on
plot([0,2.50],[Lambda_exp,Lambda_exp]*100,"Color",bluecolor)
hold on
for i=1:8
    plot([delta_t(i)-0.25,delta_t(i)],[piecewise_lambda(i),
        piecewise_lambda(i)],"Color",greencolor)
    hold on
end
plot([2,2.50],[piecewise_lambda(8),piecewise_lambda(8)],"Color
    ",greencolor)
```

And the Survival function.

Again, the same goes for the survival, where we had to build a cycle in order to plot the piecewise exponential, the "function" applied in the paths is the same as the exponential, even though it has a different name.

```
% Survival function
hold off
figure;
plot(t,Exponential(Lambda_exp*100,t),"Color",bluecolor)
hold on
plot(t,Gompertz([Lambda_gom*100,Gamma_gom],t),"Color",redcolor
    ,"LineStyle","--")
hold on
piece_t=0;
for i=1:8
    piece_t=linspace(delta_t(i)-0.25,delta_t(i),100)
    plot(piece_t,Piecewise_exp(piecewise_lambda(i),piece_t),"
        Color",greencolor)
    hold on
end
```

The final results are these:

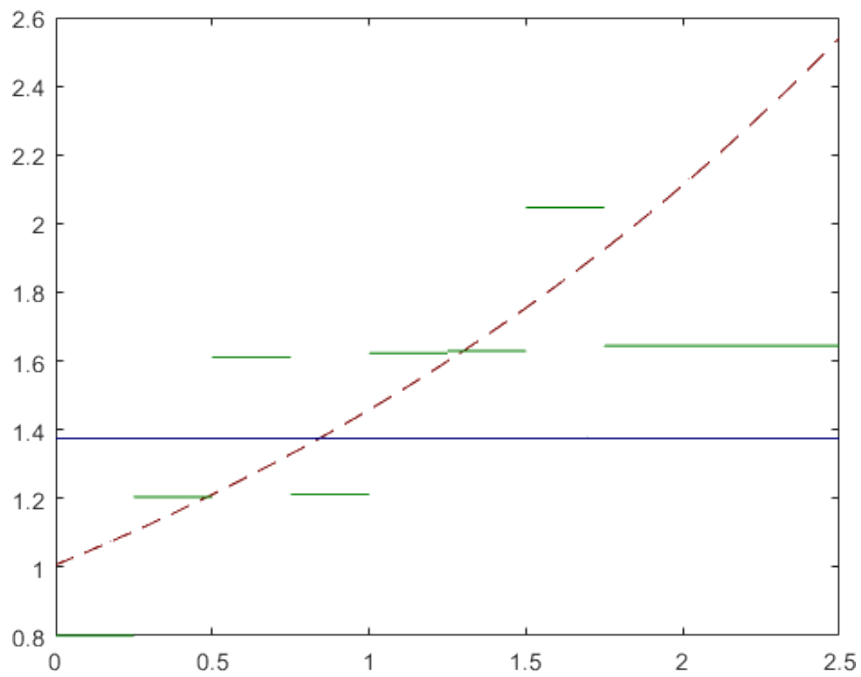


Figure 3.1: Estimated hazard function

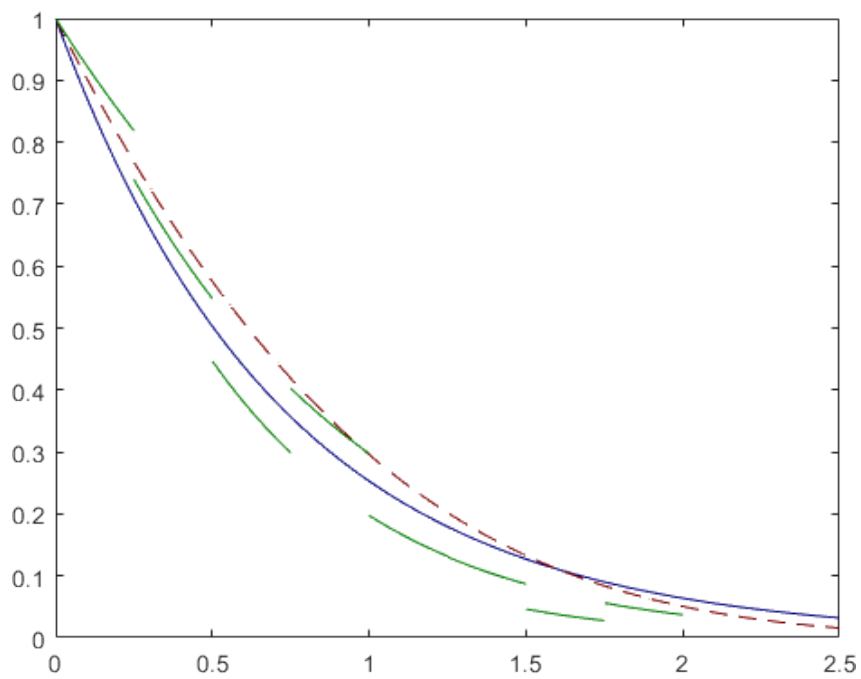


Figure 3.2: Survival function

3.4 Credit migration matrix

Credit migration matrix comes to the rescue when it is required to model the matrix of transition probabilities between different classes, due to the risk classes in the IRB framework. As a consequence, we model credit rating migration by using the Markov Chain.

We can generalize these "paths" in years (e.g. up to 150 as the example) by applying the Chapman-Kolmogorov equation:

$$p_{i,j}^{(n+m)} = \sum_{k=1}^K p_{i,k}^{(n)} \cdot p_{k,j}^{(m)}, \quad \forall n, m > 0. \quad (3.6)$$

This is, basically, what we do in the following code when i use the matlab function "dot()".

3.4.1 Piecewise constant hazard function

In this part, we are requested to estimate the piecewise constant hazard function $\lambda_i(t)$ for all the rating classes and t, up to 150, reproducing figure 3.35 of Roncalli's book.

I start by putting in the data.

```
CMM= [92.82 6.50 0.56 0.06 0.06 0.00 0.00 0.00;
      0.63 91.87 6.64 0.65 0.06 0.11 0.04 0.00;
      0.08 2.26 91.66 5.11 0.61 0.23 0.01 0.04;
      0.05 0.27 5.84 87.74 4.74 0.98 0.16 0.22;
      0.04 0.11 0.64 7.85 81.14 8.27 0.89 1.06;
      0.00 0.11 0.30 0.42 6.75 83.07 3.86 5.49;
      0.19 0.00 0.38 0.75 2.44 12.03 60.71 23.50;
      0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00];
CMM=CMM/100;
```

Then, for the sake of completeness, i obtained the matrix at 2 and at 5. This is for 2. For each i and j of the matrix i multiply the value that they represent with the dot function, that is the dot product of two vectors and that represents the Chapman-Kolmogorov equation in this case.

```
CMM1=CMM;
for i=1:8
    for j=1:8
        CMM2(i,j)=dot(CMM1(i,:),CMM(:,j));
    end
end
```

```
CMM2=CMM2*100;
```

And this is for 5. This is more elaborated than 2, because we have to "save" each path in order to multiply it by the initial value. However, 2 might be a special case of this algorithm, if we make the first for go from 2 to 2, so "for k=2:2"

```
CMM1=CMM;
for k=2:5

for i=1:8
    for j=1:8
        CMM5(i,j)=dot(CMM1(i,:),CMM(:,j));

    end
end
CMM1=CMM5;
end
CMM5=CMM5*100;
```

These are the final results, for 2:

CMM21	CMM22	CMM23	CMM24	CMM25	CMM26	CMM27	CMM28
-----	-----	-----	-----	-----	-----	-----	-----
86.2	12.02	1.47	0.18	0.11	0.01	0	0
1.17	84.59	12.23	1.51	0.18	0.22	0.07	0.02
0.16	4.17	84.47	9.23	1.31	0.51	0.04	0.11
0.1	0.63	10.53	77.66	8.11	2.1	0.32	0.56
0.08	0.24	1.6	13.33	66.79	13.77	1.59	2.6
0.01	0.21	0.61	1.29	11.2	70.03	5.61	11.03
0.29	0.04	0.68	1.37	4.31	17.51	37.34	38.45
0	0	0	0	0	0	0	100

And for 5, we can notice that AAA that remains in AAA starts getting lower.

CMM51	CMM52	CMM53	CMM54	CMM55	CMM56	CMM57	CMM58
-----	-----	-----	-----	-----	-----	-----	-----
69.23	23.85	5.49	0.96	0.31	0.12	0.02	0.03
2.35	66.96	24.14	4.76	0.86	0.62	0.13	0.19
0.43	8.26	68.17	17.34	3.53	1.55	0.18	0.55

0.24	1.96	19.69	56.62	13.19	5.32	0.75	2.22
0.17	0.73	5.17	21.23	40.72	20.53	2.71	8.74
0.07	0.47	1.73	4.67	16.53	44.95	5.91	25.68
0.38	0.24	1.37	2.92	7.13	18.51	9.92	59.53
0	0	0	0	0	0	0	100

Then, i generalized it and obtained the figure 3.35. I had to delete the DDD because, otherwise, it would have raised up an error because we were going to divide by 0.

```

t=150;
CMM1=CMM;
hazard=zeros(7,t-1)

for k=2:t
for i=1:8
    for j=1:8
        CMM2(i,j)=dot(CMM1(i,:),CMM(:,j));

    end
end
hazard(1:7,k-1)=log((1-CMM1(1:7,8))./(1-CMM2(1:7,8)))
CMM1=CMM2;
end

%figure;
tiledlayout(2,2);

nexttile

plot(hazard(1,:)*10000,"Color",bluecolor)
hold on
plot(hazard(2,:)*10000,"Color",redcolor,"LineStyle","--")
legend("AAA","AA","Location","best")

nexttile
plot(hazard(3,:)*10000,"Color",bluecolor)
hold on
plot(hazard(4,:)*10000,"Color",redcolor,"LineStyle","--")
legend("A","BBB","Location","best")

nexttile
plot(hazard(5,:)*10000,"Color",bluecolor)
hold on
plot(hazard(6,:)*10000,"Color",redcolor,"LineStyle","--")
legend("BBB","B","Location","best")

```

```
nexttile
plot(hazard(7,:)*10000,"Color",bluecolor)
legend("CCC","Location","best")
```

This is the final output, we can notice that for good initial ratings, hazard rates are low for short maturities and increase with time. For bad initial ratings, we obtain the opposite effect, because the firm can only improve its rating if it did not default.

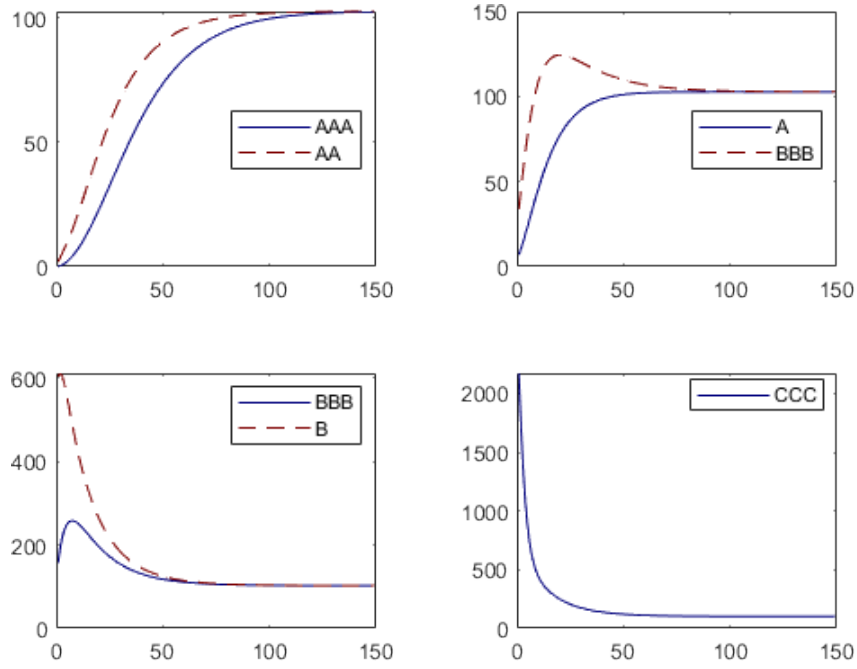


Figure 3.3: Figure 3.35 of Roncilli, made by me

3.4.2 (optional) Markov Generator

Starting from the given data of the previous subsection, i replicated tables 3.37-3.40. To obtain the table 3.39 i set up a value of n that goes up to 3, in order to have the same results.

In the Table 3.38, i put a for up to 3, in order to match the data of the book. In Israel et al. paper it means that we are going to write it as:

$$\tilde{Q} = (P - I) - (P - I)^2/2 + (P - I)^3/3 \quad (3.7)$$

Furthermore, in the last two i tried to avoid as much as possible cycles, like for or while, where it was possible; that's why i used Off_diag, which is the

part that is not on the diagonal (i.e. $i \neq j$), the rest is just an application of the given formulas.

```

CMM= [92.82 6.50 0.56 0.06 0.06 0.00 0.00 0.00;
      0.63 91.87 6.64 0.65 0.06 0.11 0.04 0.00;
      0.08 2.26 91.66 5.11 0.61 0.23 0.01 0.04;
      0.05 0.27 5.84 87.74 4.74 0.98 0.16 0.22;
      0.04 0.11 0.64 7.85 81.14 8.27 0.89 1.06;
      0.00 0.11 0.30 0.42 6.75 83.07 3.86 5.49;
      0.19 0.00 0.38 0.75 2.44 12.03 60.71 23.50;
      0.00 0.00 0.00 0.00 0.00 0.00 0.00 100.00];
CMM=CMM/100;

%Table 3.37
MG_37=round(logm(CMM)*10000,2);

%Table 3.38
I=eye(8);
MG_38=zeros(8);
for i=1:3
    MG_38=MG_38+ ((-1)^(i+1))*((CMM-I)^i)/i;
end
MG_38=round(MG_38*10000,2);

%Table 3.39
off_I=1-I
Off_Diag=MG_37.*off_I;
Diag=(( repmat(sum((Off_Diag<0).*Off_Diag,2),1,8))+MG_37).*I
MG_39=max(Off_Diag,0)+Diag;

%Table 3.40

G=abs(sum(MG_37.*I,2))+sum(max(Off_Diag,0),2);
B=sum(max(-Off_Diag,0),2);

%G= repmat(G,1,8);
B= repmat(B,1,8);
%G_cond=G>0;
Zero_cond=(MG_37.*off_I>=0);
MG_40=ones(8)
for i=1:8
    if(G(i)==0)
        MG_40(i,:)=MG_37(i,:);
    else

```



```

MG_40(i,:) = round(Zero_cond(i,:) .* (MG_37(i,:) - (B(i,:) .*
    abs(MG_37(i,:) ./ G(i,:)))) , 2)
end
end

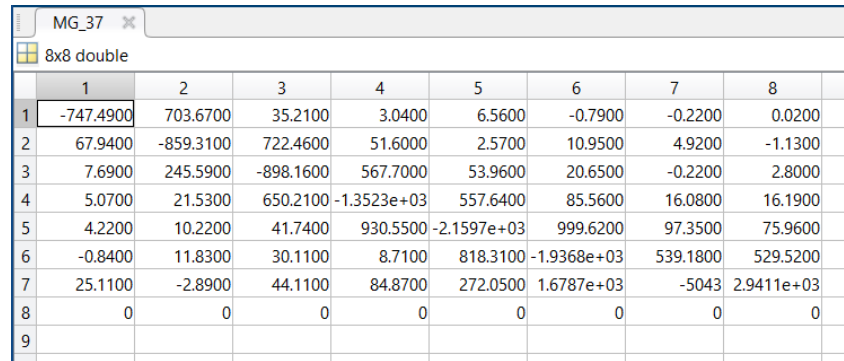
```

These are the final output obtained with array2table(MG_37), array2table(MG_38), array2table(MG_39), array2table(MG_40). (the 1 at the end is given by array2table)

MG_371	MG_372	MG_373	MG_374	MG_375	MG_376	MG_377
-----	-----	-----	-----	-----	-----	-----
-747.49	703.67	35.21	3.04	6.56	-0.79	-0.79
67.94	-859.31	722.46	51.6	2.57	10.95	10.95
7.69	245.59	-898.16	567.7	53.96	20.65	20.65
5.07	21.53	650.21	-1352.28	557.64	85.56	85.56
4.22	10.22	41.74	930.55	-2159.67	999.62	999.62
-0.84	11.83	30.11	8.71	818.31	-1936.82	-1936.82
25.11	-2.89	44.11	84.87	272.05	1678.69	1678.69
0	0	0	0	0	0	0
MG_381	MG_382	MG_383	MG_384	MG_385	MG_386	MG_387
-----	-----	-----	-----	-----	-----	-----
-747.38	703.25	35.66	2.92	6.51	-0.77	-0.77
67.91	-858.98	721.78	52.06	2.54	10.93	10.93
7.7	245.36	-897.33	566.63	54.52	20.52	20.52
5.07	21.72	648.9	-1349.45	554.59	87.12	87.12
4.21	10.15	43.2	925	-2150.09	992.32	992.32
-0.74	11.78	29.75	11.56	812.21	-1924.31	-1924.31
24.54	-2.57	43.52	83.2	274.01	1629.89	1629.89
0	0	0	0	0	0	0
MG_391	MG_392	MG_393	MG_394	MG_395	MG_396	MG_397
-----	-----	-----	-----	-----	-----	-----
-748.5	703.67	35.21	3.04	6.56	0	0
67.94	-860.44	722.46	51.6	2.57	10.95	10.95
7.69	245.59	-898.38	567.7	53.96	20.65	20.65

5.07	21.53	650.21	-1352.28	557.64	85.56	
4.22	10.22	41.74	930.55	-2159.67	999.62	
0	11.83	30.11	8.71	818.31	-1937.66	5
25.11	0	44.11	84.87	272.05	1678.69	-50
0	0	0	0	0	0	
MG_401	MG_402	MG_403	MG_404	MG_405	MG_406	M
-----	-----	-----	-----	-----	-----	---
-747.99	703.19	35.19	3.04	6.56	0	
67.9	-859.87	721.99	51.57	2.57	10.94	
7.69	245.56	-898.27	567.63	53.95	20.65	
5.07	21.53	650.21	-1352.28	557.64	85.56	
4.22	10.22	41.74	930.55	-2159.67	999.62	
0	11.83	30.1	8.71	818.13	-1937.24	
25.1	0	44.1	84.85	271.97	1678.21	-5
0	0	0	0	0	0	

In order to have a clear vision on the matrix, which may not happen by printing them, i took a screenshot of the matrixes.



	1	2	3	4	5	6	7	8
1	-747.4900	703.6700	35.2100	3.0400	6.5600	-0.7900	-0.2200	0.0200
2	67.9400	-859.3100	722.4600	51.6000	2.5700	10.9500	4.9200	-1.1300
3	7.6900	245.5900	-898.1600	567.7000	53.9600	20.6500	-0.2200	2.8000
4	5.0700	21.5300	650.2100	-1.3523e+03	557.6400	85.5600	16.0800	16.1900
5	4.2200	10.2200	41.7400	930.5500	-2.1597e+03	999.6200	97.3500	75.9600
6	-0.8400	11.8300	30.1100	8.7100	818.3100	-1.9368e+03	539.1800	529.5200
7	25.1100	-2.8900	44.1100	84.8700	272.0500	1.6787e+03	-5043	2.9411e+03
8	0	0	0	0	0	0	0	0
9								

MG_38								
8x8 double								
	1	2	3	4	5	6	7	8
1	-747.3800	703.2500	35.6600	2.9200	6.5100	-0.7700	-0.2000	0.0100
2	67.9100	-858.9800	721.7800	52.0600	2.5400	10.9300	4.8300	-1.0700
3	7.7000	245.3600	-897.3300	566.6300	54.5200	20.5200	-0.1400	2.7400
4	5.0700	21.7200	648.9000	-1.3495e+03	554.5900	87.1200	16	16.0500
5	4.2100	10.1500	43.2000	925	-2.1501e+03	992.3200	98.3600	76.8500
6	-0.7400	11.7800	29.7500	11.5600	812.2100	-1.9243e+03	523.4400	536.3100
7	24.5400	-2.5700	43.5200	83.2000	274.0100	1.6299e+03	-4.9424e+03	2.8898e+03
8	0	0	0	0	0	0	0	0
9								

MG_39								
8x8 double								
	1	2	3	4	5	6	7	8
1	-748.5000	703.6700	35.2100	3.0400	6.5600	0	0	0.0200
2	67.9400	-860.4400	722.4600	51.6000	2.5700	10.9500	4.9200	0
3	7.6900	245.5900	-898.3800	567.7000	53.9600	20.6500	0	2.8000
4	5.0700	21.5300	650.2100	-1.3523e+03	557.6400	85.5600	16.0800	16.1900
5	4.2200	10.2200	41.7400	930.5500	-2.1597e+03	999.6200	97.3500	75.9600
6	0	11.8300	30.1100	8.7100	818.3100	-1.9377e+03	539.1800	529.5200
7	25.1100	0	44.1100	84.8700	272.0500	1.6787e+03	-5.0459e+03	2.9411e+03
8	0	0	0	0	0	0	0	0
9								

MG_40								
8x8 double								
	1	2	3	4	5	6	7	8
1	-747.9900	703.1900	35.1900	3.0400	6.5600	0	0	0.0200
2	67.9000	-859.8700	721.9900	51.5700	2.5700	10.9400	4.9200	0
3	7.6900	245.5600	-898.2700	567.6300	53.9500	20.6500	0	2.8000
4	5.0700	21.5300	650.2100	-1.3523e+03	557.6400	85.5600	16.0800	16.1900
5	4.2200	10.2200	41.7400	930.5500	-2.1597e+03	999.6200	97.3500	75.9600
6	0	11.8300	30.1000	8.7100	818.1300	-1.9372e+03	539.0600	529.4100
7	25.1000	0	44.1000	84.8500	271.9700	1.6782e+03	-5.0444e+03	2.9402e+03
8	0	0	0	0	0	0	0	0
9								

Then, i obtained the probability density functions. I did it up to 300, as the paper does, i created a for where in each interaction i obtain an MG1_i, and from that i take the given values that i need at the column 8 (for instance AAA at column 8), where we have the default.

```
bluecolor=[0, 0, 125/255];
redcolor=[126/255, 0, 0];
greencolor=[0, 126/255,0];
```

```
t=1:300;
AAA=MG_39(1,:);
AA=MG_39(1,:);
```

```
%C=AAA(1,1)
f_AAA_1=zeros(1,300);
f_AA_1=zeros(1,300);

f_AAA_2=zeros(1,300);
f_AA_2=zeros(1,300);
for i=1:300
MG1_i=(MG_39/10000)*expm(i*(MG_39/10000));
f_AAA_1(i)=MG1_i(1,8);
f_AA_1(i)=MG1_i(2,8);

MG2_i=(MG_40/10000)*expm(i*(MG_40/10000));
f_AAA_2(i)=MG2_i(1,8);
f_AA_2(i)=MG2_i(2,8);
end

%figure;
plot(t,f_AAA_2,"Color",bluecolor)
hold on
plot(t,f_AAA_1,"Color",redcolor)
legend("AAA with second approach","AAA with first approach")
hold off
plot(t,f_AAA_2,"Color",bluecolor)
hold on
plot(t,f_AA_2,"Color",redcolor)
legend("AAA with second approach","AA with second approach")
```

And I obtained these two graphs:

The first graph is made of two function obtained from different approached that are quite similar, that's why i also added the legend. We can notice that, first of all, as I already said, the two methods more or less are coincident with AAA. Furthermore, the AA curve is translated to the left, with an higher maximum than AAA approach, so that, at, more or less $t=50$.

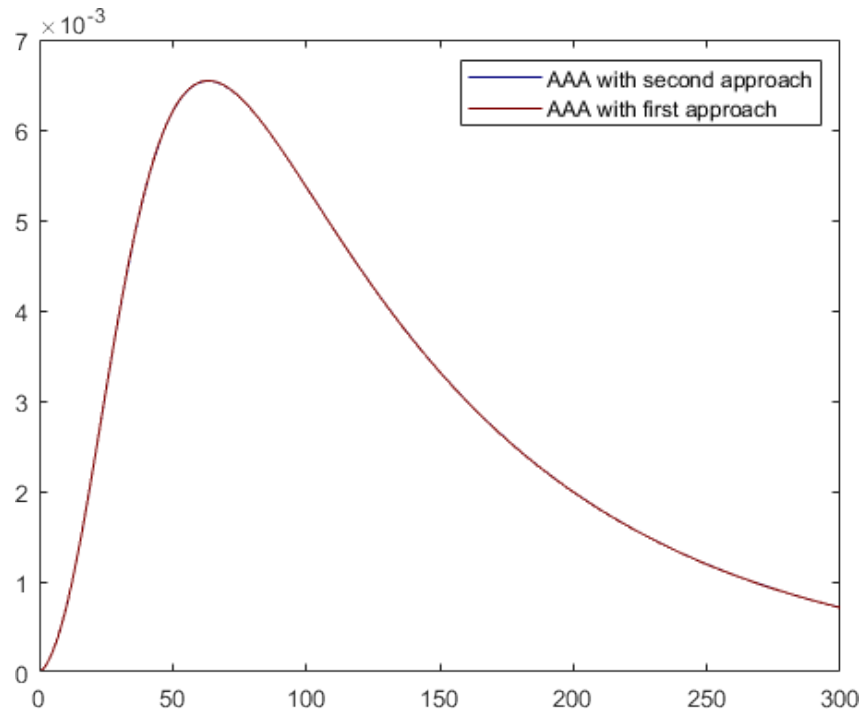


Figure 3.4: AAA obtained with second and first approach

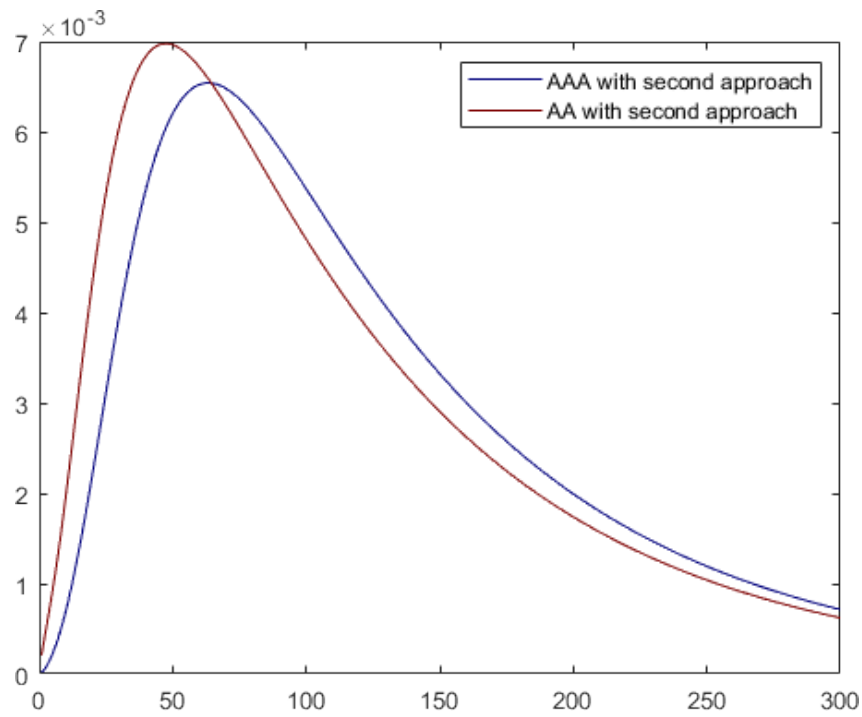


Figure 3.5: AAA and AA obtained with second approach

Chapter 4

Default correlation

We are in the copula environment, which are used in order to describe joint default probabilities. One of the main issues in risk management is, indeed, the aggregation risk.

One can assume independence, which is an assumption that simplifies everything, but you may lose some diversification, by missing out risks that are negatively correlated, furthermore, there might be positive dependence. The easiest solution is to use a multivariate normal distribution. Given the data in Example 36 of Roncalli's book, which we have already seen, and assuming a Basel-one factor model, we are asked to plot the loss distribution, compute its VaR and ES when the Gaussian Copula correlation is $\rho = 0, 0.2, 0.5$ by doing a Monte Carlo simulation with 100000 scenarios.

I put in the data and i did a similar thing that I have done previously: I adapt it to the number of simulation, i do it for LGD and for lambda.

```
n=100000;
T=5;
EAD=1000;
LGD=[0 0.10 0.20 0.25 0.30 0.40 0.50 0.60 0.70 0.75 0.80 0.90
      1];
p=[0.01 0.02 0.10 0.25 0.10 0.02 0 0.02 0.10 0.25 0.10 0.02
    0.01]; %They sum up to 1
lambda=[10 10 25 25 50 100 250 500 500 1000]/10000;
final_LGD=repelem(LGD,p*n*10)
final_LGD=reshape(final_LGD(randperm(numel(final_LGD))),10,n)
lambda= repmat(lambda',[1,n])
```

Then, I simulate all the random variables, that will be used to obtain τ and its correspondent indicator function.

```
X=repmat(normrnd(0,1,[1,n]),[10,1])
```

```
epsilon=normrnd(0,1,[10,n]);
```

After this part, i divide my problem in 3 parts:

- $\rho = 0$
- $\rho = 0.2$
- $\rho = 0.5$

Starting from $\rho = 0$:

```
rho=0;
u=normcdf(sqrt(rho).*X +sqrt(1-rho).*epsilon);

tau=[-log(u)./lambda]
indicator=(tau<=T)

L=EAD.*final_LGD.*indicator
L_tot=sum(L,1)
%figure;
histogram(L_tot)
title("\rho =0")
```

Then, moving to $\rho = 0.2$:

```
rho=0.2;
u=normcdf(sqrt(rho).*X +sqrt(1-rho).*epsilon);

tau=[-log(u)./lambda]
indicator=(tau<=T)

L=EAD.*final_LGD.*indicator
L_tot2=sum(L,1)
figure;
histogram(L_tot2)
title("\rho =0.2")
```

Finally, 0.5:

```
rho=0.5;
u=normcdf(sqrt(rho).*X +sqrt(1-rho).*epsilon);

tau=[-log(u)./lambda]
indicator=(tau<=T)

L=EAD.*final_LGD.*indicator
L_tot=sum(L,1)
```



```
figure;  
histogram(L_tot)  
title("\rho =0.5")
```

Obtaining these pictures:

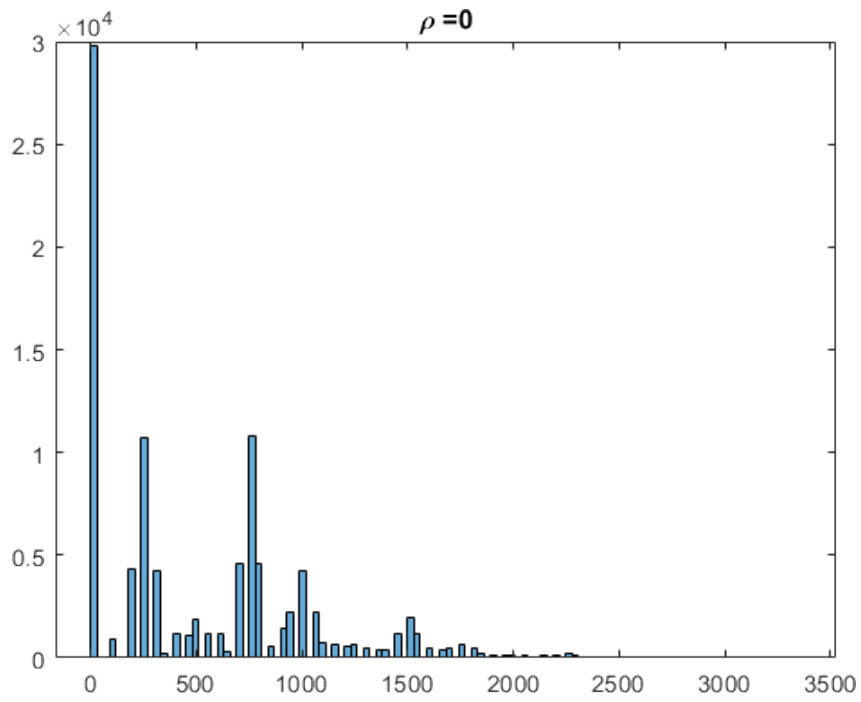
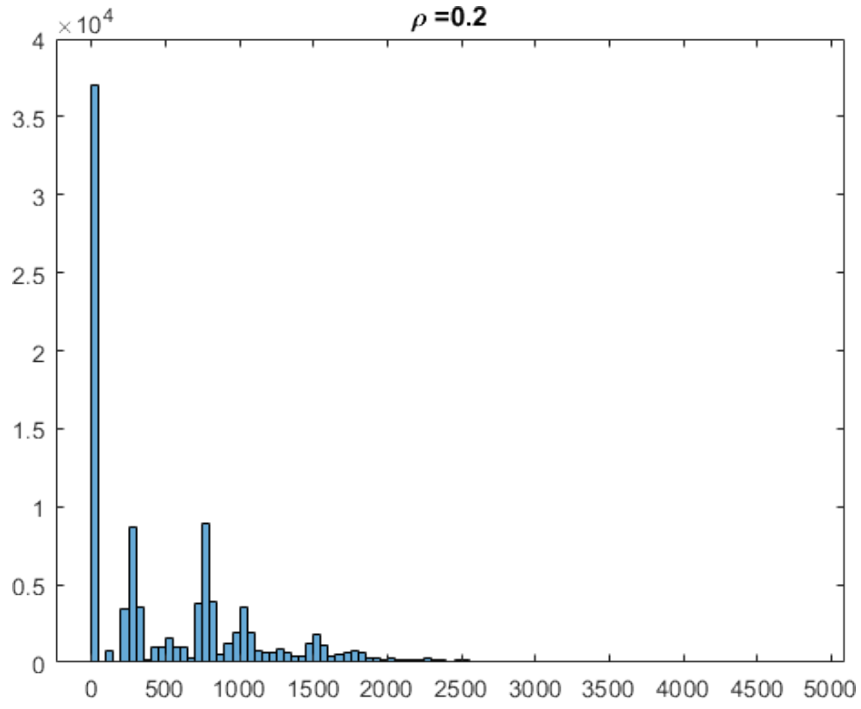
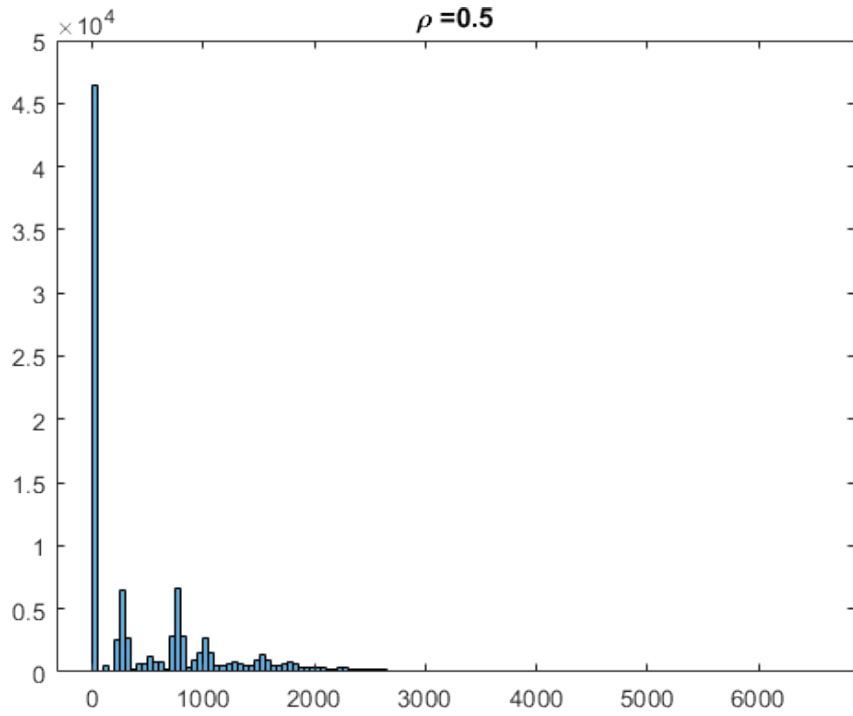


Figure 4.1: $\rho = 0$

Figure 4.2: $\rho = 0.2$ Figure 4.3: $\rho = 0.5$

VaR and Expected Shortfall

- Value at Risk: also known as "VaR" is a statistics based on the distribution of losses that measures the severity of risk. The VaR at the confidence level α is the smallest number l such that the probability that the loss L exceeds l is no larger than $(1-\alpha)$. VaR, in general, does not enjoy the subadditivity property, that might be helpful for diversification and being able to have bounds. It is said that it is not coherent. We also do not know what happens above this threshold and does not take into account very well the tails
- Expected Shortfall: $ES_\alpha = \frac{1}{1-\alpha} \int_\alpha^1 VaR_u(L) du$. It averages VaR values above the confidence level α and it is coherent

The VaR are:

```
%rho=0
VaR 99 is 2050.000000
VaR 99.5 is 2300.000000
VaR 99.9 is 2600.000000
%rho=0.2
VaR 99 is 2400.000000
VaR 99.5 is 2650.000000
VaR 99.9 is 3300.000000
%rho=0.5
VaR 99 is 3000.000000
VaR 99.5 is 3400.000000
VaR 99.9 is 4200.000000
```

And the code to obtain them is:

```
%rho=0
Var_99_0=prctile(L_tot,99)
Var_995_0=prctile(L_tot,99.5)
Var_999_0=prctile(L_tot,99.9)
fprintf("VaR 99 is %0f \n",Var_99_0)
fprintf("VaR 99.5 is %0f \n",Var_995_0)
fprintf("VaR 99.9 is %0f \n",Var_999_0)
%rho=0.2
Var_99_02=prctile(L_tot2,99)
Var_995_02=prctile(L_tot2,99.5)
Var_999_02=prctile(L_tot2,99.9)
fprintf("VaR 99 is %0f \n",Var_99_02)
```

```

fprintf("VaR 99.5 is %0f \n",Var_995_02)
fprintf("VaR 99.9 is %0f \n",Var_999_02)
%rho= 0.5
Var_99_05=prctile(L_tot5,99)
Var_995_05=prctile(L_tot5,99.5)
Var_999_05=prctile(L_tot5,99.9)
fprintf("VaR 99 is %0f \n",Var_99_05)
fprintf("VaR 99.5 is %0f \n",Var_995_05)
fprintf("VaR 99.9 is %0f \n",Var_999_05)

```

The ES are:

```

%rho=0
ES 99 is 2360.664840
ES 99.5 is 2515.435636
ES 99.9 is 2920.779780
%rho=0.2
ES 99 is 2803.247447
ES 99.5 is 3081.794595
ES 99.9 is 3674.426426
%rho=0.5
ES 99 is 3552.898148
ES 99.5 is 3928.523524
ES 99.9 is 4760.091091

```

and the code to obtain them:

```

%rho=0
alpha99_0=linspace(99,100,1000)
ES99_0=mean(prctile(L_tot,alpha99_0))
alpha995_0=linspace(99.5,100,1000)
ES995_0=mean(prctile(L_tot,alpha995_0))
alpha999_0=linspace(99.9,100,1000)
ES999_0=mean(prctile(L_tot,alpha999_0))
fprintf("ES 99 is %0f \n",ES99_0)
fprintf("ES 99.5 is %0f \n",ES995_0)
fprintf("ES 99.9 is %0f \n",ES999_0)
%rho=0.2
alpha99_02=linspace(99,100,1000)
ES99_02=mean(prctile(L_tot2,alpha99_02))
alpha995_02=linspace(99.5,100,1000)
ES995_02=mean(prctile(L_tot2,alpha995_02))
alpha999_02=linspace(99.9,100,1000)
ES999_02=mean(prctile(L_tot2,alpha999_02))
fprintf("ES 99 is %0f \n",ES99_02)

```

```
fprintf("ES 99.5 is %0f \n",ES995_02)
fprintf("ES 99.9 is %0f \n",ES999_02)
%rho=0.5
alpha99_5=linspace(99,100,1000)
ES99_5=mean(prctile(L_tot5,alpha99_5))
alpha995_5=linspace(99.5,100,1000)
ES995_5=mean(prctile(L_tot5,alpha995_5))
alpha999_5=linspace(99.9,100,1000)
ES999_5=mean(prctile(L_tot5,alpha999_5))
fprintf("ES 99 is %0f \n",ES99_5)
fprintf("ES 99.5 is %0f \n",ES995_5)
fprintf("ES 99.9 is %0f \n",ES999_5)
```


Chapter 5

Counterparty Credit Risk

In this problem we have bought at time 0 100 ATM call options of an asset initially valued 100 \$ with one-year maturity. The asset evolves as a geometric brownian motion with $r = 5\%$ and $\sigma = 20\%$ under the risk neutral measure.

At time $t_0 = 0.5$ the asset price is 114.77\$.

Using 10000 simulated scenarios, estimate the pdf of MtM (1) and of the exposure at time 1, $e(1)$.

We start by plugging in the data:

```
Q=100;
S_0=100;
K=100;
T=1;
r=0.05;
sigma=0.2;

t_0=0.5;
S_t0=114.77;
N=10000;
```

Now, we exploit the fact that the asset price evolves as a GBM, so that we divide our simulation in two part:

- 0-0.5, where the price at t_0 is 114.77 because it is given, so we simulate the possible prices evolution until we obtain the correct one. As a consequence, we will end up with a price, which is 114.77.
- 0.5-1, where we start from 114.77, which is the price at $t=0.5$. Then, we simulate 100000 scenarios of our prices that will make 100 time steps until they reach time 1, as a consequence we will obtain 100000 prices

at time 1, which is equal to the number of scenario.

This is the code for the time interval 0-0.5:

```
n=100;
t=T/2;
dt=t/n;
S_05=zeros(1,n);
drift= repmat(dt,[1,n]);
increment_drift=cumsum(drift)
while(round(S_05(100),2)~=S_t0)
diffusion=normrnd(0,sqrt(dt),[1,n]);
increment_diff=cumsum(diffusion)
increment=exp((r-(sigma^2)/2)*increment_drift+sigma*
    increment_diff)
S_05=S_0*increment;
end
```

And this is the code for the time interval 0.5-1:

```
diffusion_simul_incr=cumsum(normrnd(0,sqrt(dt),[N,n]),2);
drift_simul_incr= repmat(t_0+increment_drift,N,1)
increment_simul=exp(sigma*diffusion_simul_incr+(r-(sigma^2)/2)
    *drift_simul_incr)
S_simul=S_t0*increment_simul;

plot(linspace(0.5,1,n),S_simul);
hold on
plot(linspace(0,0.5,n),S_05);
```

This gives us this plot: Then, we apply the Black and Scholes Formula, in order to obtain the call price at time 0:

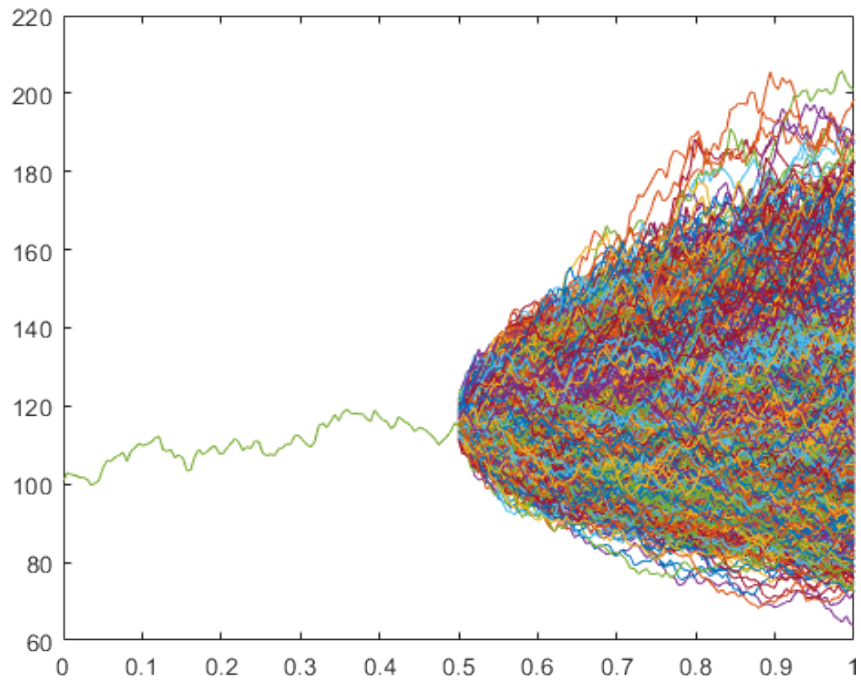
```
%    apply Black and Scholes to obtain C_0
d1=(log(S_0 / K)+(r + 0.5*(sigma)^2)*T)/(sigma *sqrt(T));
d2=d1-sigma*sqrt(T);

C_0= S_0 * normcdf(d1) - K*exp(-r*T)*normcdf(d2);
```

Then, I extract the prices at time 1, in order to obtain the Call price at time 1, and the MtM(1) and e_1.

```
S_1=S_simul(:,100);
C_1=max(S_1-S_0,0);

MtM_1=Q*(C_1-C_0);
e_1=max(MtM_1,0);
```

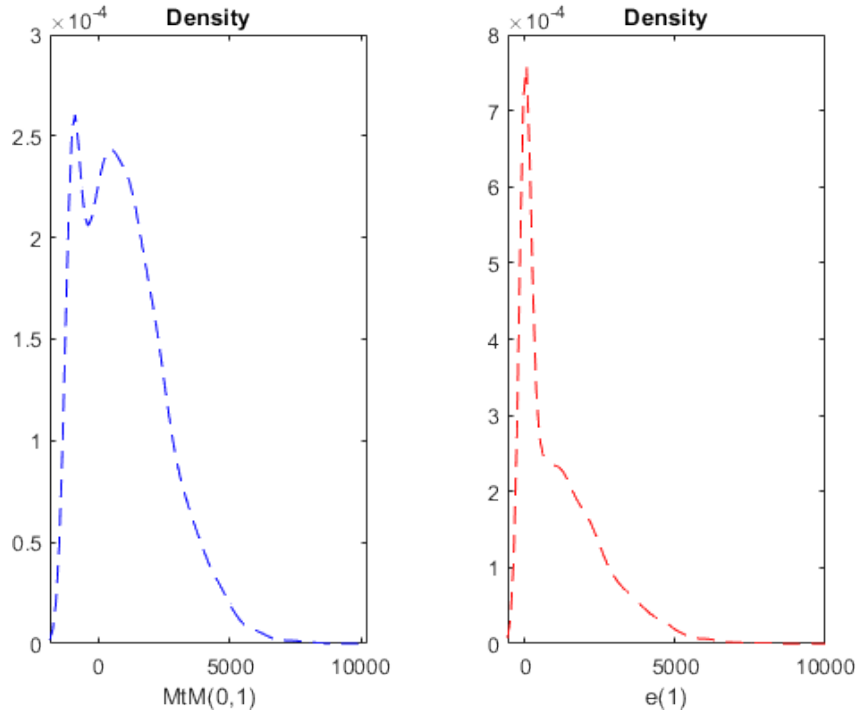
Now, I am going to plot them

```

tiledlayout(1,2);
nexttile
[f, xi] = ksdensity(MtM_1);
plot(xi, f, "LineStyle","--", "Color","blue");
xlabel("MtM(0,1)")
title("Density")
nexttile
[f2, xi2] = ksdensity(e_1);
plot(xi2, f2, "LineStyle","--", "Color","red");
xlabel("e(1)")
title("Density")

```

The result is



Now, I switch to the point of view of t_0 . Again, we apply Black and Scholes, with Initial price 114.77 and an interval of time equal to 0.5:

```
S_t0=114.77
t_0=0.5

d1=(log(S_t0 / K)+(r + 0.5*(sigma)^2)*t_0)/(sigma *sqrt(t_0));
d2=d1-sigma*sqrt(t_0);

C_t0= S_t0 * normcdf(d1) - K*exp(-r*t_0)*normcdf(d2);
```

By applying the same formula as before, we obtain the new MtM and e. From the last one, we obtain EE and PFE on 95% and 99%.

```
MtM_t1=Q*(C_1-C_t0);
e_t1=max(MtM_t1,0);

EE=mean(e_t1)
PFE_95=prctile(e_t1,95)
PFE_99=prctile(e_t1,99)
fprintf("Expected Exposure is %0f \n",EE)
fprintf("Potential Future Exposure at 95 is %0f \n",PFE_95)
fprintf("Potential Future Exposure at 99 is %0f \n",PFE_99)
```

Expected Exposure is 735.532927
Potential Future Exposure at 95 is 3173.236180
Potential Future Exposure at 99 is 4639.736149

Now, given and expected LGE=50% and a default probability of the counterparty equal to 0.02% over the 1-year horizon, we want to compute the CVA at time 0. Which, by applying this code, is:

```
E_LGD=0.5;  
PD=0.02;  
CVA=C_0*(1-E_LGD*PD)
```

CVA at time 0 is 10.346078