CCruncher - Technical Document

Gerard Torrent Gironella

Version 1.8 - R727

May 20, 2011

Abstract

CCruncher computes the credit risk of massive loan portfolios taking into account default correlations between economic sectors. This is done by determining the probability distribution of portfolio loss using the Monte Carlo method. The simulation of obligor's default times is achieved using a copula that combines the survival functions with the observed sectorial correlations. Risk is obtained using the usual risk statistics, including Expected Loss, Value at Risk and Expected Shortfall.

Keywords: credit risk, Monte Carlo, survival functions, Gaussian copula, t-Student copula, value at risk, expected shortfall, block matrix.

Contents

1 Introduction

This introduction was originally published in *Ercim News 78* [14] as a contribution to the special theme *Mathematics for Finance and Economy*. There have been done some minor changes in order to adapt to the current situation.

Financial institutions need tools to simulate their credit portfolios in the same way that they simulate their equity portfolios. CCruncher allows each payment in a massive loan portfolio to be simulated, taking into account the probability of default for each creditor and the correlations between them.

While the portfolios of loans given to SMEs and portfolios of corporate bonds do not have the glamour of the equity or foreign exchange markets, they are nevertheless the core business of many financial institutions. Such portfolios tend to have values of thousands of millions of dollars and may be composed of tens of thousands of small loans.

At first glance, credit portfolios may seem harmless because payments are based on pre-agreed dates. Nothing could be farther from the truth. Defaults frequently occur in clusters, affecting entire sectors of industry. The correlation between defaults is the main difficulty in risk assessment, and this is the rationale behind many of the approaches to credit risk valuation. The impact of the correlation on the loss distribution function is notorious, and causes an asymmetrical distribution function with long tails.

CCruncher is an open-source project for the simulation of massive portfolios of loans where the unique risk is the default risk. It is addressed to financial institutions searching for a well-documented and efficient tool. It consists of an engine that runs in batch mode and is designed to be integrated into the risk management systems of financial institutions for risk assessment and stress testing. The method used to determine the distribution of losses in the portfolio is the Monte Carlo algorithm, because it allows us to consider the majority of variables involved, such as the date and amount of each payment. This approach is conceptually equivalent to the valuation of portfolios of equities and derivatives using the Monte Carlo method. In the case of market risk, the price of the underlying assets is simulated using a geometric Brownian motion with a given volatility and trend. In the credit risk case, obligors' defaults are simulated using a copula with given survival rates and correlations.

Sklar's theorem states that any multivariate distribution can be decomposed into two parts, the marginal distributions and a copula that reflects the structure of

dependencies between components. This result is important because it allows us to separate the default probabilities from the default correlations.

CCruncher implements two of the most commonly used copulas in finance, the Normal and T-Student copulas. Because the default time of each obligor is modelled as a random variable, the size of the copula can be over 50.000, where each component represents a obligor. Simulating low-dimensional copulas poses no problem, but many of the algorithms operating in low dimensions are unsuitable for large dimensions. To solve this problem, the Cholesky decomposition was adapted to operate with a large correlation matrix composed of blocks, in order to reduce the memory space and the number of operations involved.

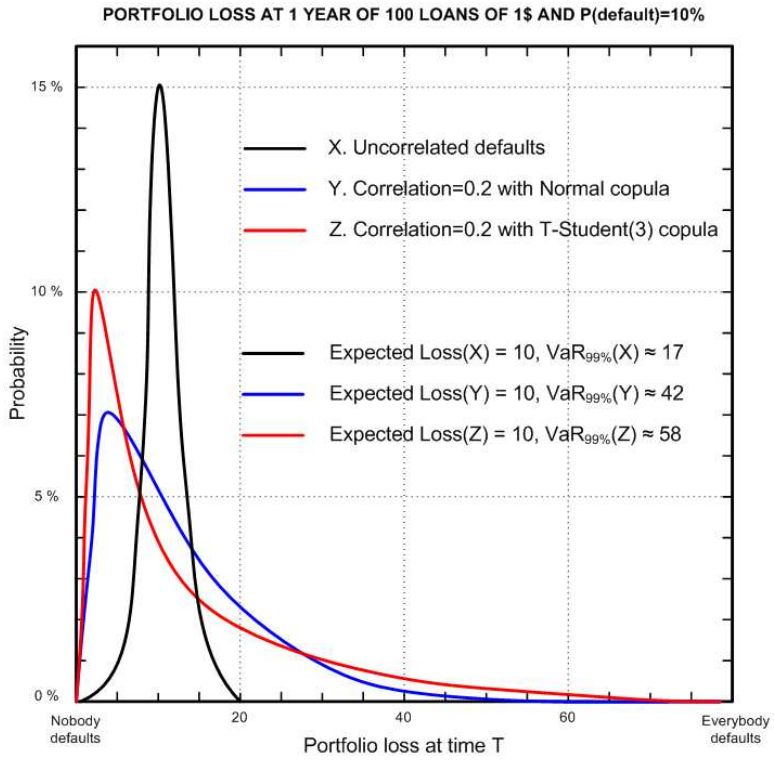

Figure 1: Impact of correlation and copula selection on portfolio loss distribution

Each Monte Carlo run simulates the default time for each debtor by combining a copula sampling with the survival functions. This default time is used to determine the loss caused by the default, mitigated by an estimated recovery at a given time. The default amount is discounted to the present value using the interest rate provided by the yield curve. The sum of all losses provides the loss value of the portfolio for that simulation. The completion of thousands of

simulations allows us to obtain the loss distribution function and compute the usual risk indicators, such as the Value at Risk or Expected Shortfall. Depending on the portfolio size and the required accuracy, the computational cost may be high.

CCruncher is currently a one-man project with no financial support, and is looking for collaborators with expertise in applied and/or theoretical financial mathematics to participate in a GPL (GNU General Public License) structure. The CCruncher system has been tested against both simple hand-made portfolios and large automatically generated portfolios. Future work is being considered to add a graphical interface to make the tool available to the public, such as portfolio bond managers.

2 Parameters

This section explains the input parameters that are required by CCruncher.

2.1 Ratings and Survival Functions

A credit rating estimates the probability that an individual or corporation will pay back a loan. A poor rating indicates a high risk of defaulting. Each rating has an associated survival function. This function indicates the probability that a obligor with an initial rating (X) will not have defaulted at time t.

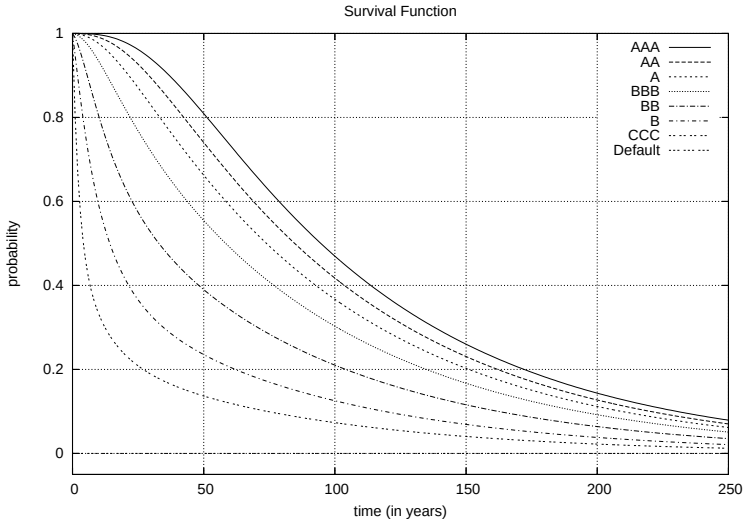

Figure 2: Survival functions

The survival data is often not available, and where it is available, it is often inconsistent or not representative. For this reason, it is better to work with the transition matrix, assuming that the rating transitions follow a Markov model. Appendix A.1 shows how to determine the survival functions using the transition matrix.

2.2 Sectors and Correlations

The risk of a credit portfolio depends crucially on correlations between economic sectors. These sectors are groupings of companies that react similarly to economic conditions. Some examples of sectors include: energy, financial, technology, media and entertainment, utilities, and health care. Default correlations between sectors measure the default time dependence between the defined sectors. This can be expressed in tabular form:

	Sector_1	$\dots$	Sector_m
Sector_1	$\rho_{1,1}$	$\dots$	$\rho_{1,m}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sector_m	$\rho_{1,m}$	$\dots$	$\rho_{m,m}$

$$\rho_{i,j} = \text{Corr}(\text{Sector}_i, \text{Sector}_j)$$

Note that diagonal elements are different than 1 because they are the default time correlations between members of the same sector. The elements outside of the diagonal are default time correlations between elements of distinct sectors.

Appendix A.3 demonstrates a simple method to obtain an estimation of the correlations table based on the historical information about defaults.

2.3 Portfolio

The portfolio is composed by the obligors. Each obligor has an initial rating, belongs to a sector and has one or more assets. Each asset is defined by its creation date, expected exposure and recovery at certain dates.

Exposure. Amount that lender can lose in case of obligor's default on the indicated date. When cashflow events are know in advance (see Figure 3) exposure at date t can be computed as:

$$\text{exposure}_t = \sum_{i \geq t} \text{cashflow}_i$$

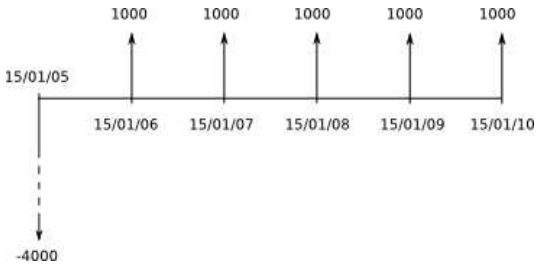

Figure 3: Asset cashflow example

Recovery. It is a ratio that indicates the portion of loss that will be recovered in the case of default at a fixed time. This value takes into account guarantees, including the cost of taking legal action, and recovery rate based on historical data.

If a obligor defaults, the loss is the exposure weighted by the recovery rate at the default time.

$$\text{loss}_t = (1 - \text{recovery}_t) \cdot \text{exposure}_t$$

3 Resolution

This section explains how CCruncher computes the credit risk of a portfolio. Figure 4 shows the schema followed to simulate the portfolio losses.

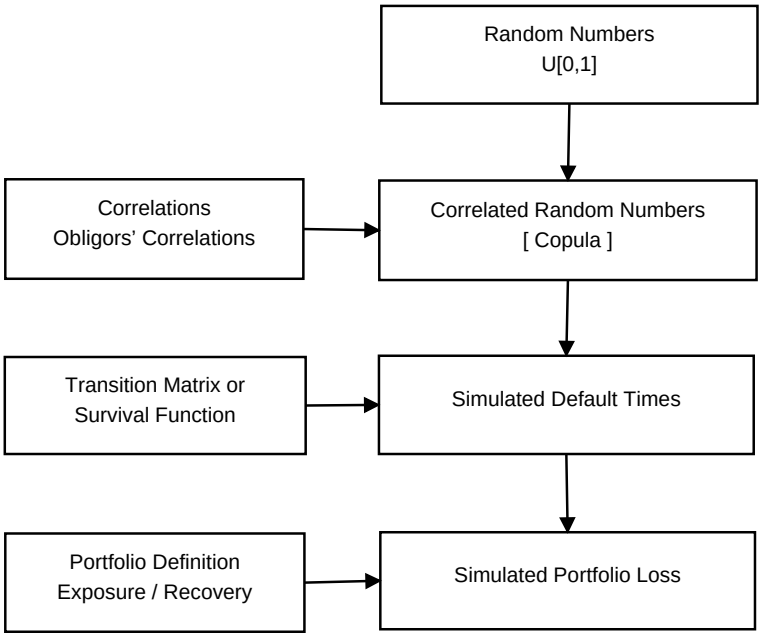

Figure 4: Monte Carlo simulation schema

3.1 Obligors correlation matrix

We need to translate from default correlations between sectors to default correlations between obligors. Suppose we have n obligors and m sectors. Each obligor belongs to a sector. $\rho_{i,j}$ are the estimated default time correlations between sectors:

	Sector_1	$\dots$	Sector_m
Sector_1	$\rho_{1,1}$	$\dots$	$\rho_{1,m}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Sector_m	$\rho_{1,m}$	$\dots$	$\rho_{m,m}$

$$\rho_{i,j} = \text{Corr}(\text{Sector}_i, \text{Sector}_j)$$

Sort the obligors so that those who belong to sector 1 are at the beginning and those in sector m are at the end. Then, create the obligors correlation matrix, taking as the correlation between two obligors the correlation between their sectors (see Figure 5).

$$\Sigma = \begin{pmatrix}
1 & \cdots & \rho_{1,1} & & \rho_{1,k} & \cdots & \rho_{1,k} & & \rho_{1,m} & \cdots & \rho_{1,m} \\
\vdots & \ddots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
\rho_{1,1} & \cdots & 1 & & \rho_{1,k} & \cdots & \rho_{1,k} & & \rho_{1,m} & \cdots & \rho_{1,m} \\
 & & & \ddots & & & & & & & \\
\rho_{1,k} & \cdots & \rho_{1,k} & & 1 & \cdots & \rho_{k,k} & & \rho_{k,m} & \cdots & \rho_{k,m} \\
\vdots & \ddots & \vdots & & \vdots & \ddots & \vdots & & \vdots & & \vdots \\
\rho_{1,k} & \cdots & \rho_{1,} & & \rho_{k,k} & \cdots & 1 & & \rho_{k,m} & \cdots & \rho_{k,m} \\
 & & & & & & & \ddots & & & \\
\rho_{1,m} & \cdots & \rho_{1,m} & & \rho_{k,m} & \cdots & \rho_{k,m} & & 1 & \cdots & \rho_{m,m} \\
\vdots & \ddots & \vdots & & \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\
\rho_{1,m} & \cdots & \rho_{1,m} & & \rho_{k,m} & \cdots & \rho_{k,m} & & \rho_{m,m} & \cdots & 1
\end{pmatrix}$$

Figure 5: Obligors correlation matrix

Observe that this is a correlation matrix (symmetric, $|\rho_{i,j}| \leq 1$, $\rho_{i,i} = 1$) that is composed of blocks. This matrix will be decomposed using the Cholesky algorithm, so it must be positive-definite.

3.2 Monte Carlo simulation

Monte Carlo methods are a class of computational algorithms for simulating the behaviour of various physical and mathematical systems. Each simulation consists of computing a random default time for each obligor and then calculating the loss of the portfolio. Default times must fulfil two conditions: the obligor's survival functions and the obligor's correlation matrix. The link between the marginal distributions (survival functions) and the dependence structure (correlations) is achieved with a copula.

3.2.1 Correlated random numbers generation

A copula [12] [15] is a multivariate random variable in which each component is uniform, $U[0,1]$. Each simulation requires a set of random numbers between $[0,1]$ and is correlated with the obligors correlation matrix, Σ.

$$\left|\begin{array}{c|c|c|c}
u_{11} & u_{12} & \cdots & u_{1n} \\
u_{21} & u_{22} & \cdots & u_{2n} \\
\vdots & \vdots & \vdots & \vdots \\
u_{k1} & u_{k2} & \cdots & u_{kn}
\end{array}\right|$$

n number of obligors
k number of simulations
$u_{.i} \sim U[0,1]$
$Corr(u_{.i}, u_{.j}) = \Sigma_{ij}$

Two copula generators, the Gaussian and the t-Student copulas, are implemented. These two copulas have good properties: they are commonly used in finance, exist algorithms to simulate them for any correlation matrix, and these algorithms work for large dimensions (see Appendices A.4 and A.5).

Erik Kole et al. [4] have compared some copulas used in risk management. They observe that the t-Student copula can be considered to model the dependence between the different elements of a portfolio:

> "For a portfolio consisting of stocks, bonds and real estate, these tests provide clear evidence in favor of the Student's t copula, and reject both the correlation-based Gaussian copula and the extreme value-based Gumbel copula. In comparison with the Student's t copula, we find that the Gaussian copula underestimates the probability of joint extreme downward movements, while the Gumbel copula overestimates this risk. Similarly we establish that the Gaussian copula is too optimistic on diversication benefits, while the Gumbel copula is too pessimistic. Moreover, these differences are significant."

3.2.2 Default times simulation

Given a obligor k and a random number ($u_k \in [0, 1]$) (generated using a copula), we simulate the default time considering the inverse of the obligor survival function at u_k [3]. Figure 6 shows this graphically.

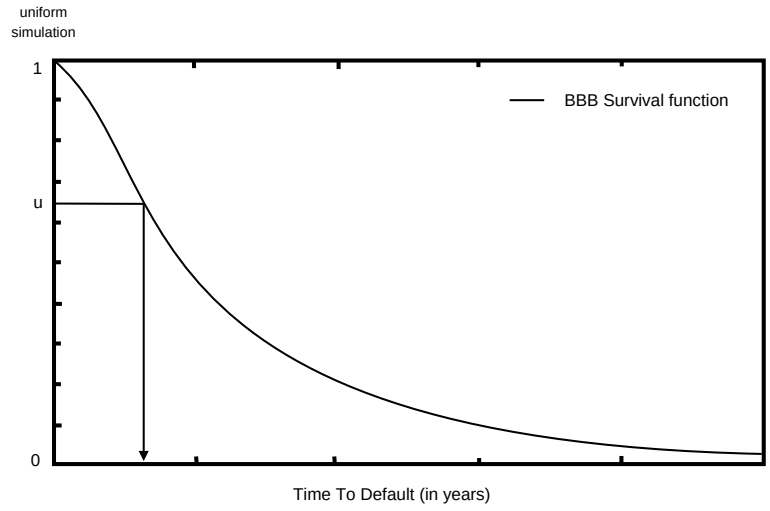

Figure 6: Default time generation with initial rating BBB

3.2.3 Portfolio loss evaluation

At this point, all obligors have a simulated default time. The loss caused by any obligor is the exposure at the default time weighted by the recovery at the default time. To obtain the simulated value of the portfolio loss, add up all obligors' losses:

$$\text{PortfolioLoss} = \sum_{i=1}^{N} \text{ObligorLoss}_i$$

where N is the number of obligors in the portfolio. For each obligor, we use the simulated default time, t_i, described in the previous section, to compute the obligor loss:

$$\text{ObligorLoss}_i = \begin{cases} \sum_{j=1}^{K_i} (1 - \text{recovery}_{t_i}^j) \cdot \text{exposure}_{t_i}^j & \text{if } t_i \leq T \\ 0 & \text{if } t_i > T \end{cases}$$

where K_i is the number of assets subscribed to obligor i, $\text{recovery}_{t_i}^j$ is the recovery of the asset j of the obligor i at time t_i, $\text{exposure}_{t_i}^j$ is the exposure of the asset j of the obligor i at time t_i and T is the time horizon where risk is computed.

3.3 Risk computation

After N simulations (eg. 20000, 500000 or more) we have a list of numbers, $x_1, ..., x_N$, in which each number represents a simulated portfolio loss. The more values there are, the more accuracy there is in the results. All risk statistics (e.g., Expected Loss) have an error margin that will be estimated.

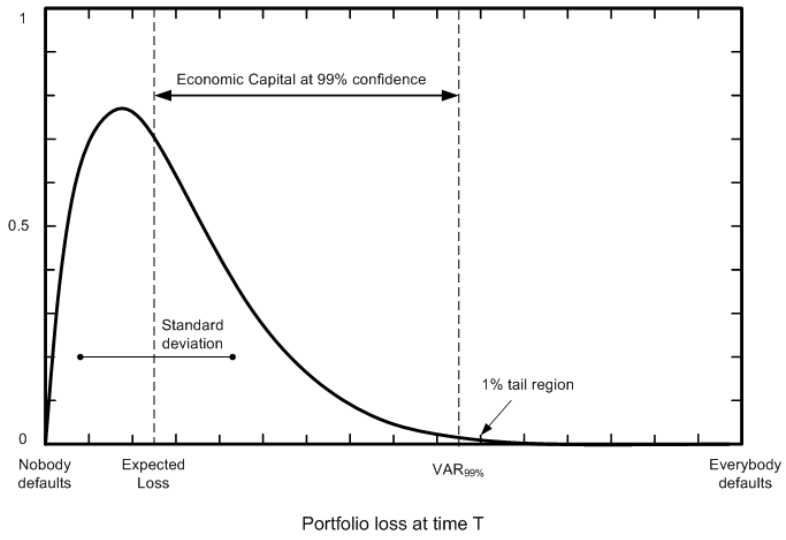

Figure 7: Portfolio loss at time T

3.3.1 Expected Loss

The expected loss is the mean of the portfolio loss distribution. The Central Limit Theorem [13] grants that:

$$\mu = \widehat{\mu} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \frac{\widehat{\sigma}}{\sqrt{N}}$$

where α is the error confidence level, ϕ^{-1} is the $N(0,1)$ inverse cumulative distribution function, and $\widehat{\mu}$ and $\widehat{\sigma}$ are the mean and standard deviation estimators, respectively:

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\widehat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \widehat{\mu})^2}$$

3.3.2 Portfolio Loss Standard Deviation

Another usual risk statistic is the standard deviation of the portfolio loss. The Central Limit Theorem [13] grants that:

$$\sigma = \widehat{\sigma} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \frac{\widehat{\sigma}}{\sqrt{2N}}$$

where α is the error confidence level, ϕ^{-1} is the $N(0,1)$ inverse cumulative distribution function, and $\widehat{\sigma}$ is the standar deviation estimator that was defined previously.

3.3.3 Value At Risk

Value at Risk [9] is the most commonly used risk value. We call it VaR_β where β is the VaR confidence level (e.g., VaR at 95%). VaR is another way to say quantile. Thus, $\text{VaR}_\beta = q_\beta = \inf\{x | F(x) \geq \beta\}$ and

$$\text{VaR}_\beta = \widehat{q_\beta} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \text{stderr}(q_\beta)$$

where α is the error confidence level, β is the VaR confidence level, ϕ^{-1} is the $N(0,1)$ inverse cumulative distribution function, $\widehat{q_\beta}$ is the quantile estimator, and $\text{stderr}(q_\beta)$ is the estimation of the standard error.

$$\widehat{q_\beta} = x_{k:N}$$

where

- k fulfils $\frac{k}{N} \leq \beta < \frac{k+1}{N}$
- $x_{k:N}$ is the k-th element of ascendant-sorted values.

Determine $\text{stderr}(q_\beta)$ using the Maritz-Jarret method described in [7]:

$$\begin{aligned} M &= [N\beta + 0.5] \\ a &= M - 1 \\ b &= N - M \\ W_i &= B(a, b, \tfrac{i+1}{N}) - B(a, b, \tfrac{i}{N}) \\ C_k &= \textstyle\sum_{i=1}^{N} W_i \cdot x_i^k \end{aligned}$$

where $[x]$ is the integer part of x and $B(a,b,x)$ is the incomplete beta function:

$$B(a,b,x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$$

Thus,

$$\text{stderr}(q_\beta) = \sqrt{C_2 - C_1^2}$$

3.3.4 Expected Shortfall

VaR is not a distance because it does not fulfil the sub-additive property [8], $VaR(A+B) \nleq VaR(A) + VaR(B)$. Similar to VaR, Expected Shortfall is a consistent risk measure [2]. It can be described as the average of the $\beta\%$ worst losses:

$$ES_\beta = \widehat{ES_\beta} \pm \phi^{-1}\left(\frac{1-\alpha}{2}\right) \cdot \text{stderr}(ES_\beta)$$

where α is the error confidence level, β is the ES confidence level, ϕ^{-1} is the $N(0,1)$ inverse cumulative distribution function, $\widehat{ES_\beta}$ is the ES estimator and stderr(ES_β) is the estimation of the standard error.

If we select the simulation portfolio loss values $(x_1, ..., x_N)$ that are bigger than VaR_β,

$$y_1, y_2, y_3, \cdots, y_K \qquad \text{where} \quad y_i > VaR_\beta$$

then,

$$\widehat{ES_\beta} = \frac{1}{K}\sum_{i=1}^{K} y_i$$

$$\text{stderr}(ES_\beta) = \frac{\sqrt{\frac{1}{K-1}\sum_{i=1}^{K}\left(y_i - \widehat{ES_\beta}\right)^2}}{\sqrt{K}}$$

3.3.5 Economic Capital

Compute the Economic Capital at confidence level β as:

$$\text{Economic Capital} = VaR_\beta - \text{Expected Loss}$$

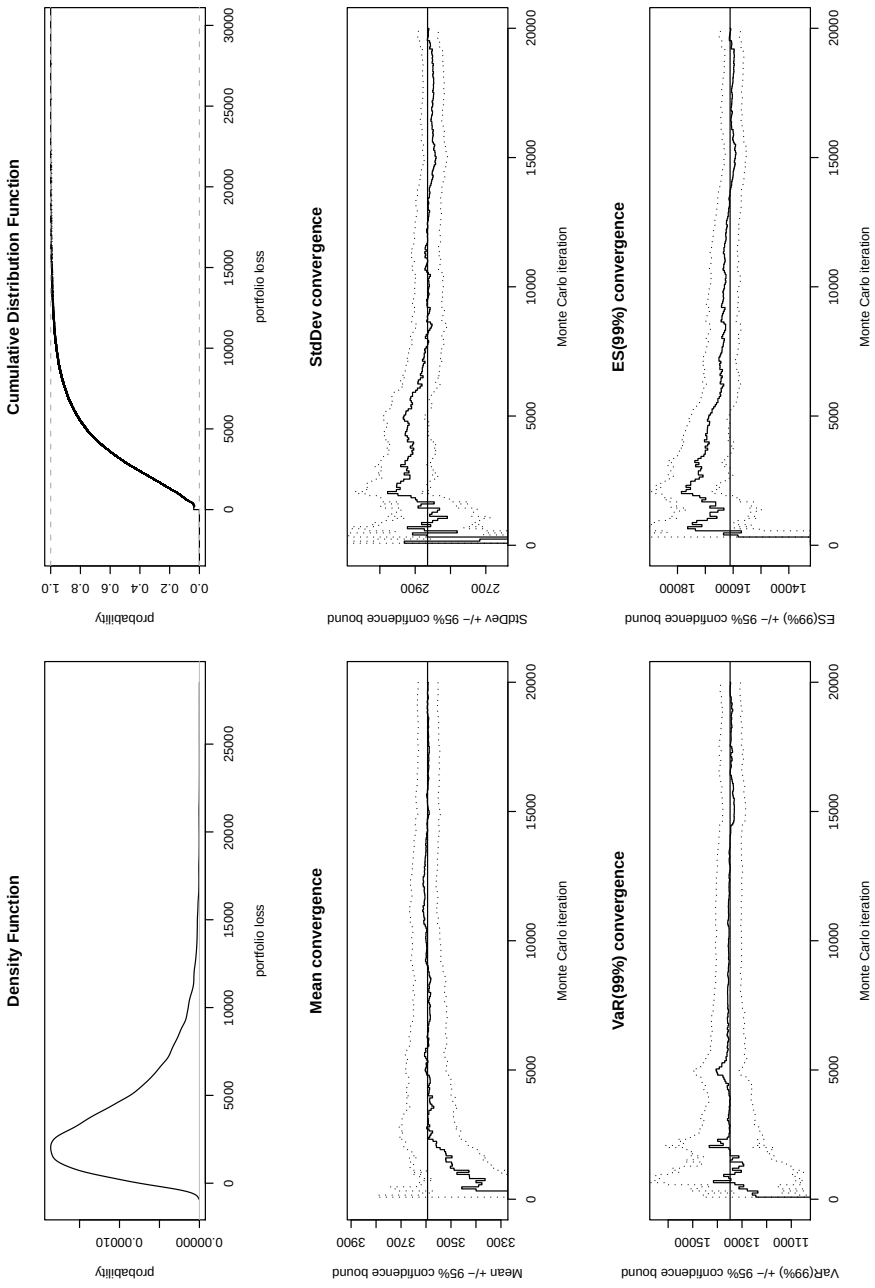

Figure 8: CCruncher results

4 Other considerations

CCruncher takes into consideration other concepts that have not been discussed up to this point with the purpose of simplifying the content.

4.1 Stochastic exposure

In most cases, exposure is not known in advance and is estimated based on historical data. CCruncher allows to describe the exposure of an asset at time *t* using the lognormal, exponential, uniform, gamma, and truncated normal distributions. When obligor defaults, CCruncher simule a value from the exposure distribution at time *t* to determine the asset loss.

4.2 Stochastic recovery

In most cases, recovery is not known in advance and is estimated based on historical data. CCruncher allows to describe the recovery of an asset at time *t* using the beta distribution, the most popular recovery distribution. When obligor defaults, CCruncher simule a value from the recovery distribution at time *t* to determine the asset loss.

4.3 Risk aggregation

CCruncher simulates the whole portfolio loss at time T. It also allows (in the same execution) the simulation of subportfolios by defining one or more aggregators. Consequently, one can determine which obligors (or products, branches, regions, assets, etc.) increase the risk of the portfolio. Each subportfolio has its own list of simulated values that can be processed to obtain the risk indicators for this subportfolio.

4.4 Yield curve

CCruncher can consider that the value of money decreases with time, following a curve (e.g., a yield curve). CCruncher use this curve to compute the present value of the exposures. Allowed methods are simple interest, compound interest and continuous interest. The formulas used to compute the present value are:

$$C_{t_0} = \frac{C_t}{1+r\cdot(t-t_0)} \quad \text{(Simple)} \qquad C_{t_0} = \frac{C_t}{(1+r)^{(t-t_0)}} \quad \text{(Compound)} \qquad C_{t_0} = \frac{C_t}{exp(r\cdot(t-t_0))} \quad \text{(Continuous)}$$

Where t_0 is the current date, C_{t_0} is the present value of the exposure at time t_0, C_t is the exposure at time t, r is the interest rate given by the yield curve at time t and $t - t_0$ is the number of years between t_0 and t.

4.5 Antithetic technique

The random number generation using a copula is time intensive. The Gaussian and t-Student copulas are symmetric, which means that $(u_1, u_2, \cdots, u_n)$ is equiprobable to $(1 - u_1, 1 - u_2, \cdots, 1 - u_n)$. In CCruncher's antithetic mode, each copula generation is used twice, $(u_1, u_2, \cdots, u_n)$ and $(1-u_1, 1-u_2, \cdots, 1-u_n)$, reducing by half the number of generated copulas.

4.6 Parallel computing

Monte Carlo problems are embarrassingly parallel problems. In the jargon of parallel computing, an embarrassingly parallel workload (or embarrassingly parallel problem) is one for which no particular effort is needed to segment the problem into a very large number of parallel tasks, and there is no essential dependency (or communication) among those parallel tasks.

In CCruncher, the master sends a distinct seed to each slave to initialize its RNG and waits for results (the simulated values of the portfolio loss) from slaves that store them in a file. When one of the two stop criteria is achieved (the maximum execution time or the maximum number of simulations are exceeded), the master sends a stop signal to the slaves and leaves.

5 Numerical example

We compute the credit risk at $T = 30$ years for the following portfolio.

Id	Obligor name	Rating	Sector	Recovery	Assets
O1	Future Skyline	AA	Construction	80%	A1
O2	Ramses builds	BBB	Construction	75%	A2, A3
O3	Gramdateresi	AA	Consumer goods	60%	A4
O4	Tantor Pinori	BB	Consumer goods	90%	A5, A6, A7
O5	White & Rebolan	B	Services	30%	A8
O6	Adign Consulting	BBB	Services	60%	A9
O7	Advanced Engineering	AAA	Services	50%	A10
O8	Sigmacle Research	CCC	Services	60%	A11
O9	Loglament Asia	AA	Services	50%	A12

Table 1: Portfolio composition

The following table lists the asset events in months from the current date.

Month	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
6	10$				100$							70$
12		50$				20$	80$		20$			
36											30$	
60		30$			10$		20$	80$				
90						30$						
120	15$		40$		30$							
180					40$	70$						
216		10$			50$							80$
240				100$								
312	100$		40$		70$				70$			
360						15$					40$	
384	90$				90$					100$	90$	

Table 2: Asset events

This example assumes that there are no survival functions, but we have the one-year transition matrix T_1 (extracted from [6]).

	AAA	AA	A	BBB	BB	B	CCC	Default
AAA	90.81	8.33	0.68	0.06	0.12	0.00	0.00	0.00
AA	0.70	90.65	7.79	0.64	0.06	0.14	0.02	0.00
A	0.09	2.27	91.05	5.52	0.74	0.26	0.01	0.06
BBB	0.02	0.33	5.95	86.93	5.30	1.17	0.12	0.18
BB	0.03	0.14	0.67	7.73	80.53	8.84	1.00	1.06
B	0.00	0.11	0.24	0.43	6.48	83.46	4.07	5.21
CCC	0.22	0.00	0.22	1.30	2.38	11.24	64.86	19.78
Default	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

Table 3: One-year transition matrix

The sectorial default time correlations are:

	Construction	Consumer goods	Services
Construction	0.50	0.20	0.30
Consumer goods	0.20	0.60	0.34
Services	0.30	0.34	0.40

Table 4: Sector correlation matrix

5.1 Obtaining the survival functions

Appendix A.1 is used to compute the survival functions with a time resolution of one month. We scale[1] the transition matrix to one month by using $T_{\frac{1}{12}} = T_1^{\frac{1}{12}}$. The one-month transition matrix, $T_{\frac{1}{12}}$, is given by:

$$\begin{pmatrix} 99.1972 & 0.7588 & 0.0320 & 0.0020 & 0.0112 & -0.0011 & -0.0001 & 0.0000 \\ 0.0635 & 99.1745 & 0.7083 & 0.0392 & 0.0015 & 0.0120 & 0.0018 & -0.0007 \\ 0.0074 & 0.2057 & 99.1980 & 0.5102 & 0.0557 & 0.0189 & -0.0001 & 0.0042 \\ 0.0015 & 0.0239 & 0.5507 & 98.8021 & 0.5164 & 0.0871 & 0.0077 & 0.0107 \\ 0.0027 & 0.0113 & 0.0396 & 0.7581 & 98.1558 & 0.8774 & 0.0889 & 0.0664 \\ -0.0007 & 0.0098 & 0.0196 & 0.0111 & 0.6447 & 98.4443 & 0.4463 & 0.4249 \\ 0.0233 & -0.0023 & 0.0170 & 0.1287 & 0.2166 & 1.2298 & 96.4213 & 1.9657 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 100.0000 \end{pmatrix}$$

Note the negative values in the computed one-month transition matrix. These values result from the fact that the original matrix (T_1) is not a regular Markov

[1] CAUTION: calculations should be performed using absolute values, not percentages.

matrix. In these cases, CCruncher regularizes the matrix by applying the algorithm described in Appendix A.2. To simplify this example, the negative values are replaced by 0.

Compute the survival values at month one by using the one-month transition matrix (the d subindex means the default column):

$$S(., 1 \text{ month}) = \vec{1} - (T_{1/12})_{.d} = \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} - \begin{pmatrix} 0.0000 \\ 0.0000 \\ 0.0042 \\ 0.0107 \\ 0.0664 \\ 0.4249 \\ 1.9657 \end{pmatrix} = \begin{pmatrix} 100.000 \\ 100.000 \\ 99.996 \\ 99.989 \\ 99.934 \\ 99.975 \\ 98.034 \end{pmatrix}$$

Then, compute the survival values at month two:

$$S(., 2 \text{ months}) = \vec{1} - (T^2_{1/12})_{.d} = \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} - \begin{pmatrix} 0.0000 \\ 0.0000 \\ 0.0084 \\ 0.0221 \\ 0.1372 \\ 0.8524 \\ 3.8664 \end{pmatrix} = \begin{pmatrix} 100.000 \\ 100.000 \\ 99.992 \\ 99.978 \\ 99.863 \\ 99.148 \\ 96.134 \end{pmatrix}$$

Next, compute the survivals values at month three:

$$S(., 3 \text{ months}) = \vec{1} - (T^3_{1/12})_{.d} = \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} - \begin{pmatrix} 0.0000 \\ 0.0000 \\ 0.0129 \\ 0.0342 \\ 0.2122 \\ 1.2821 \\ 5.7045 \end{pmatrix} = \begin{pmatrix} 100.000 \\ 100.000 \\ 99.987 \\ 99.966 \\ 99.788 \\ 98.718 \\ 94.296 \end{pmatrix}$$

Repeat the same procedure for 360 months to obtain the survival functions shown in Table 5.

Month	AAA	AA	A	BBB	BB	B	CCC
1	100.000	100.000	99.996	99.989	99.934	99.575	98.034
2	100.000	100.000	99.992	99.978	99.863	99.148	96.134
3	100.000	100.000	99.987	99.966	99.788	98.718	94.296
4	100.000	100.000	99.983	99.953	99.709	98.286	92.518
5	100.000	100.000	99.978	99.939	99.626	97.853	90.798
6	100.000	100.000	99.973	99.925	99.539	97.418	89.134
7	100.000	100.000	99.968	99.909	99.448	96.981	87.525
8	100.000	100.000	99.963	99.893	99.353	96.544	85.967
9	100.000	100.000	99.957	99.876	99.255	96.106	84.460
10	100.000	100.000	99.952	99.858	99.153	95.668	83.000
11	100.000	100.000	99.946	99.840	99.048	95.229	81.588
12	100.000	100.000	99.940	99.820	98.940	94.790	80.220
13	100.000	99.999	99.934	99.800	98.828	94.351	78.896
14	100.000	99.998	99.928	99.778	98.714	93.912	77.613
15	100.000	99.997	99.921	99.756	98.596	93.474	76.370
...	...	...	...	...	...	...	...
169	99.213	97.964	95.345	88.888	72.479	50.213	28.155
170	99.200	97.936	95.292	88.793	72.333	50.062	28.073
171	99.187	97.908	95.240	88.698	72.188	49.912	27.992
172	99.173	97.880	95.187	88.604	72.043	49.764	27.911
173	99.159	97.851	95.134	88.509	71.899	49.616	27.832
...	...	...	...	...	...	...	...
308	95.906	92.251	86.516	76.085	56.684	35.993	20.548
309	95.871	92.198	86.445	75.999	56.596	35.924	20.511
310	95.837	92.145	86.375	75.913	56.509	35.855	20.474
311	95.801	92.093	86.304	75.828	56.423	35.787	20.437
312	95.766	92.040	86.234	75.742	56.336	35.719	20.400
...	...	...	...	...	...	...	...
358	93.986	89.478	82.955	71.932	52.638	32.889	18.868
359	93.944	89.419	82.883	71.851	52.563	32.833	18.838
360	93.902	89.361	82.812	71.771	52.488	32.778	18.808

Table 5: Survival functions table

5.2 Creating the obligors correlation matrix

Use Section 3.1 to create the obligors correlation matrix. In this example, the debtors are sorted by sector, and therefore, no reorder is required.

$$\Sigma = \left(\begin{array}{cc|cc|ccccc} 1.00 & 0.50 & 0.20 & 0.20 & 0.30 & 0.30 & 0.30 & 0.30 & 0.30 \\ 0.50 & 1.00 & 0.20 & 0.20 & 0.30 & 0.30 & 0.30 & 0.30 & 0.30 \\ \hline 0.20 & 0.20 & 1.00 & 0.60 & 0.34 & 0.34 & 0.34 & 0.34 & 0.34 \\ 0.20 & 0.20 & 0.60 & 1.00 & 0.34 & 0.34 & 0.34 & 0.34 & 0.34 \\ \hline 0.30 & 0.30 & 0.34 & 0.34 & 1.00 & 0.40 & 0.40 & 0.40 & 0.40 \\ 0.30 & 0.30 & 0.34 & 0.34 & 0.40 & 1.00 & 0.40 & 0.40 & 0.40 \\ 0.30 & 0.30 & 0.34 & 0.34 & 0.40 & 0.40 & 1.00 & 0.40 & 0.40 \\ 0.30 & 0.30 & 0.34 & 0.34 & 0.40 & 0.40 & 0.40 & 1.00 & 0.40 \\ 0.30 & 0.30 & 0.34 & 0.34 & 0.40 & 0.40 & 0.40 & 0.40 & 1.00 \end{array}\right)$$

5.3 Copula initialization

In this example, we simulate the default times using a Gaussian copula. We compute the covariance matrix, Σ', following Steps 1 and 2 in Appendix A.4.

Create the covariance matrix Σ' by mapping each Σ component, ρ_{ij}, to $2sin(\frac{\pi}{6}\rho_{ij})$:

$$\Sigma' = \left(\begin{array}{ccccccccc} 1.0000 & 0.5176 & 0.2091 & 0.2091 & 0.3129 & 0.3129 & 0.3129 & 0.3129 & 0.3129 \\ 0.5176 & 1.0000 & 0.2091 & 0.2091 & 0.3129 & 0.3129 & 0.3129 & 0.3129 & 0.3129 \\ 0.2091 & 0.2091 & 1.0000 & 0.6180 & 0.3542 & 0.3542 & 0.3542 & 0.3542 & 0.3542 \\ 0.2091 & 0.2091 & 0.6180 & 1.0000 & 0.3542 & 0.3542 & 0.3542 & 0.3542 & 0.3542 \\ 0.3129 & 0.3129 & 0.3542 & 0.3542 & 1.0000 & 0.4158 & 0.4158 & 0.4158 & 0.4158 \\ 0.3129 & 0.3129 & 0.3542 & 0.3542 & 0.4158 & 1.0000 & 0.4158 & 0.4158 & 0.4158 \\ 0.3129 & 0.3129 & 0.3542 & 0.3542 & 0.4158 & 0.4158 & 1.0000 & 0.4158 & 0.4158 \\ 0.3129 & 0.3129 & 0.3542 & 0.3542 & 0.4158 & 0.4158 & 0.4158 & 1.0000 & 0.4158 \\ 0.3129 & 0.3129 & 0.3542 & 0.3542 & 0.4158 & 0.4158 & 0.4158 & 0.4158 & 1.0000 \end{array}\right)$$

Next, apply Cholesky to Σ'. Then $\Sigma' = B \cdot B^{\top}$, where:

$$B = \left(\begin{array}{ccccccccc} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.5176 & 0.8556 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.2091 & 0.1179 & 0.9708 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.2091 & 0.1179 & 0.5773 & 0.7805 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.3129 & 0.1764 & 0.2760 & 0.1392 & 0.8806 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.3129 & 0.1764 & 0.2760 & 0.1392 & 0.2172 & 0.8534 & 0.0000 & 0.0000 & 0.0000 \\ 0.3129 & 0.1764 & 0.2760 & 0.1392 & 0.2172 & 0.1688 & 0.8365 & 0.0000 & 0.0000 \\ 0.3129 & 0.1764 & 0.2760 & 0.1392 & 0.2172 & 0.1688 & 0.1382 & 0.8250 & 0.0000 \\ 0.3129 & 0.1764 & 0.2760 & 0.1392 & 0.2172 & 0.1688 & 0.1382 & 0.1170 & 0.8167 \end{array}\right)$$

5.4 Monte Carlo simulation

Each Monte Carlo trial consists of three parts: the copula simulation, the default times simulation and the portfolio loss evaluation.

Copula simulation. To simulate the Gaussian copula, we follow Steps 3 to 5 in Appendix A.4. To simulate the multivariate $\vec{Z} \sim N(0, \Sigma')$, simulate nine independent $N(0,1)$ variables and multiply by B.

$$\vec{Z} = B \cdot \begin{pmatrix} +0.0689 \\ -0.3096 \\ -0.2790 \\ +0.4435 \\ -1.2855 \\ -0.8553 \\ -0.6456 \\ -0.0275 \\ -0.6733 \end{pmatrix} = \begin{pmatrix} +0.0689 \\ -0.2292 \\ -0.2929 \\ +0.1630 \\ -1.1803 \\ -1.0575 \\ -1.0120 \\ -0.5839 \\ -1.1143 \end{pmatrix}$$

Finally, obtain the copula $\vec{U}$ in the following way:

$$\vec{U} = \begin{pmatrix} \Phi(+0.0689) \\ \Phi(-0.2292) \\ \Phi(-0.2929) \\ \Phi(+0.1630) \\ \Phi(-1.1803) \\ \Phi(-1.0575) \\ \Phi(-1.0120) \\ \Phi(-0.5839) \\ \Phi(-1.1143) \end{pmatrix} = \begin{pmatrix} 0.5275 \\ 0.4093 \\ 0.3848 \\ 0.5647 \\ 0.1189 \\ 0.1452 \\ 0.1558 \\ 0.2797 \\ 0.1326 \end{pmatrix}$$

where $\Phi(x)$ is the $N(0,1)$ cumulative distribution function.

Default times simulation. To simulate the default time, use the inverse of the survival functions, as explained in Section 3.2.2. The first obligor has an initial rating AA and a copula value of 52.75. In Table 5 search for the month in which AA is close to 52.75, and consider this month, 961, as the month in which the first obligor defaults.

Obligor	Rating	$\vec{U}_i$	Default month
O1	AA	0.5275	961
O2	BBB	0.4093	898
O3	AA	0.3848	> 1000
O4	BB	0.5647	310
O5	B	0.1189	> 1000
O6	BBB	0.1452	> 1000
O7	AAA	0.1558	> 1000
O8	CCC	0.2797	171
O9	AA	0.1326	> 1000

Table 6: Simulated default times

Portfolio loss evaluation. To evaluate the portfolio loss, we use the simulated default times and the portfolio composition (Tables 1 and 2), as explained in Section 3.2.3.

Obligor	Default month	Asset	Exposure	Recovery	Loss
O1	961	A1		80%	
O2	898	A2		75%	
O2	898	A3		75%	
O3	> 1000	A4		60%	
O4	310	A5	70\$ + 90\$	90%	16\$
O4	310	A6	15\$	90%	1.5\$
O4	310	A7		90%	
O5	> 1000	A8		30%	
O6	> 1000	A9		60%	
O7	> 1000	A10		50%	
O8	171	A11	40\$ + 90\$	60%	52\$
O9	> 1000	A12		50%	

Table 7: Portfolio loss evaluation

The simulated portfolio loss is the sum of all asset losses, or \$ 69.5 in this Monte Carlo trial.

A Appendices

A.1 From transition matrix to survival functions

The T-years transition matrix gives the probability of changing from rating r_i to rating r_j in a period of T years:

$$M_T = \begin{pmatrix} m_{1,1} & \dots & m_{1,n} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \dots & m_{n,n} \end{pmatrix} \qquad m_{i,j} = P(r_i \to r_j; T)$$

where n is the number of ratings and $m_{i,j}$ is the probability that a obligor with rating r_i changes to rating r_j in T years. Figure 9 shows a transition matrix in which the probability that a obligor with rating AA changes to rating B in one year is 0.14%.

	AAA	AA	A	BBB	BB	B	CCC	Default
AAA	90.81	8.33	0.68	0.06	0.12	0.00	0.00	0.00
AA	0.70	90.65	7.79	0.64	0.06	0.14	0.02	0.00
A	0.09	2.27	91.05	5.52	0.74	0.26	0.01	0.06
BBB	0.02	0.33	5.95	86.93	5.30	1.17	0.12	0.18
BB	0.03	0.14	0.67	7.73	80.53	8.84	1.00	1.06
B	0.00	0.11	0.24	0.43	6.48	83.46	4.07	5.21
CCC	0.22	0.00	0.22	1.30	2.38	11.24	64.86	19.78
Default	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

Figure 9: One-year transition matrix

The transition matrix can be scaled in time by using the following rules:

$$\begin{aligned} M_{T_1+T_2} &= M_{T_1} \cdot M_{T_2} \\ M_{k \cdot T} &= M_T^k \\ M_{\frac{T}{k}} &= \sqrt[k]{M_T} \end{aligned} \tag{1}$$

The root of a matrix can be computed as:

$$M = P \cdot D \cdot P^{-1} \longrightarrow M^\gamma = P \cdot D^\gamma \cdot P^{-1}$$

where P is a matrix composed of the eigenvectors of M, and D is a diagonal matrix composed of the eigenvalues of M. The inverse of matrix P can be computed using the LU decomposition, as explained in *Numerical Recipes in C*[2]:

[2]http://www.nr.com

$$P \cdot P^{-1} = L \cdot U \cdot P^{-1} = Id$$

Sometimes, the scaled transition matrix does not satisfy the Markov conditions (the row sum is equal to one, and all elements are non-negatives). In this case, we need to transform this matrix to the relevant Markov matrix. This process is called regularization and is explained in Appendix A.2.

Thus we can compute the default probability at any time and at any initial rating (in other words, the survival functions) by calculating:

$$Survival(r_i, t) = 1 - (M_t)_{i,n}$$

where r_i is the initial rating, t is the time, M_t is the transition matrix for time t (scaled from M_T using (1)), and n is the index of the default rating, r_n.

A.2 Transition matrix regularization

The algorithm described here is extracted from [1]. The QOM (Quasi-Optimization of the root Matrix) algorithm regularizes a transition matrix, m_{ij}, of dimension n row by row. The steps to regularize the i-th row are:

Step 1. Compute the difference between the row sum and one. Divide by n and subtract this value from all non-zero components:

$$m_{ij} \neq 0 \Longrightarrow m_{ij} = m_{ij} - \frac{1}{n}\left(\sum_{j=1}^{n} m_{ij} - 1\right)$$

Step 2. If all the row elements are non-negative and sum to one, then stop, as the row is regularized.

Step 3. Fix any negative row element to zero and go to Step 1.

Apply the previous algorithm for every row. The algorithm stops after m steps, where $m \leq n$ (Merkoulovitch 2000). The final matrix is a regularized matrix.

A.3 Correlation matrix estimation

In this section, we demonstrate a simple method for estimating the default times correlation matrix. Suppose that we have the number of components and the number of defaulted components for each sector for the last few years (e.g., 1980-2008):

Year	Sector_1	$\dots$	Sector_m
1	$a_{1,1}$	$\dots$	$a_{1,m}$
2	$a_{2,1}$	$\dots$	$a_{2,m}$
3	$a_{3,1}$	$\dots$	$a_{3,m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$T-1$	$a_{T-1,1}$	$\dots$	$a_{T-1,m}$
T	$a_{T,1}$	$\dots$	$a_{T,m}$

Year	Sector_1	$\dots$	Sector_m
1	$d_{1,1}$	$\dots$	$d_{1,m}$
2	$d_{2,1}$	$\dots$	$d_{2,m}$
3	$d_{3,1}$	$\dots$	$d_{3,m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$T-1$	$d_{T-1,1}$	$\dots$	$d_{T-1,m}$
T	$d_{T,1}$	$\dots$	$d_{T,m}$

where $a_{i,j}$ is the number of obligors in sector j at year i, and $d_{i,j}$ is the number of defaulted obligors in sector j at year i.

We use the following expression of correlation:

$$Correl(S_i, S_j) = \frac{P(S_i \cap S_j) - P(S_i) \cdot P(S_j)}{\sqrt{P(S_i) \cdot (1 - P(S_i))} \cdot \sqrt{P(S_j) \cdot (1 - P(S_j))}}$$

where

$$P(S_i) = \frac{1}{T} \sum_{t=1}^{T} \frac{d_{t,i}}{a_{t,i}}$$

$$P(S_i \cap S_j) = \begin{cases} \frac{1}{T} \sum_{t=1}^{T} \frac{d_{t,i} \cdot d_{t,j}}{a_{t,i} \cdot a_{t,j}} & \text{if } i \neq j \\ \frac{1}{T} \sum_{t=1}^{T} \frac{d_{t,i} \cdot (d_{t,i}-1)}{a_{t,i} \cdot (a_{t,i}-1)} & \text{if } i = j \end{cases}$$

Observe that $P(S_i)$ is the probability that an individual belonging to sector i defaults over a year. To determine $P(S_i \cap S_j)$, divide all combinations of pairs of defaulted individuals by all combinations of pairs of individuals. You can find more information about correlation matrix estimation in [11] and [10].

A.4 Gaussian copula simulation

We describe the algorithm to simulate a Gaussian copula satisfying the correlation matrix Σ [15] [5]:

Step 1. We create the covariance matrix Σ' mapping each Σ component, ρ_{ij}, to $2sin(\frac{\pi}{6}\rho_{ij})$:

$$\Sigma' = \begin{pmatrix} 1 & 2sin(\frac{\pi}{6}\rho_{12}) & \dots & 2sin(\frac{\pi}{6}\rho_{1n}) \\ 2sin(\frac{\pi}{6}\rho_{12}) & 1 & \dots & 2sin(\frac{\pi}{6}\rho_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ 2sin(\frac{\pi}{6}\rho_{1n}) & 2sin(\frac{\pi}{6}\rho_{2n}) & \dots & 1 \end{pmatrix}$$

Step 2. We apply Cholesky[3] to Σ', then $\Sigma' = B \cdot B^\top$, where:

$$B = \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

Step 3. We simulate a $N(0,1)$ n times:

$$\vec{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad y_k \sim N(0,1) \text{ independents}$$

Step 4. We simulate a multivariate normal, $\vec{Z} \sim N(\vec{0}, \Sigma')$:

$$B \cdot \vec{Y} = \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} = \vec{Z}$$

Step 5. Finally, we obtain the copula $\vec{U}$:

$$\begin{pmatrix} \Phi(z_1) \\ \vdots \\ \Phi(z_n) \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \vec{U}$$

where $\Phi(x)$ is the $N(0,1)$ cumulative distribution function.

[3] Appendix A.6 adapts Cholesky decomposition to deal with the block matrix.

A.5 T-Student copula simulation

We describe the algorithm to simulate a t-Student copula with ν degrees of freedom to satisfy the correlation matrix Σ [5]:

Step 1. We create the covariance matrix Σ', mapping each Σ component, ρ, to $f(\rho)$:

$$\Sigma' = \begin{pmatrix} 1 & f(\rho_{12}) & \dots & f(\rho_{1n}) \\ f(\rho_{12}) & 1 & \dots & f(\rho_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ f(\rho_{1n}) & f(\rho_{2n}) & \dots & 1 \end{pmatrix}$$

where

$$h(\nu) = \frac{\pi}{6} + \frac{1}{0.44593 + 1.3089 \cdot \nu}$$
$$f(\rho) = \frac{sin(h(\nu) \cdot \rho)}{sin(h(\nu))}$$

This mapping is done because the correlation used to simulate the copula is different from the copula correlation. The following lines explain how functions f and h have been estimated.

We define the function $g(\rho, \nu)$ as the numerically estimated correlation of a bivariate t-Student distribution with ν degrees of freedom, simulated with the procedure described in this section with $f(\rho) = \rho$. Observe that $g^{-1}(x) = f(x)$. To obtain an approximation of $f(x)$, we compute $g(x)$ at regular intervals. We fit the computed values $(g(\rho_i, \nu), \rho_i)$ to $f(x) = sin(a \cdot x)/sin(a)$ using least squares to determine the parameter a. We repeat the previous procedure for various values of the variable ν. We fit the computed values (ν_j, a_j) to $h(y) = \frac{\pi}{6} + \frac{1}{b + c \cdot y}$ using least squares to determine the parameters b and c.

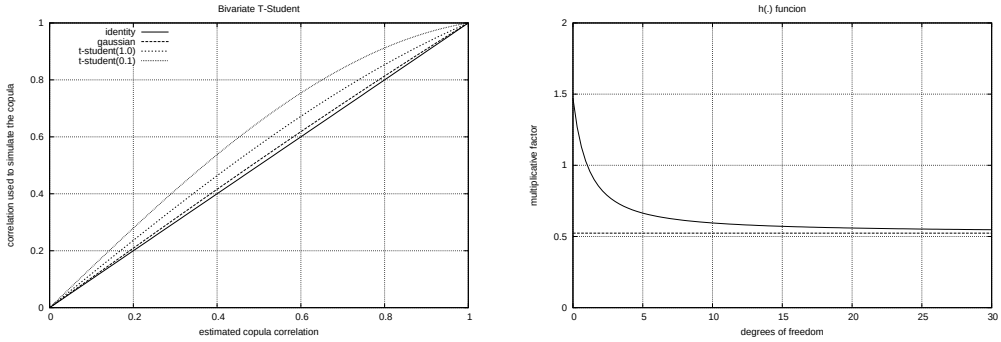

Figure 10: T-Student copula correlations

Step 2. We apply Cholesky[4] to Σ', then $\Sigma' = B \cdot B^\top$, where:

$$B = \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

Step 3. We simulate a $N(0,1)$ n times:

$$\vec{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad y_k \sim N(0,1) \text{ independents}$$

Step 4. We simulate a chi-square with ν degrees of freedom:

$$s \sim \chi^2(\nu)$$

Step 5. We simulate a multivariate t-Student distribution with ν degrees of freedom, $\vec{Z} \sim \frac{\sqrt{\nu}}{\sqrt{\chi^2(\nu)}} \cdot N(\vec{0}, \Sigma')$:

$$\frac{\sqrt{\nu}}{\sqrt{s}} \cdot B \cdot \vec{Y} = \frac{\sqrt{\nu}}{\sqrt{s}} \cdot \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} = \vec{Z}$$

[4]Appendix A.6 adapts Cholesky decomposition to deal with the block matrix.

Step 6. Finally we obtain the copula $\vec{U}$:

$$\begin{pmatrix} t_\nu(z_1) \\ \vdots \\ t_\nu(z_n) \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \vec{U}$$

where $t_\nu(x)$ is the t-Student with ν degrees of freedom cumulative distribution function.

A.6 Cholesky decomposition for a symmetric block matrix

The Cholesky algorithm decomposes a symmetric matrix that is positive-definite into a lower triangular matrix and the transpose of the lower triangular matrix. The algorithm description can be found in *Numerical Recipes in C*[5]:

$$A = U^\top \cdot U$$

If we have a portfolio of 50000 obligors, then the correlation matrix size will be 50000×50000. This requires up to 19 GB of RAM. The multiplication of this matrix by a vector implies 2500000000 multiplications. This is far too many. So, we adapt the Cholesky algorithm in order to consider that the obligors correlation matrix is a block matrix with 1's in the diagonal. For instance:

$$A = \left(\begin{array}{cccc|ccc} 1 & 0.5 & 0.5 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.5 & 1 & 0.5 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.5 & 1 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.5 & 0.5 & 0.5 & 1 & 0.1 & 0.1 & 0.1 \\ \hline 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.3 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.3 & 1 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.3 & 0.3 & 1 \end{array}\right)$$

We decompose the previous matrix using the standard Cholesky decomposition:

$$U = \left(\begin{array}{cccc|ccc} 1.00000 & 0.50000 & 0.50000 & 0.50000 & 0.10000 & 0.10000 & 0.10000 \\ 0 & 0.86603 & 0.28868 & 0.28868 & 0.05774 & 0.05774 & 0.05774 \\ 0 & 0 & 0.81650 & 0.20412 & 0.04082 & 0.04082 & 0.04082 \\ 0 & 0 & 0 & 0.79057 & 0.03162 & 0.03162 & 0.03162 \\ \hline 0 & 0 & 0 & 0 & 0.99197 & 0.28630 & 0.28630 \\ 0 & 0 & 0 & 0 & 0 & 0.94975 & 0.21272 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.92563 \end{array}\right)$$

[5] http://www.nr.com

We can see that U has repeated elements that can be kept in RAM memory in this way:

$$U = \left|\begin{array}{c|cc} 1.00000 & 0.50000 & 0.10000 \\ 0.86603 & 0.28868 & 0.05774 \\ 0.81650 & 0.20412 & 0.04082 \\ 0.79057 & 0 & 0.03162 \\ 0.99197 & 0 & 0.28630 \\ 0.94975 & 0 & 0.21272 \\ 0.92563 & 0 & 0 \end{array}\right|$$

That is, for each row, we keep the diagonal value and the value of each sector. With this strategy, the required memory size is $N \times (M + 1)$, where N is the number of obligors, and M is the number of sectors. With this consideration, the memory required to store a 50000×50000 obligors correlation matrix is only 4.2 Mb.

We use the fact that matrix U has repeated elements to reduce the number of operations that are required to multiply U by a vector. For example:

$$\begin{aligned} (U \cdot x)_2 &= 0.0 \cdot x_1 + 0.86603 \cdot x_2 + 0.28868 \cdot x_3 + 0.28868 \cdot x_4 + \\ &\quad 0.05774 \cdot x_5 + 0.05774 \cdot x_6 + 0.05774 \cdot x_7 \\ &= 0.86603 \cdot x_2 + 0.28868 \cdot (x_3 + x_4) + 0.05774 \cdot (x_5 + x_6 + x_7) \end{aligned}$$

Considering this, we can reduce the number of operations from N^2 to $N \times (M + 1)$, where N is the number of obligors, and M is the number of sectors. In the 50000 obligors example, the operations number reduces from 2500000000 to only 500000.

A.7 Condition number for a symmetric block matrix

The condition number of a matrix indicates how far or close it is to a singular matrix. If the condition number is close to 1, then the matrix is said to be well-conditioned; a matrix with a high condition number is said to be ill-conditioned.

Given a symmetric block matrix with 1's in the diagonal, Σ, we desire to obtain the condition number of the Cholesky decomposition matrix, B. Using the condition number definition and its properties, we know that:

$$\kappa(B) = \|B\|_2 \cdot \|B^{-1}\|_2 = \frac{\sigma_{max}(B)}{\sigma_{min}(B)} = \sqrt{\frac{\lambda_{max}(\Sigma)}{\lambda_{min}(\Sigma)}}$$

where $\sigma_{max}(B)$ and $\sigma_{min}(B)$ are the maximal and minimal singular values of B, respectively, and $\lambda_{max}(\Sigma)$ and $\lambda_{min}(\Sigma)$ are the maximal and minimal absolute values of the eigenvalues of Σ, respectively. The eigenvalues of Σ can be computed using the following unproved proposition.

Proposition. Let Σ be a symmetric block matrix with 1's in the diagonal, where k is the number of blocks, a_{ij} is the block value and n_i is the dimension of the i-th block. Then, the eigenvalues of Σ are:

$$\underbrace{1-a_{11},\cdots,1-a_{11}}_{n_1-1},\underbrace{1-a_{22},\cdots,1-a_{22}}_{n_2-1},\cdots\underbrace{1-a_{kk},\cdots,1-a_{kk}}_{n_k-1},\lambda_1,\cdots,\lambda_k$$

where λ_i are the eigenvalues of the following deflated matrix:

$$K=\begin{pmatrix} 1+(n_1-1)\cdot a_{11} & n_2\cdot a_{12} & \cdots & n_k\cdot a_{1k} \\ n_1\cdot a_{12} & 1+(n_2-1)\cdot a_{22} & \cdots & n_k\cdot a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ n_1\cdot a_{1k} & n_2\cdot a_{2k} & \cdots & 1+(n_k-1)\cdot a_{kk} \end{pmatrix}$$

References

[1] Marina Sidelnikova Alexander Kreinin. Regularization algorithms for transition matrices. *Algo Research Quarterly, Vol. 4, Nos. 1/2*, 2001.

[2] Dirk Tasche Carlo Acerbi. Expected shortfall: a natural coherent alternative to value at risk. *BIS*, 2001.

[3] Darrel Duffie and Kenneth J. Singleton. *Credit Risk. Pricing, Measurement, and Management*. Princeton University Press, 2003.

[4] Marno Verbeek Erik Kole, Kees Koedijk. Selecting copulas for risk management. 2006.

[5] Elisabeth Joossens Francesco Saita, Francesca Campolongo and Maria Egle Romano. Pricing multiasset equity options with copulas: an empirical test.

[6] Greg M. Gupton, Christopher C. Finger, and Mickey Bhatia. *CreditMetrics - Technical Document*. J.P. Morgan & Co. Incorporated, 1997.

[7] Jonathan E. Grindlay Jaesub Hong, Eric M. Schlegel. New spectral classification technique for x-ray sources: quantile analysis. 2004.

[8] Stefan R. Jaschke. Quantile-var is the wrong measure to quantify market risk for regulatory purposes. *BIS*, 2001.

[9] Philippe Jorion. *Value at Risk*. McGraw-Hill, 1997.

[10] David X. Li. *On Default Correlation: A Copula Function Approach*. The RiskMetrics Group, 2000.

[11] Douglas Lucas. *Default Correlation: From Definition to Proposed Solutions*. CDO Research, 2004.

[12] Alexander McNeil Paul Embrechts and Daniel Straumann. Correlation and dependence in risk management: properties and pitfalls. *RiskLab*, 1999.

[13] Murray R. Spiegel. *Estadística*. Schaum, 1997.

[14] Gerard Torrent. Simulating large portfolios of credit: The creditcruncher project. *Ercim News*, (78):35–36, 2009.

[15] Shaun S. Wang. Aggregation of correlated risk portfolios: Models & algorithms. *CAS Committee on Theory of Risk*.