POLITECNICO DI TORINO

Master Course
in Mathematical Engineering

Master Thesis

High-dimensional simulations of portfolio credit risk

Supervisor
Prof. Patrizia Semeraro

Candidate
Michele Carone

Academic Year 2022-2023

Alla mia famiglia

† Ai nonni Michele, Vito e Antonia

Summary

Credit risk is one of the most important types of risk that any financial institution must face with. In the financial world, the essence of control relies on how well the model mimes reality and on the precision of the computational methods used. In this thesis, we consider the most common credit risk models: "CreditRisk+" of Credit Suisse Financial Product, and the Gaussian and t-copula models. All of them belong to the Bernoulli mixture models, used in literature for tractability reasons, and for their generical modeling approach. We, therefore, focus on Bernoulli mixture models, without loss of generality. We consider the case of homogeneous portfolios, i.e. exchangeable Bernoulli mixture model. We aim to quantify the credit risk related to the obligors' default. To satisfy this target, we calculate the proper risk metrics as tail loss probability and conditional loss for the Bernoulli mixture models. The problem relies on how to conduct and set the simulation to calculate these risk indicators: the naïve Monte Carlo simulation in rare event settings is problematic as the number of repetitions should increase enormously to get reliable estimates of metrics. This work proposes to overcome the common pitfall of the naïve approach, following the new efficient simulation proposed by Baṣoğlu, Hörmann, and Sak. This innovative algorithm combines the importance sampling based on cross-entropy, and inner replication using the geometric shortcut, and we apply it to all three models described. An overview analysis is based on the computation of Value-at-Risk (VaR) showing the improvements in the benchmark methods for the credit risk models.

Acknowledgements

Un sentito ringraziamento va alla mia Relatrice, Prof. Patrizia Semeraro, per avermi affiancato durante il percorso di redazione della tesi e per la grande stima e attenzione dimostrata nei miei confronti.

Dedico questo lavoro di tesi ai miei genitori simbolo di saggezza e perseveranza e al piccolo Gabriele simbolo di amore e speranza. Ringrazio la mia famiglia, le mie sorelle Serena e Antonella, inisieme a Tony per avermi insegnato la cura e l'amore verso me stesso. Anche se lontani ho sentito il vostro affetto e la consolazione nei momenti più diffcili.
Ringrazio tutte le persone che si sono affacciate nel mio percorso universitario, per aver contributo ad arricchire e a formare la persona che oggi sono. Partendo dai miei primi colleghi, Matteo e Giampiero: i primi gregari sempre pronti a pedalare fino in cima alle più dure salite. Ringrazio Dario e Francesco, esempi di diligenza e constanza nel persguire gli obiettivi.
Grazie gli amici del collegio, per aver sopportato il mio particolare carattere spesso scontroso e irascibile, a voi tutti il più grande grazie per aver ricreato la gioia familiare. In particolare: Flavio, Sara, Leonardo Ossìno, Leornardo G., Stefano, Mattia, Laura, Rosalinda, Adele e Davide Rossetti.
Ringrazio i miei più cari amici, per le gioie e le emozioni regalate in tutti questi anni. Davide, per il suo carattere partenopeo e l'affetto fraterno. Grazie a Lorenzo, già esperto di simulazioni in alte dimensioni, per essere stato una guida universitaria e un esempio di vita: questi ringraziamenti non potranno mai spiegare le variazioni su tema composte su un vasto repertorio musicale. A Peppe e Agostino, per tutti quelli che sanno e quelli che non sanno. Ai miei gemelli Roberto e Alessandro, per avermi fatto sentire in Puglia

anche a kilometri di distanza.
Ringrazio Francesca, per l'immancabile sostegno in qualsiasi momento. Grazie per avermi fatto da specchio dell'anima e da bussola in un oceano di incertezze.
Ringrazio Simona, per avermi insegnato l'esistenza di emozioni forti, spesso abbandonate nei meandri del cuore. Grazie per avermi accompagnato giorno dopo giorno fino a questo traguardo.

"Qualunque fiore tu sia,
quando verrà il tuo tempo, sboccerai.
Prima di allora
una lunga e fredda notte potrà passare.
Anche dai sogni della notte trarrai forza e nutrimento.
Perciò sii paziente verso quanto ti accade
e curati e amati
senza paragonarti
o voler essere un altro fiore,
perché non esiste fiore migliore di quello
che si apre nella pienezza di ciò che è.
E quando ciò accadrà,
potrai scoprire
che andavi sognando
di essere un fiore
che aveva da fiorire."
(Daisaku Ikeda)

Contents

List of Tables 9

List of Figures 10

1 Introduction to Credit Risk simulation 13

2 Bernoulli mixture models 17

2.1 Exchangeable Bernoulli Mixture model 18

2.2 Poisson approximation 19

2.3 Notation adopted 20

2.4 CreditRisk+ 21

2.4.1 Exchangeable version of Creditrisk+ 22

2.5 Gaussian copula model 24

2.6 t-copula model 25

2.7 Structure of models 26

3 Efficient simulation 29

3.1 Naive Monte Carlo simulation 30

3.2 CEGS simulation 32

3.2.1 Geometric shortcut 32

3.2.2 Cross-entropy method 35

3.2.3 How to implement CEGS simulation 42

4 Models settings and simulations implementation 47
4.1 Parameters setting . . . 48
4.1.1 CreditRisk+ parameters . . . 48
4.1.2 Gaussian copula model parameters . . . 49
4.1.3 t-copula model parameters . . . 50
4.2 Naive Monte Carlo (NV) implementation . . . 52
4.3 CEGS implementation . . . 53
4.3.1 CreditRisk+ . . . 54
4.3.2 Gaussian copula model . . . 57
4.3.3 t-copula model . . . 60
4.4 Experimental result . . . 66

5 Conclusions and future works 73

List of Tables

4.1 Experimental results for tail loss probabilities 69
4.2 Experimental results for conditional excesses 69
4.3 Computational times 70

List of Figures

2.1 Structure of Bernoulli mixture model analyzed 27
3.1 Geometric shortcut across obligors 34
3.2 Flowchart of CEGS method 42
3.3 Details of third block of flowchart of CEGS method 43
3.4 Detailed link of 10th an 11th block of flowchart of CEGS method 44
4.1 Distribution used to generate the factor loadings of t-copula model 51
4.2 Objective function of cross-entropy optimization problem for CreditRisk+ model 57
4.3 Objective function of cross-entropy optimization problem for Gaussian copula model 60
4.4 Detail on convexity of objective function fro Gaussian copula model 61
4.5 Objective function of cross-entropy optimization problem for t-copula model 65
4.6 t-copula model: objective function for theta optimization subproblem . . . 65
4.7 t-copula model: objective function for v_D optimization subproblem 66
4.8 Distributions of losses and VaRs at 95% for the Naive Monte Carlo simulations 71
4.9 Distributions of losses and VaRs at 95% for the CEGS Monte Carlo simulations 72

List of Algorithms

1 Naive Monte Carlo simulation 31

2 Cross Entropy method, for a grouping of first D' elements of $\mathbf{\Psi}$ 40

3 Tail loss probability and conditional excess with CEGS simulation 45

Miliardi di mondi
esistono ancora
Miliardi di vite,
per provare ancora

[Niccolò Contessa - i cani, Calabi-Yau min. 2:44]

Chapter 1

Introduction to Credit Risk simulation

Credit is an absolute must in the financial system, affects everyone and drives the global economy. Credit allows individuals to finance their needs of acquiring a house, car, furniture etc., assists companies to start or expand their business, and enables governments to finance public interest projects. If managed well, it can build an economy, produce an efficient allocation of capital and wealth and bring prosperity. The allocation of credit is performed by financial intermediaries such as commercial and investment banks, saving and loan associations, insurance companies, mutual funds, pension funds and finance companies. They are crucial to the healthy functioning of financial markets because of their role in deciding who gets credit and at which price. Over the past three decades intermediaries begin to offer increasingly sophisticated products and innovative financial contracts. However, if their risk is not fully understood, they can lead to devastating repercussion on the financial system. After the crisis of a German bank (Herstatt Bank), in 1974 the central bank governors of the Group of Ten countries established the Basel Committee on Banking Supervision (BCBS). Since the issuance of the first Concordat of 1975, the Committee, constituted by representatives of central banks and banking supervisory bodies, dictated international standards which aim is to ensure banking regulation.

The agreements have occurred over time: Basel I in 1988 and Basel II in 2004. Then, after the financial crisis of 2007, the importance of credit and credit management has increased and even if financial intermediaries have always been regulated, their regulation has had to change dramatically. In fact, based on the traumatic experience of the crisis, in 2010 the Committee published the first text of Basel III. This accord introduced stronger risk management requirements for banks. To better understand what credit risk is we can give a definition. Credit risk is the risk of financial loss due to the borrower's bond issuer's or counterparties (the obligors) failure to honour their financial obligations. This can be due to the inability or unwillingness of the counterparty, but this last case is less common with respect to the first one. This inability is linked with the concept of default. Default can be defined as a missed or delayed payment of a contractual obligation or legal receivership of the obligor that will probably cause one or more missed or delayed future payments. The sources of credit risk can be various (deposits, prepayment of goods or services, contingent claims, bonds, derivatives etc.), but the focus of our analysis will be on loans. A loan is a cash outflow provided from the lender to the borrower with the promise to repay it back at a later scheduled date. Of course, the loan has a cost which is defined as the interest rate paid by the borrower to the lender on the loan principal amount at scheduled interest payment dates. The full term and conditions of the loan are defined in the loan agreement. Going into more detail, loans can be of different types: secured or unsecured. As the name says, a secure loan imply a lower credit risk for the lender than an unsecured one. This is due to the fact that with a secured loan the borrower pledges asset as collateral that can be, in case of necessity, repossessed and sold by the lender to recover the sums owed. Classical examples of secured personal loans are mortgages or car loan, which are respectively secure on the house or on the car. On the other hand unsecured loans do not involve any type of collateral. Common examples include credit cards, personal loans etc. .

For our analysis, we focus on models for credit portfolios discussing the typical credit risk management issues that arise when we built a portfolio of non-traded credit products, such as the commercial loans of a bank. The key in modeling is to catch the real dependence structure of the default events. The impact on the portfolio credit loss distribution,

due to the presence of the event's dependence, is reflected principally in the shape of the right tail of the loss distribution. It is common wisdom, from the financial point of view, to expect dependence between the default of different obligors. The randomly fluctuating macroeconomic factors are responsible for the firm's wealth, and since different firms are affected by common factors, the correlation between their default is unavoidable. Do these reasons open the discussion around which model captures reality with the best precision of the computational method?

The development of the market for credit derivatives and the Basel III process has generated a lot of interest in quantitative credit risk models, so credit risk modeling is a very active subfield of quantitative finance and risk management. In this section we provide a brief overview of the various model types that are used in credit risk, focusing on static models. In fact, credit risk management models are used to determine the loss distribution of a loan (or a bond) portfolio over a fixed time period (typically at least one year) and to compute loss-distribution-based risk measures. Hence these models are typically static, in the sense that they focus on the loss distribution for the fixed time period rather than a stochastic process describing the evolution of risk in time. Credit risk models can be divided into *structural* or *firm-value models* on the one hand and *reduced-form models* on the other. In *firm-value models*, default occurs whenever a stochastic variable, representing an asset value, falls below a threshold representing liabilities. For these reasons, static structural models are often called threshold models. In reduced-form models, the mechanism leading to default is left unspecified. The default time is modeled as a non-negative random variable, whose distribution typically depends on economic covariables. The most commonly used are the *mixture models*. In this model, defaults are assumed to be conditionally independent given a set of common factors. The factors, as underlined previously, are interpreted as macroeconomic variables and are also modelled stochastically. The spread of mixture models is motivated by tractability reasons.

We are interested in calculating the tail loss probability and conditional excess for the Bernoulli mixture model of portfolio credit risk. The principal target of this work is to develop a simulation methodology to quantify credit risk, as risks related to the obligors' default. The previous quantities constitute important metrics that allow having a realistic

measure of risk. Grouping all the previous elements mentioned before, we consider a Bernoulli mixture model for a loan portfolio. A possible method for calculating these risk measures is to use Monte Carlo simulation, but the problem of *rare-event simulation* arises. Unless the number of simulations is very large, *rare-event* settings lead to an estimate of metrics affected by high variability. The problem is that most repetitions are not significant and less informative in the computation, resulting in the degradation of the simulation. In this thesis, we implement the new method to estimate tail loss probability and conditional excess, proposed by Basoglu et al. [2018], applied to Bernoulli mixture models of credit risk. It is a combination of importance sampling and inner replication for the simulation of the random variables that introduce the dependence across obligors. The technique adopted belongs to the methods of variance reduction, useful to improve the quality of estimates in Monte Carlo simulation framework. We implement the importance sampling strategy based on the cross-entropy method, which increases the probability of rare defaults. The remaining source of the variance is the simulation of obligors' default, for which Basoglu et al. [2018] employ inner replications using the geometric shortcut method. For these purposes we use the MATLAB environment to implement all the algorithms proposed, taking the advantage of the *Financial toolbox* and the *Statistic and Machine learning toolbox*. This thesis is organized as follows: Chapter 2 gives an overview focused on Bernoulli mixture models and analyzes the three models chosen for the simulation: CreditRisk+, Gaussian copula model, and t-copula model. In Chapter 3 we explain Monte Carlo simulation and the variance reduction methods adopted. Chapter 4 discusses the implementation details of the entire simulation, starting from the setting of the parameter to the calculation of risk metrics. In the last Section 4.4 of the same chapter are presented and compared the main results. In Chapter 5 we report the conclusion and future works.

Chapter 2

Bernoulli mixture models

In a mixture model the default risk of an obligor is assumed to depend on a set of common economic factors, such as macroeconomic variables, which are also modelled stochastically. As anticipated before, given a realization of the factors, defaults of individuals are assumed to be independent. Dependence between defaults stems from the dependence of individual default probabilities on the set of common factors.

Definition 1 *Given some $D < J$ and a D-dimensional random vector $\mathbf{\Psi} = (\Psi_1, ..., \Psi_D)'$, the random vector $\mathbf{Y} = (Y_1, ..., Y_J)'$ follows a Bernoulli mixture model with factor vector $\mathbf{\Psi}$ if there are functions $p_j : \mathbb{R}^D \to [0,1], 1 \leq j \leq J$, such that, conditional on $\mathbf{\Psi}$, the components of $\mathbf{Y}$ are independent Bernoulli random variables satisfying $P(Y_j = 1|\mathbf{\Psi} = \boldsymbol{\psi}) = p_j(\boldsymbol{\psi})$.*

For $\mathbf{y} = (y_1, ..., y_J)'$ in $\{0,1\}^J$ we have that:

$$P(\mathbf{Y} = \mathbf{y} \mid \mathbf{\Psi} = \boldsymbol{\psi}) = \prod_{j=1}^{J} p_j(\boldsymbol{\psi})^{y_j} \ (1 - p_j(\boldsymbol{\psi}))^{1-y_j} \tag{2.1}$$

and the uncoditional distribution of the default indicator vector $\mathbf{Y}$ is obtained by integrating over the distribution of the factor $\mathbf{\Psi}$. In particular, the default probability of company j is given by:

$$p_j \ = \ P(Y_j = 1) \ = \ E(p_j(\mathbf{\Psi})).$$

The two-stage hierarchical structure of a Bernoulli mixture model facilitates sampling from the model: first we generate the economic factor realizations, then we generate the pattern of defaults conditional on those realizations. The second step is easy because of the conditional independence assumption, as reported in A. J. McNeil [2005].

2.1 Exchangeable Bernoulli Mixture model

An important simplification occurs if the function p_j are all identical. In this case the Bernoulli mixture model is termed *exchangeable*, since the random vector $\mathbf{Y}$ is exchangeable. It is convenient to introduce the rv $\mathcal{Q} := p_1(\psi)$ and to denote the distribution function of this mixing variable by $G(q)$. Conditional on $\mathcal{Q} = q$, the number of the defaults M is the sum of the m independent Bernoulli variables with parameter q and it therefore has a binomial distribution with parameters q and m, i.e. $P(M = K \mid \mathcal{Q} = q) = \binom{m}{k} q^k (1-q)^{m-k}$. The unconditional distribution of M is obtained by integrating over q:

$$P(M = k) = \binom{m}{k} \int_0^1 q^k (1-q)^{m-k} \, dG(q) \tag{2.2}$$

In this case is easy to compute default probabilities and joint default probabilities taking the advantage of exchangeability property. In the simple case we have:

$$\pi = E(Y_1) = E(E(Y_1|Q)) = E(Q) \tag{2.3}$$

and more generally:

$$\pi_k = P(Y_1 = 1, ..., Y_k = 1) = E(E(Y_1 \cdots Yk|\mathcal{Q})) = E(\mathcal{Q}^k) \tag{2.4}$$

so that unconditional default probabilities of the first and higher order are seen to be moments of the mixing distribution. Moreover, for $i \neq j$ $\text{cov}(Y_i, Y_j) = \pi_2 - \pi^2 = \text{Var}(\mathcal{Q}) \geq 0$, which means that in an exchangiable Bernoulli mixture model the default correlation ρ_Y:

$$\rho_Y := \rho(Y_i, Y_j) = \frac{\pi_2 - \pi}{\pi - \pi^2} \qquad i \neq j, \tag{2.5}$$

where π and π_2:

$$E(Y_i) = E(Y_i^2) = P(Y_i = 1) = \pi \qquad \forall i, \tag{2.6}$$

$$E(Y_i Y_j) = P(Y_i = 1, Y_j = 1) = \pi_2 \qquad \forall i \neq j, \tag{2.7}$$

is always non-negative. Any value of ρ_Y in [0,1] can be obtained by an appropriate choice of the mixing distribution G. In particular, if $\rho_Y = \mathrm{Var}(\mathcal{Q}) = 0$, the random variable $\mathcal{Q}$ has a degenerate distribution with all mass concentrated on the point π and the default indicators are independent. The case $\rho_Y = 1$ corresponds to a model where $\pi = \pi_2$ and the distribution of $\mathcal{Q}$ is concentrated on the points 0 and 1.

2.2 Poisson approximation

Since default is typically a rare event, we can approximate Bernoulli indicator rvs[1] for default with Poisson rvs and to approximate Bernoulli mixture models with Poisson mixture models. Assume that, given the factors $\mathbf{\Psi}$, the default indicator variables satisfying $P(Y_j = 1|\mathbf{\Psi} = \boldsymbol{\psi}) = p_j(\boldsymbol{\psi})$. Moreover, assume that the distribution of $\mathbf{\Psi}$ is such that the conditional default probabilities $p_j(\boldsymbol{\psi})$ tend to be very small. In this case the Y_j variables can be approximated by conditionally independent Poisson variable $\tilde{Y}_j$ satisfying $\tilde{Y}_j \,|\, \mathbf{\Psi} = \boldsymbol{\psi} \sim \mathrm{Poi}(p_j(\boldsymbol{\psi}))$, since:

$$P(\tilde{Y}_j = 0|\mathbf{\Psi} = \boldsymbol{\psi}) = \exp\left(-p_j(\boldsymbol{\psi})\right) \approx 1 - p_j(\boldsymbol{\psi}), \tag{2.8}$$

$$P(\tilde{Y}_j = 1|\mathbf{\Psi} = \boldsymbol{\psi}) = p_j(\boldsymbol{\psi}) \exp\left(-p_j(\boldsymbol{\psi})\right) \approx p_j(\boldsymbol{\psi}). \tag{2.9}$$

In the following we report the formal definition of this model known as *Poisson Mixture model*, as a parallelization of the general Bernoulli mixture model 2.1.

Definition 2 *Given some $D < J$ and a D-dimensional random vector $\mathbf{\Psi} = (\Psi_1, ..., \Psi_D)'$, the random vector $\tilde{\mathbf{Y}} = (\tilde{Y}_1, ..., \tilde{Y}_J)'$ follow a Poisson mixture model with factors $\mathbf{\Psi}$ if there are functions $\lambda_j : \mathbb{R}^D \to (0, \infty), 1 \leq j \leq J$, such that, conditional on $\mathbf{\Psi} = \boldsymbol{\psi}$, the random vector $\tilde{\mathbf{Y}}$ is a vector of independent Poisson distributed rvs with rate parameter $\lambda_j(\boldsymbol{\psi})$.*

If $\tilde{\mathbf{Y}}$ follows a Poisson model and if we define the indicators $Y_j = \mathbb{1}_{\{\tilde{Y}_j \geq 1\}}$, then $\mathbf{Y}$ follows a Bernoulli mixture model and the mixing variables are related by $p_j(\cdot) = 1 - \exp\left(-\lambda_j(\cdot)\right)$.

[1] From this point, we abbreviate the term random variable with rv.

2.3 Notation adopted

After the previous overview of the models that will be treated in our analysis, we focus on the main actor of the simulation: the total loss of portfolio. To be consistent and make easier the implementation of the simulation we decide to follow the same notation as Basoglu et al. [2018], showing the main similarities of the previous notation of A. J. McNeil [2005]. Suppose that there are J obligors in our portfolio and Y_j denotes the Bernoulli random variable for the jth obligors, that is equal to:

- 1 if the jth obligor default;
- 0 otherwise.

As in Definition 1, p_j denotes the marginal probability that the jth obligor defaults, and now we introduce c_j, which quantifies the loss resulting from the default of the same obligor. The total loss is given by:

$$L = \sum_{j=1}^{J} c_j Y_j \tag{2.10}$$

But our analysis aims to find the risk measures, discussed in the Introduction 1:

- Tail loss probability:

$$y = P(L > \tau) = E[\mathbb{1}_{\{L>\tau\}}] \tag{2.11}$$

- Conditional excess:

$$r = E[L|L > \tau] \tag{2.12}$$

where τ stands for a fixed threshold value. Following 2.1 we set D-dimensional ($D < J$) random vector:

$$\mathbf{\Psi} = (\Psi_1, ..., \Psi_D)' \tag{2.13}$$

that represent the vector of macroeconomic factors take into account. If we select a suitable functions $p_j(\mathbf{\Psi}) \quad j = 1, ..., J$, as the conditional default probabilities, the random vector $\mathbf{Y} = (Y_1, ..., Y_J)$ follows a Bernoulli mixture model. Each component of the last vector is an independent Bernoulli random variable, conditioned on $\mathbf{\Psi}$. Changing the

notation in 2.1 and considering $(\epsilon_1, ..., \epsilon_j)'$ in $\{0,1\}^J$, as in Basoglu et al. [2018], our Bernoulli mixture model is:

$$P(Y_1 = \epsilon_1, ..., Y_J = \epsilon_J \mid \mathbf{\Psi}) = \prod_{j=1}^{J} p_j(\mathbf{\Psi})^{\epsilon_j} \ (1 - p_j(\mathbf{\Psi}))^{1-\epsilon_j} \tag{2.14}$$

We restrict the analysis to three examples of the general model described before:

- Creditrisk+;
- Gaussian copula model;
- t-copula model.

The three model share the same hierarchical structure, leaving immutable the Bernoulli setting, and differ from the model behind $p_j(\mathbf{\Psi})$.

2.4 CreditRisk+

The CreditRisk+ model is a credit portfolio risk model that was introduced by Credit Suisse in 1997. It is designed to help financial institutions measure and manage their credit risk exposure at both the portfolio and individual transaction levels. It is based on the concept of default probability, which is the likelihood that a borrower will fail to repay its debt obligations. The model uses a statistical approach to estimate the probability of default for each borrower in a portfolio based on its financial and business characteristics. These characteristics can include factors such as the borrower's credit rating, industry, geographic location, and financial performance [CreditSuisse [2023]]. The CreditRisk+ model has the structure of the Poisson mixture model as described in Section 2.2, but we consider it in a Bernoulli fashion, to get in light the difference with other models. The vector $\mathbf{\Psi} = (\Psi_1, ..., \Psi_D)'$ are independent Gamma random variable with:

- $\alpha_d = \sigma_d^{-2}$ as shape parameter for $d = 1, ..., D$
- $\beta_d = \sigma_d^2$ as scale parameter for $d = 1, ..., D$.

In this case for a given $\mathbf{\Psi}$ vector the conditional default probability looks like:

$$p_j(\mathbf{\Psi}) = 1 - \exp\left(-a_{j0} - a_{j1}\Psi_1 - ... - a_{jD}\Psi\right), \qquad j = 1, ..., J \tag{2.15}$$

The coefficients $a_{j0}, ..., a_{jD}$ are all positive and can be considered as weights of the macroeconomic factors. Notice that each coefficient depends on two indexes, one properly for the obligor j and one for the stochastic factor d. Further consideration will be made in Chapter 4 when we discuss how to set the *a priori* parameter of the simulation.

2.4.1 Exchangeable version of Creditrisk+

In this section, we propose the exchangeable version of CreditRisk+, showing its useful property. Reusing the notation of section 2.1, we recast the random vector $\mathbf{Y}$ which follows the Bernoulli mixture model. CreditRisk+ is exchangeable if the function p_j are all identical, thus the conditional default probabilities are independent of the obligors. In our model, the dependence among probabilities and obligors is due to the weight of the systematic factors a_{jd}, for a certain obligor j and factor d. Assuming independence from obligors, we rewrite the equation 2.15 as:

$$p(\mathbf{\Psi}) = 1 - \exp\left(-a_0 - a_1\Psi_1 - ... - a_D\Psi\right), \qquad \forall j = 1, ..., J \tag{2.16}$$

We can introduce the random variable $Q := p(\mathbf{\Psi})$, which combines multiple random variables, thus it is called a mixing variable. It has a distribution function $G(p)$, where $Q = p$. Conditional con $Q = p$ the number of defaults M follows a Bernoulli process with parameter p and a number of trials m.

$$P(M = k \mid \mathcal{Q} = p) = \binom{m}{k} p^k (1-p)^{m-k} \tag{2.17}$$

We focus in particular on another version of this model. CreditRisk+ is an exchangeable model also in the case when we consider:

- the unidimensional vector of factor $\mathbf{\Psi}$;
- the shape and scale parameters of gamma distribution are $\alpha = \beta$.

To this aim, we can write:

$$\Psi \sim Gamma(\alpha, \beta)$$
$$Q := 1 - e^{-k\Psi}, \qquad k > 0$$

For $q \in (0,1)$:

$$P(Q \leq q) = P(1 - e^{-k\Psi} \leq q) = P(\Psi \leq -\frac{\ln(1-q)}{k})$$

So we can relate the two densities:

$$g_Q(q) = g_\Psi\left(-\frac{\ln(1-q)}{k^2(1-q)}\right)$$

Using the form of gamma density function:

$$\begin{aligned} g_Q(q) &= \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{1}{k(1-q)} \left(\frac{-\ln(1-q)}{k}\right)^{\alpha-1} \exp\left(\frac{\beta \ln(1-q)}{k}\right) \\ &= \left(\frac{\beta}{k}\right)^\alpha \frac{1}{\Gamma(\alpha)} (-\ln(1-q))^{\alpha-1} (1-q)^{\frac{\beta}{k}-1} \end{aligned} \tag{2.18}$$

It is reasonable to use the approximation:

$$-\ln(1-q) \approx q \tag{2.19}$$

because in the credit risk model we are in *rare-event* settings and we choose consequentially parameters in such a way that the mass of the distribution Q is concentrated on values of q close to zero. Compare g_Q, taking the approximation 2.19 , with Beta distribution:

$$B(q) = \frac{1}{\beta(a,b)} q^{a-1}(1-q)^{b-1}, \qquad a, b > 0,\ 0 < q < 1,$$

where $\beta(a,b)$ denote the beta function. So we can consider the following approximation from 2.19 and 2.18:

$$g_Q \sim Beta\left(\alpha, \frac{\beta}{k}\right) \qquad \alpha = \beta$$

Reaching this result we can follow the main result for beta mixing distribution proposed by A. J. McNeil [2005], allowing us to conclude that the conditional default probabilities are all identical.

For the sake of generality, as Basoglu et al. [2018] suggest, we conduct the analysis on the general version of CreditRisk+. This is motivated by the fact the others models have a general form. Considering a model with exchangeability property can lead to biased results, strictly related only to a particular class of obligors.

2.5 Gaussian copula model

The second model analyzed is the Gaussian copula model. Before discussing it we present the unidimensional probit-normal mixing distribution. Recall that Q is the conditional default probability. Given:

$$\Psi \sim N(0,1), \qquad \mu \in \mathbb{R} \qquad \sigma > 0 \tag{2.20}$$

the conditional default probability is:

$$Q = \Phi(\mu + \sigma\Psi) \tag{2.21}$$

where Φ is the standard normal distribution function. From this point, we derive the multidimensional version of 2.21. we introduce the multivariate normal vector:

$$\mathbf{Z} = (Z_1, ..., Z_J)' \tag{2.22}$$

$$Z_j \sim N(0,1) \qquad j = 1, ..., J \tag{2.23}$$

Each component of the vector is one of the J latent variables necessary to model dependencies across obligors. Default of jth obligor occurs when:

$$Y_j = \mathbf{1}_{\{Z_j > z_j\}}, \qquad j = 1, ..., J$$

where:

$$z_j = \Phi^{-1}(1 - p_j) \tag{2.24}$$

indicate a proper threshold value. Φ^{-1} denotes the inverse of the cumulative distribution function of the standard normal. This new model introduces the correlation among obligors represented by:

$$Z_j = bj\epsilon_j + a_{j1}\Psi_1 + ... + a_{jD}\Psi_D, \qquad j = 1, ..., J \tag{2.25}$$

where:

- ϵ_j is the idiosyncratic factor, specific for the jth obligor. It is an independent standard normal random variable.

- $\Psi_1, ..., \Psi_D$ are the systematic risk factors affecting all obligors. They are independent standard normal random variables.

It looks like a multi factors model, as French [2023] style, if we think of our factors as macroeconomic variables. Further consideration can be made on loadings factors b_j and $a_{j1}, ..., a_{jD}$. They are constant, non-negative, and given *a priori* such that they satisfy the constraint:

$$b_j^2 + a_{j1}^2 + ... + a_{jD}^2 = 1 \tag{2.26}$$

Collecting all elements of this model, the conditional default probabilities of the Gaussian copula model, given the vector $\boldsymbol{\Psi}$ is:

$$\boldsymbol{\Psi} = (\Psi_1, ..., \Psi_D)' \tag{2.27}$$

$$p_j(\boldsymbol{\Psi}) = P(Y_j = 1|\boldsymbol{\Psi}) = \Phi((\boldsymbol{a_j}\boldsymbol{\Psi} + \Phi^{-1}(p_j))b_j^{-1}), \qquad j = 1, ..., J \tag{2.28}$$

where $\boldsymbol{a_j} = (a_{j1}, ..., a_{jD})$. Notice that, also this model can be exchangeable if the conditional default probabilities are all identical $p(\boldsymbol{\Psi})$ $\forall j = 1, ..., J$, thus independent from obligors j. The same result, seen in the exchangeable version of CreditRisk+ (Sec. 2.4.1), here is still valid.

2.6 t-copula model

The last model is the t-copula model, which presents the same structure as the previous model with the difference that latent variables follow multivariate t-distribution, rather than normal as in 2.25. The new model correlates the obligors by defining:

$$T_j = \left(b_j\epsilon_j + \sum_{d=1}^{D-1} a_{jd}\Psi_d\right)\left(\frac{\Psi_D}{\eta}\right)^{-\frac{1}{2}}, \qquad j = 1, ..., J. \tag{2.29}$$

As in 2.25, the systematic factors $\Psi_1, ..., \Psi_{D-1}$, the idiosyncratic factor ϵ_j and all factor loadings $a_{j1}, ..., a_{j(D-1)}, b_j$, except for a_{jD}, play the same role as in the Gaussian copula model, except now:

- Ψ_D is a chi-square random variable with η degree of freedom that is independent of $\Psi 1, ..., \Psi_{D-1}$ and ϵ_j.

Default occurs if:

$$Y_j = \mathbb{1}_{\{T_j > t_j\}} \tag{2.30}$$

where t_j is a threshold value related to the marginal default probability of the obligor:

$$t_j = F_\eta^{-1}(1 - p_j) \tag{2.31}$$

where F_η^{-1} is the inverse cumulative of t-distribution with η degree of freedom. Given the vector $\mathbf{\Psi}$, The conditional default probabilities is:

$$\mathbf{\Psi} = (\Psi_1, ..., \Psi_D)' \tag{2.32}$$

$$p_j(\mathbf{\Psi}) = P(Y_j = 1|\mathbf{\Psi}) = \Phi\left(\frac{\tilde{\boldsymbol{a_j}}\tilde{\mathbf{\Psi}} - \sqrt{\frac{\Psi_D}{\eta}}F_\eta^{-1}(1 - p_j)}{b_j}\right), \qquad j = 1, ..., J, \tag{2.33}$$

where $\tilde{\boldsymbol{a_j}} = (a_{j1}, ..., a_{j(D-1)})$ and $\tilde{\mathbf{\Psi}} = (\Psi_1, ..., \Psi_{D-1})'$.

2.7 Structure of models

The 2.15, 2.28 and 2.33 represent the conditional default probabilities of the three Bernoulli mixture models analyzed in this work. We notice that each of them constitutes the second step of the hierarchical structure of the model, as shown in Flowchart 2.1, which must be substituted in the general model 2.14. For simulation purposes, we illustrate this structure shared by all models, to taste and anticipate how the simulation will be implemented, in Figure 2.1.

The orientation of the arrows indicates the right path will be followed in future algorithms proposed in Chapter 4.

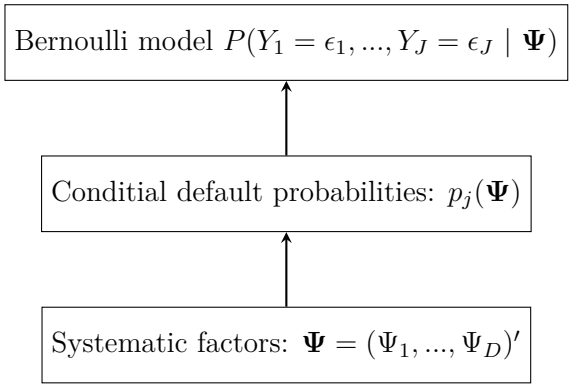


Figure 2.1: Structure of Bernoulli mixture model analyzed

Chapter 3

Efficient simulation

In the previous chapter, we fully described the three models of interest, computing for everyone the conditional default probability $p_j(\mathbf{\Psi}), \quad j = 1, ..., J$, giving the realization of systematic factor. This probability refers to jth obligor, but now we consider the portfolio of all J obligors. For each model, the total loss of portfolio L (as 2.10) is the object of interest, but it's not informative in the performance assessment. Our goal is to obtain an efficient simulation of the loss distribution, so we need to compute sensible metrics, based on loss, to measure the quality of the simulation itself. The two risk metrics are:

- tail loss probability:

$$P(L > \tau) \tag{3.1}$$

- conditional excess:

$$E[L \mid L > \tau] \tag{3.2}$$

each of them is the result of simulations. In plain English we want to test the performance of simulations, basing the assessment on the product of itself. We cannot directly compute the metrics 3.1 and 3.2. Therefore we are looking for the proper estimates of these quantities. For the tail loss probability, we can simply take the empirical estimate from the empirical loss distribution, given a fixed threshold τ:

$$\hat{y} = P(L > \tau) \tag{3.3}$$

where the $\hat{y}$ denotes the estimate. More attention regards the estimate of conditional excess, which we denote $r = E[L|L > \tau]$. As reported in Glasserman and Li [2004], if we assume $\hat{y} > 0$, and we call $\hat{x} = E[L\mathbf{1}_{\{L>\tau\}}]$, a proper estimator is:

$$\hat{r} = \frac{\hat{x}}{\hat{y}}. \tag{3.4}$$

These are the general estimators for the risk metrics analyzed. We use the term *general* to underline that we must adapt the formula based on the simulation we are using to compute the simulated loss distribution. We study two simulation approaches: the Naive Monte Carlo simulation (NV simulation) and the method which combines the variance reduction techniques known as Cross-Entropy and Geometric Shortcut (CEGS simulation). In the following, we present how they work and show how the estimators 3.3 and 3.4 change. All the simulations share the raw structure as anticipated in flowchart 2.1.

3.1 Naive Monte Carlo simulation

The Monte Carlo method is a rather general name for any approach to risk measurement that involves the simulation of an explicit parametric model for risk-factor changes. To simulate the distribution of losses of our three Bernoulli mixture models, we need the conditional default probabilities, but firstly, looking at the flowchart 2.1, we must generate the random input vector of the systematic factors $\mathbf{\Psi}$. For every model, we assume that its probability density function is $f((\mathbf{\Psi}); \mathbf{u})$, where $\mathbf{u}$ is the set of the parameters of the density function specific for the model analyzed. Thus we simulate from $f((\mathbf{\Psi}); \mathbf{u})$, the realization of $\mathbf{\Psi}$ N times for all J obligors. If we collect all data, we end up with a $N \times J$ dimensional matrix. For each obligor $j = 1, ..., J$ and replication $k = 1, ..., N$ of $\mathbf{\Psi}_k$ we can calculate the $p_j(\mathbf{\Psi}_k)$. We recall that the $p_j(\cdot)$ is specific for each model. All these quantities are used to generate the default indicator Y_j:

$$Y_j \sim Bernoulli(p_j(\mathbf{\Psi})) \qquad j = 1, ..., J \tag{3.5}$$

Notice that if all data are stored in a matrix, $p_j(\cdot)$ has the same $N \times J$-dimension of the matrix of factors $\mathbf{\Psi}$ and Y_j are stored in a J-dimensional matrix. At last, we compute the

total loss L of the portfolio and the two risk metrics:

$$\hat{y}^{NV} = \frac{1}{N}\sum_{k=1}^{N} \mathbb{1}_{\{L^{(k)}>\tau\}} \tag{3.6}$$

$$\hat{r}_{NV} = \frac{\hat{x}_{NV}}{\hat{y}^{NV}} \tag{3.7}$$

where:

$$\hat{x}_{NV} = E[L\mathbb{1}_{\{L>\tau\}}] = \left(\sum_{k=1}^{N} L^k \mathbb{1}_{\{L^{(k)}>\tau\}}\right), \tag{3.8}$$

as reported in Basoglu et al. [2018]. N, the total number of replications, is set as large as possible to have good estimates, within the obvious constraints of computation time. For the sake of clarity, $\hat{y}^{NV}$, in 3.6, is the Naive counterpart of the tail loss probability estimate 3.3, and similarly $\hat{r}^{NV}$, in 3.7 ,the Naive counterpart of conditional excess estimate 3.4. In the following, there is the pseudo-code of the Naive simulation Algorithm 1, proposed by the authors.

Algorithm 1 Naive Monte Carlo simulation

1: **for** *replication* $k = 1, ..., N \ldots$ **do**
2: generate Ψ_d from $f(\ldots, \boldsymbol{u})$, d=1,..., D independently
3: compute $p_j(\boldsymbol{\Psi}),\ j = 1, ..., J$
4: generate $Y_j \sim Bernoulli(p_j(\boldsymbol{\Psi})),\ j = 1, ..., J$
5: compute $L^{(k)} = \sum_{j=1}^{J} c_j Y_j$
6: **end for**
7: return $\hat{y}^{NV}$ as in 3.6 and $\hat{r}_{NV}$ as in 3.7

In this algorithm, we have two sources of randomness: the generation of $\boldsymbol{\Psi}$ and $Y_j,\ j = 1, ..., J$. Basoglu et al. [2018] define the generation of the first *outer simulation*, to underline its exogeneity in all algorithm, and the second *inner simulation*, to underline the causality effect with the first. The variability of the *outer simulation* unavoidably affects the inner simulation, wasting, principally the computation of the tail loss probability estimates [1]. There is a propagation effect: the variance of the outer simulation of $\boldsymbol{\Psi}$ amplifies the variance of the inner simulation of Y_j, therefore the loss distribution L perceives the increase in the variance. Our problem related to the high variability belongs to a set of

[1] notice that the quality of the conditional excess estimate is a consequence of the quality of the tail loss probability.

general problems that arise, as in this case, in *rare-event simulation*. As anticipated in Chapter 1, our interest is to obtain a good simulation in the tail of the entire loss distribution. In the Naive Monte Carlo, most simulations are located, obviously, around the mean of loss, so they are influential in the tail estimation. There exists variance-reduction techniques that allow us to overcome such problems. As proposed by Basoglu et al. [2018], to reduce the variability of *outer simulation*[2], we employ importance sampling based on Cross-Entropy (CE) method. The problem with *inner simulation* is a sort of endogenous variance, due to the regeneration of a new set of random variables. Intuition suggests that we can recycle the yet-simulated variables, and taking the advantage of results of Sak and Hörmann [2012], we use the so-called Geometric Shortcut(GS) method.

3.2 CEGS simulation

In this section, we describe the new simulation method proposed by Basoglu et al. [2018] that combines both Cross-Entropy (CE) and Geometric Shortcut (GS) methods, which they call in abbreviated form the CEGS method. We present firstly the Geometric shortcut method, responsible for the *inner* variance reduction, i.e the variance in the simulation of Bernoulli variables Y_j. Then we describe the Cross-entropy that helps us to reduce the variance of the *outer* simulation, i.e in the simulation of the systematic factors $\mathbf{\Psi}$. The order in which they are proposed does not match the order they are implemented in the simulation. If we look at the flowchart 2.1, since we start simulating the $\mathbf{\Psi}$, we use first the Cross-entropy, and when we simulate the Bernoulli process, we use the Geometric Shortcut method.

3.2.1 Geometric shortcut

To give the idea behind this method, following Sak and Hörmann [2012], let's consider the easier case of default simulation: all the J obligors are independent, each of then has a proper default probability $p_j,\ j = 1, ..., J$ and relative exposure to default $c_j,\ j =$

[2]Main responsible parties for the problems in rare event simulation.

$1, ..., J$. To generate one realization of the loss L we simply simulate all default indicators $Y_1, Y_2, ..., Y_J$ and calculate $L = \sum_{j=1}^{J} c_j Y_j$; to obtain an independent sample of $L^{(i)}$ of size N we have to repeat that procedure N times. This is the simplistic case of what we have already done in Section 3.1. Credit risk portfolios with many obligors are relevant in practice which makes the naive approach for simulating L very slow. How can we speed up the Naive Monte Carlo simulation for this particular case? As we simulate N independent realization $L^{(i)}$ it is clear that we have to simulate a total of $N \times J$ default indicator A_{ij}. We precise that the index $i = 1, ..., N$ is related to realization, and j refers to obligors. As default probabilities are typically small, around 0.01 or less, the default indicator matrix A is sparse[3], we can speed up the simulation if we find a way to simulate the $L^{(i)}$ values by considering only the non-zero values of A_{ij}. Assume to fill it column-wise and observe that each column A_j of the matrix is holding an independent sequence of default indicators and is thus a Bernoulli sequence with parameters $p = p_j$ the default probability of the j-th obligor. Thus the random index I_1, where $A_{I_1 j}$ is equal to 1[4] for the first time, follows a geometric distribution with parameter $p = p_j$; the index of the second default $I_2 = I_1 + G$, where G is again a geometric random variate independent of I_1 and so on. This *geometric shortcut* idea allows one to jump over all zero entries directly to the next non-zero entry in a column. When a default is generated in this manner for the jth obligor (column) and the i-th independent repetition (row) we have to increment the vector of losses of all repetitions using $L^{(i)} = L^{(i)} + c_j$. Figure 3.1 presents how geometric shortcut works; notice that the obligors are fixed in the columns and repetitions in the rows. We underline that generating geometric random variates by repeated Bernoulli trials would not speed up the Naive simulation algorithm. Thus we generate geometric random variate with an inversion algorithm whose speed does not depend on p. We use:

$$G = \left\lceil \frac{\log(1-U)}{\log(1-p)} \right\rceil, \tag{3.9}$$

where G denotes the geometric random variate and U a $U(0,1)$ random variate.

[3] many zero elements

[4] Default occurred

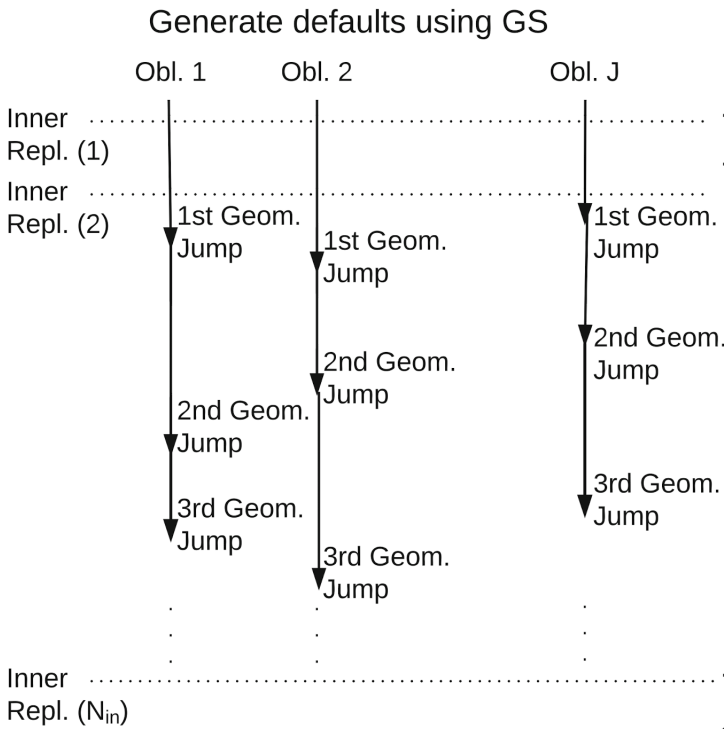


Figure 3.1: Geometric shortcut across obligors

We extend the previous example of independent obligors to the dependent case. In our models, the obligors are correlated by the presence of a common set of risk factors, and also by a copula model. We recall that a copula catches also the correlation undetectable by the standard Pearson's correlations. As seen in the Algorithm 1, conditional on realization of $\mathbf{\Psi}$, obligors default independently in our models. Let's start looking at some relevant structure properties of the Algorithm 1, enlighted by Sak and Hörmann [2012]. The algorithm consists of an outer part with N independent generations of the vector $\mathbf{\Psi}$ (line 1-2), and an inner part where, depending on the values of $\mathbf{\Psi}$, the default probabilities are calculated and the defaults are simulated (line 4). An important question is, how many times we should repeat the simulation of the default for a single fixed $\mathbf{\Psi}$-vector? We call this number N_{in}. In the Naive simulation one replication of $\mathbf{\Psi}$ is used one time to simulate the obligor's defaults. As Sak and Hörmann [2012] reported, it seems sensible to increase N_{in} for credit risk simulations as increasing the number of the inner repetitions is much cheaper than increasing the number of outer ones, especially if we consider the slow

functions that must be evaluated to calculate the $p_j(\mathbf{\Psi})$. This argument is even more valid if we consider the geometric shortcut idea that allows us to simulate the inner repetitions very fast. Sak and Hörmann [2012] decided to select N_{in} such that the computation time compared to the naive algorithm with $N_{in} = 1$ is not much increased. It can be proved that when $N = 1$ the computation time of the inner repetition of the naive algorithm is $O(m)$, compared when using the geometric shortcut is $O(N_{in} + J + N_{in}J\bar{p}(\mathbf{\Psi}))$. Thus we get the same complexity as for the Naive algorithm if we select:

$$N_{in} = \min\left(\frac{1}{\bar{p}(\mathbf{\Psi})},\ J\right), \tag{3.10}$$

$$\bar{p}(\mathbf{\Psi}) = \frac{1}{J}\sum_{j=1}^{J} p_j(\mathbf{\Psi}), \tag{3.11}$$

where $\bar{p}(\mathbf{\Psi})$ denotes the average value of the default probabilities $p_j(\mathbf{\Psi})$ for the current $\mathbf{\Psi}$-vector. Noticing the Algorithm 3, in line 6 we pre-compute the optimal number of the inner replication. In line 10-11 we perform the geometric shortcut, generating a geometric random variable with the formula 3.9, instead of the Bernoulli ones.

3.2.2 Cross-entropy method

In our *rare-event* environment, we want to increase the conditional default probability $p_j(\mathbf{\Psi})$. In literature, the common procedure is using importance sampling, and we briefly report how it works. To increase the observed frequency of default in the simulation, an easy way shared in most risk management papers, consider changing the scale parameter, to have fat tails in the distribution with or without the changing of the shift parameter. The basic idea of importance sampling is to use a different probability distribution, called the importance distribution, to generate the samples instead of the original distribution. Here we propose a variation on importance sampling based on the Cross-Entropy. The Cross-Entropy method, abbreviated as CE, provides a simple, efficient, and general method for solving complicated optimization problems. Its success relies also on estimating probabilities of rare events simulation. Here we discuss the main idea behind it, giving some theoretical definitions and re-proposing what's our aim.

Let $\mathbf{\Psi} = (\mathbf{\Psi}_1, ..., \mathbf{\Psi}_N)$ a multivariate random vector taking values in some space χ.

Assume that $f(\cdot;\mathbf{u})$ be a family of probability density functions on χ, with respect to some base measure ν. We consider $\mathbf{u}$ as vector of real-valued parameter. Thus for any measurable function H we have:

$$\mathbb{E}H(\mathbf{\Psi}) = \int_{\chi} H(\mathbf{\Psi}) f(\mathbf{\Psi};\mathbf{u}) \nu\left(d\mathbf{\Psi}\right) \tag{3.12}$$

where for simplicity we take $\nu\left(d\mathbf{\Psi}\right) = d\mathbf{\Psi}$. Let L be some real-valued function on χ. Suppose we are interested in the probability that $L(\mathbf{\Psi})$ is greater than or equal to some real value τ, under $f(\cdot;\mathbf{u})$. This probability can be expressed as:

$$\ell = \mathbb{P}_{\mathbf{u}}(L(\mathbf{\Psi}) \geq \tau) = \mathbb{E}_{\mathbf{u}} \mathbb{1}_{\{L(\mathbf{\Psi}) \geq \tau\}}. \tag{3.13}$$

If this probability is smaller than 10^{-3} we call $(L(\mathbf{\Psi}) \geq \tau)$ a *rare event*. This is our environment: L corresponds to the simulated loss distribution and the related risk measure is the event's probability of observing a loss greater than a value threshold. $\mathbf{\Psi}$ corresponds to our vector of systematic factors, i.e. $\mathbf{\Psi} = (\Psi_1, ..., \Psi_D)$. We have already seen a straightforward way to estimate ℓ with the Naive Monte Carlo simulation. Drawn a random sample $\mathbf{\Psi}_1, ..., \mathbf{\Psi}_N$ from original probability density function $f(\cdot;\mathbf{u})$. Our estimate is:

$$\frac{1}{N} \sum_{k=1}^{N} \mathbb{1}_{\{L(\mathbf{\Psi}_{(k)}) > \tau\}} \tag{3.14}$$

that corresponds to the Naive estimate $\hat{y}^{NV}$, shown in eq. 3.6. In this equation 3.14, we underline that the loss distribution is a function of the distribution of systematic risk factors. This estimator, for our risk metrics, is unbiased. However, a large simulation effort is required in order to estimate ℓ.

An alternative approach is based on importance sampling (IS): take a random sample $\mathbf{\Psi}_1, ..., \mathbf{\Psi}_N$ from an *importance sampling density* $g(\cdot)$, different from $f(;\mathbf{u})$. Assume that $g(\mathbf{\Psi} = \boldsymbol{\psi}) = 0 \;\; \Rightarrow \;\; \mathbb{1}_{\{L(\mathbf{\Psi}_{(k)}) > \tau\}} f(\mathbf{\Psi} = \boldsymbol{\psi}, \mathbf{u}) = 0$. Using the new density g we can represent ℓ as

$$\ell = \int \mathbb{1}_{\{L(\mathbf{\Psi}_{(k)}) > \tau\}} \frac{f(\mathbf{\Psi},\mathbf{u})}{g(\mathbf{\Psi})} g(\mathbf{\Psi})\, d\mathbf{\Psi} \;\; = \;\; \mathbb{E}_g \mathbb{1}_{\{L(\mathbf{\Psi}) \geq \tau\}} \frac{f(\mathbf{\Psi},\mathbf{u})}{g(\mathbf{\Psi})}, \tag{3.15}$$

where the subscript g means that the expectation is taken with respect to g. An unbiased estimator of ℓ is:

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{L(\mathbf{\Psi}_{(i)}) > \tau\}} W(\mathbf{\Psi}_i) \;\; = \;\; \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{L(\mathbf{\Psi}_{(i)}) > \tau\}} \frac{f(\mathbf{\Psi}_i;\mathbf{u})}{g(\mathbf{\Psi}_i)}, \tag{3.16}$$

where $\hat{\ell}$ is called *importance sampling* (IS) or the *likelihood ratio* (LR) estimator and $W(\mathbf{\Psi}_i)$ is called the *likelihood ratio* (LR). Notice that in the particular case where there is no change of measure, i.e $g = f$, we have that $W = 1$, and the LR estimator reduces to the Naive estimator 3.14. What is the best choice of g? Clearly, if we estimate ℓ using the change of measure with density

$$g^*(\mathbf{\Psi}) := \frac{\mathbb{1}_{\{L(\mathbf{\Psi}_{(i)})>\tau\}} f(\mathbf{\Psi}_i; \mathbf{u})}{\ell} \tag{3.17}$$

Substituting 3.17 in 3.16, we have:

$$\mathbb{1}_{\{L(\mathbf{\Psi}_{(i)})>\tau\}} \frac{f(\mathbf{\Psi}_i; \mathbf{u})}{g^*(\mathbf{\Psi}_i)} = \ell \tag{3.18}$$

for all i-th replications. In conclusion, we end up with the estimator 3.16, which has zero variance, and we need to produce only $N = 1$ sample.

The drawback is that g^* depends on the unknown parameter ℓ. It is often convenient to choose g in the family of densities $f(\cdot; \mathbf{u})$. The idea now is to choose the parameter vector, called the *reference parameters*, either *importance parameter* or *tilting parameter* $\mathbf{v}$ such that the distance between two densities g^* and $f(\cdot; \mathbf{v})$ is minimal. An innovative and convenient measure of distance between two densities g and h is the *Kullback-Leibler distance*, which is also called the *Cross-Entropy* between g and h.

Definition 3 *the Kullback-Leibler distance is defined as:*

$$\mathcal{D}(g, h) = \mathbb{E}_g \left[\log \frac{g(\mathbf{\Psi})}{h(\mathbf{\Psi})} \right] = \int g(\psi) \log \left(g(\psi) \right) d\psi - \int g(\psi) \log \left(h(\psi) \right) d\psi. \tag{3.19}$$

As de Boer et al. [2005] write, we note that $\mathcal{D}$ is not a "distance" in the formal sense, in fact, it is not symmetric.

Minimizing the *Cross-Entropy* between g^* and the new density $f(\cdot; \mathbf{v})$ is equivalent to choosing $\mathbf{v}$ such that $-\int g^*(\mathbf{\Psi}) \log \left(f(\mathbf{\Psi}; \mathbf{v}) \right) d\mathbf{\Psi}$ is minimized, which is equivalent to solving the problem:

$$\max_{\mathbf{v}} \int g^*(\mathbf{\Psi}) \log \left(f(\mathbf{\Psi}; \mathbf{v}) \right) d\mathbf{\Psi} \tag{3.20}$$

Replacing g^* from eq.(3.17) in (3.20), we obtain:

$$\max_{\mathbf{v}} \int \frac{\mathbb{1}_{\{L(\mathbf{\Psi})>\tau\}} f(\mathbf{\Psi}; \mathbf{u})}{\ell} \log \left(f(\mathbf{\Psi}; \mathbf{v}) \right) d\mathbf{\Psi}, \tag{3.21}$$

neglecting ℓ, it is equivalent to the maximization program:

$$\max_{\mathbf{v}} \int \mathbb{1}_{\{L(\mathbf{\Psi})>\tau\}} \log\left(f(\mathbf{\Psi};\mathbf{v})\right) f(\mathbf{\Psi};\mathbf{u})\, d\mathbf{\Psi}. \tag{3.22}$$

The last expression constitutes the main result of applying the Cross-Entropy (CE) method to our general problem. For our context, the problem 3.22 is a stochastic optimization problem as the indicator function is uncertain for a given vector $\mathbf{\Psi}$. It depends on the Bernoulli random variables $Y_1, ..., Y_J$. Recall that, in our simulation, $\mathbf{u} \in \mathbb{R}^D$ represents the vector of the parameters of the original distribution, and the $\mathbf{v} \in \mathbb{R}^D$ are the reference parameters that are looking for. We rewrite the problem 3.22 replacing the indicator function with $E[\mathbb{1}_{\{L(\mathbf{\Psi})>\tau\}}|\mathbf{\Psi}] = P(L(\mathbf{\Psi}) > \tau|\mathbf{\Psi})$. We obtain the solution of problem 3.22 estimating the objective function with Monte Carlo simulation:

$$\max_{\mathbf{v}} \quad \sum_{k=1}^{M} P(L(\mathbf{\Psi}^{(k)}) > \tau|\mathbf{\Psi}^{(k)}) \log\left(f(\Psi^{(k)};\mathbf{v})\right) \tag{3.23}$$

where M is the number of replication and $\mathbf{\Psi}^{(1)}, ..., \mathbf{\Psi}^{(M)}$ are independently generated from $f(.;\mathbf{u})$. Our aim, in the general view of the simulation, is to find the optimal reference parameters $\mathbf{v}^*$ in order to, better simulate the systematic factors $\mathbf{\Psi}$ from $f(\mathbf{\Psi};\mathbf{v}^*)$ and thus the loss distribution L. We need to, further reformulate the maximization problem. The reformulations that we will apply in the following are allowed to the distribution properties of $\mathbf{\Psi}$. Let's have a look at the distribution of $\mathbf{\Psi}$ in the three model analyzed:

- CreditRisk+ : the components of vector $\mathbf{\Psi}$ follow $Gamma(\alpha, \beta)$ distribution;
- Gaussian copula model: the components of vector $\mathbf{\Psi}$ follow $N(0,1)$ distribution;
- t-copula model : the first $D-1$ components of $\mathbf{\Psi}$ follow $N(0,1)$ distribution and the D component follow χ^2 distribution.

All of them belong to the exponential family. These considerations are relevant to give a suitable way to recast the first term on the left of the optimization problem 3.23. Thanks to the work of Glasserman and Li [2004] use the approximation:

$$P(L(\mathbf{\Psi^{(k)}}) > \tau|\mathbf{\Psi}) \approx 1 - \Phi\left(\frac{\tau - \sum_{j=1}^{J} c_j p_j(\mathbf{\Psi}^{(k)})}{\sqrt{\sum_{j=1}^{J} c_j^2 \left[p_j(\mathbf{\Psi}^{(k)}) - p_j(\mathbf{\Psi}^{(k)})^2\right]}}\right) \tag{3.24}$$

The remaning part of the objective function of 3.23 to reformulate is: $\log\left(f(\mathbf{\Psi}^{(\mathbf{k})}, \mathbf{v})\right)$. The dimension of our problem is dependent on $\mathbf{v} \in \mathbb{R}^D$, i.e we have to optimize with respect to D variables.

In Basoglu et al. [2018], the authors, suggest combining identical D'-th elements of $\mathbf{\Psi}$, $(\Psi_1, .., \Psi_{D'})$ in a group, to reduce the dimension of the problem. Instead of D variables, we have only one optimization variable θ for the first D':

$$v_d = \sum_{j=1}^{J} a_{jd} c_j \theta, \qquad d = 1, ..., D'. \tag{3.25}$$

This is justified by the fact that we want to find the maximum increasing direction of loss function 3.26 considering only D' components:

$$\frac{\left(\sum_{j=1}^{J} a_{j1} c_j, ..., \sum_{j=1}^{J} a_{jD'} cJ\right)'}{\|\left(\sum_{j=1}^{J} a_{j1} c_j, ..., \sum_{j=1}^{J} a_{jD'} cJ\right)'\|}. \tag{3.26}$$

The optimization variable θ plays the same role as D' variable and the remaining $v_{D'+1}, ..., v_D$ variables remain free from a further combination. They constitute the Importance sampling parameters, as well as the variable of the problem 3.23, whose dimension now is reduced to $D - D' + 1$. We look now at how to construct the group of identical elements i.e. for each model, observing the structure of the conditional independent default probability $p_j(\mathbf{\Psi})$[5]. In our models, following Basoglu et al. [2018]:

- CreditRisk+: from 2.15, all element of $\mathbf{\Psi}$ come from the same *Gamma* distribution. Therefore we choose $D' = D$: one optimization variable θ is involved.
- Gauss copula model: from 2.28, all element of $\mathbf{\Psi}$ come form the same *Standard Normal* distribution. Therefore we choose $D' = D$: one optimization variable θ is involved.
- t-copula model: from 2.33, the first $D-1$ element of $\mathbf{\Psi}$, i.e $\Psi_1, ..., \Psi_{D-1}$, come from the same *Standard Normal* distribution. Therefore we can set $D' = D - 1$. Recall that the last factor Ψ_D follows a *Chi-squared* distribution. We conclude by this,

[5] expecially on vector $\mathbf{\Psi}$.

that the problem considered with the t-copula model, will have two optimization variables, θ and v_D.

As anticipated in Chapter 2, in all models the single components, i.e. factors, of vector $\mathbf{\Psi}$, are independent of each other. Taking all this result, we can reformulate the problem 3.23 in such a way:

$$\begin{aligned}\max_{\theta, v_{D'+1},\dots,v_D} \quad & \textstyle\sum_{k=1}^{M} P(L(\mathbf{\Psi}^{(k)}) > \tau | \mathbf{\Psi}^{(k)}) \left[\sum_{d=1}^{D'} \log\left(f\left(\Psi_d^{(k)}; \sum_{j=1}^{J} a_{jd} c_j \theta \right) \right) \right. \\ & \left. + \textstyle\sum_{d=D'+1}^{D} \log\left(f\left(\Psi_d^{(k);v_d} \right) \right) \right]\end{aligned} \tag{3.27}$$

where we use the 3.24, in order to calculate the $P(L(\mathbf{\Psi}^{(k)}) > \tau | \mathbf{\Psi}^{(k)})$. We can summarize how the *Cross-Entropy* method and its parameters optimization works step by step, showing a generic pseudo code in the Algorithm 2.

Algorithm 2 Cross Entropy method, for a grouping of first D' elements of $\mathbf{\Psi}$

1: generate $\Psi_d^{(k)}$ d=1,..., D
2: solve for θ, $v_{D'+1}, ..., v_D$ using 3.27
3: compute v_d, $d = 1, ..., D'$ using 3.25
4: return $v_1, ..., v_D$

Once exploited the geometric shortcut, and cross entropy method, we just have to compute the estimations of tail loss probability and conditional excess, using importance sampling. For the moment we focus on tail loss probability estimation because, as we will see after, it is a building block of the second estimate. Following A. J. McNeil [2005], if we have to estimate a quantity from a Bernoulli mixture model, using the importance sampling technique we can use the general formula:

$$\hat{\theta}_n^{IS} = \frac{1}{n} \sum_{i=1}^{n} \rho(\mathbf{\Psi}^{(k)}; \mathbf{u}, \mathbf{v}) \hat{\theta_{n1}}^{IS,1}(\mathbf{\Psi}_i) \tag{3.28}$$

where the $\hat{\theta_{n1}}^{IS,1}$ represents the naive estimator of the quantity using new parameters. $\rho(\mathbf{\Psi}^{(k)}; \mathbf{u})$ is the likelihood ratio computed under the new importance sampling parameters[6]. Let's recast in our case of computation of tail loss probability, the general formula.

[6]the index k in the formula indicate the iteration in the simulation, and the number n, the total number of iteration. As n increases, the better the final estimate obtain

The naive estimator part is the same as in 3.6 but with the difference that we take also into account the geometric shortcut. At the moment we are limiting to changing the number of replication from N to N_{in}. Therefore estimator is:

$$\bar{p}_{in}^{(k)} = \frac{1}{N_{in}} \sum_{l=1}^{N_{in}} \mathbf{1}_{\{L_{in}^{l} > \tau\}} \tag{3.29}$$

The likelihood ratio, instead, looks like this:

$$\rho(\mathbf{\Psi}^{(k)}; \mathbf{u}, \mathbf{v}) = \prod_{d=1}^{D} f(\Psi_d^{(k)}; u_d) \left(f(\Psi_D^{(k)}; v_d \right)^{-1} \tag{3.30}$$

where u_d refers to all the original parameters of factors distribution and v_d to the new parameters obtained from the cross-entropy method. Combining 3.29 and 3.30 the estimator of tail loss probability $P(L > \tau)$ using cross-entropy and geometric shortcut (CEGS) is:

$$\hat{y}^{CEGS} = \frac{1}{N} \sum_{k=1}^{N} \rho^{(k)} \bar{p}_{in}^{(k)} \tag{3.31}$$

At this point, we focus on the estimator of conditional excess for the CEGS method. As in 3.4, we need the quantity $\hat{x}^{CEGS}$ and $\hat{y}^{CEGS}$. The last is already computed in 3.31. For the first we report the results by Sak and Hörmann [2012] and Glasserman [2005]. In this case, all the computations involve the new importance sampling distribution of loss. The geometric shortcut appears in the computation of the average of the inner replication loss. Given the loss for a fixed outer replication k:

$$L^{(k,l)} \mathbf{1}_{\{L^{(k,l)} > \tau\}} \qquad for\ l = 1, ..., N_{in}^{(k)} \tag{3.32}$$

where l is the index sum. The average of inner replication loss is:

$$\bar{L}_{in}^{(k)} = \frac{1}{N_{in}^{(k)}} \left(\sum_{l=1}^{N_{in}^{(k)}} L^{(k,l)} \mathbf{1}_{\{L^{(k,l)} > \tau\}} \right) \tag{3.33}$$

We point out that $N_{in}^{(k)}$ is the number of inner repetitions for the kth outer replication. Therefore the conditional excess estimate $\hat{r}^{CEGS}$ is:

$$\hat{r}^{CEGS} = \frac{\hat{x}^{CEGS}}{\hat{y}^{CEGS}} = \frac{\sum_{k=1}^{N} \rho^{(k)} \bar{L}_{in}^{(k)}}{\sum_{k=1}^{N} \rho^{(k)} \bar{p}_{in}^{(k)}} \tag{3.34}$$

3.2.3 How to implement CEGS simulation

We presented all the elements required to simulate with the CEGS method, anticipating the final product of this method, i.e the estimates of the risk metrics. But, we left where the cross entropy and the geometric shortcut have to locate in the steps of the entire simulation. Before giving the full algorithm, as made in Algorithm 1, we describe all passages required, thanks also to the flowchart in fig. 3.2. We treat this part in a generic form, valid for all three models.

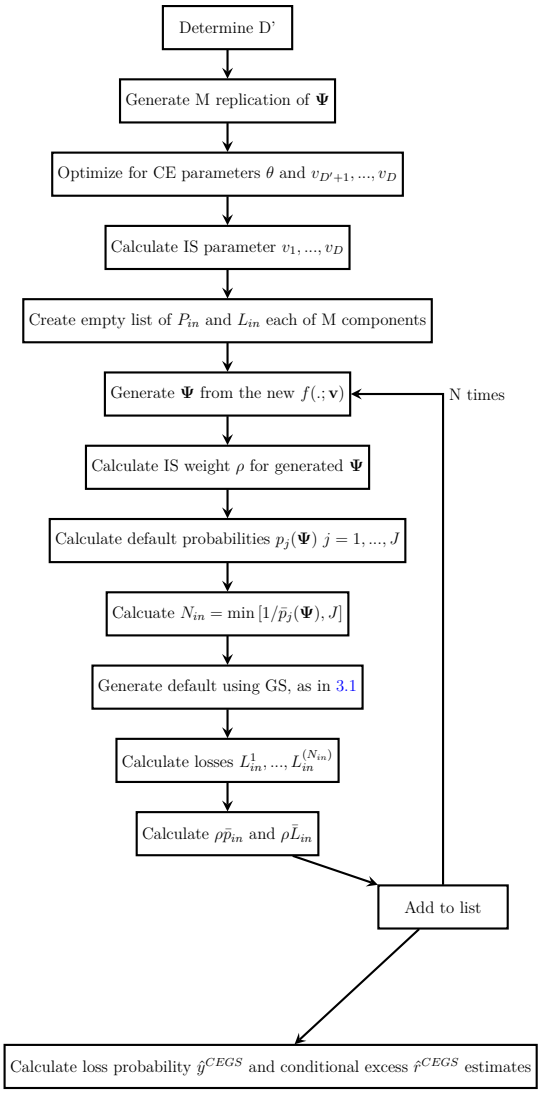


Figure 3.2: Flowchart of CEGS method

Firstly we have to determine how many factors, i.e components of vector $\mathbf{\Psi}$, we can combine in the group of the identical element, as introduced in section 3.2.2. We, therefore, choose the right value of D' according to the model. The cross-entropy method starts after we have simulated the systematic factor for a fixed number M. We use this simulation to build the objective function of problem 3.27. Recall that the importance sampling parameters combined are represented in the problem by a single optimization variable θ and the remaining parameter, are free optimization variables. To keep it clear, the third block of the flowchart is further developed in Figure 3.3. At the 5th block of flowchart, we

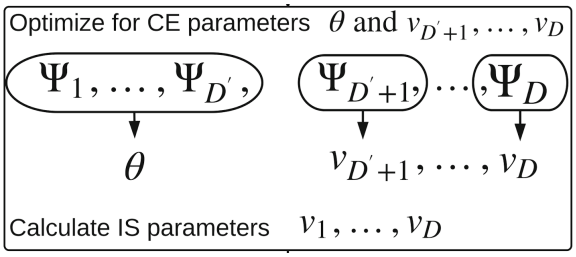


Figure 3.3: Details of third block of flowchart of CEGS method

prepare the elements required to store the result of the incoming Monte Carlo simulation. Once we obtained the importance sampling parameters $\mathbf{v}$, we can simulate from the new distribution $f(.;\mathbf{v})$: this is the starting point of the simulation. We generate N samples of systematic factor $\mathbf{\Psi}^{(k)}$ [7] consequently the likelihood ratio $\rho(\mathbf{\Psi};\mathbf{u},\mathbf{v})$ (3.30), and the conditional probability for each obligor $p_j(\mathbf{\Psi})$. A preliminary phase to apply the geometric shortcut method is computing the optimal number of inner replication (3.10). Then we simulate the default for each obligor which is a geometric random variable instead of Bernoulli, using (3.9). Recall also that the geometric shortcut method exploits the already simulated $\mathbf{\Psi}$. For each inner replication N_{in}, having simulated default across J obligors, we compute the loss $L_{in}^{(k)}$. Figure 3.4 gives a clear view of how the 10th and 11th blocks are linked. Having collected all losses and conditional probability, we can compute the quantities $\bar{p}_{in}$ and $\bar{L}_{in}$, i.e the average of losses and the average of conditional default probability. These two values are fundamental to compute the estimates of interest: the

[7] k denote the index of simulation, $k = 1, ..., N$.

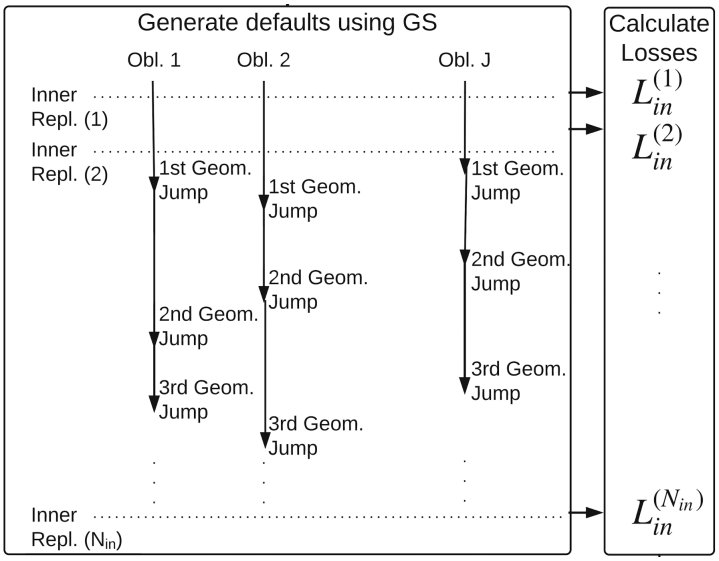


Figure 3.4: Detailed link of 10th an 11th block of flowchart of CEGS method

tail loss probability $\hat{y}^{CEGS}$ and the conditional excess $\hat{r}^{CEGS}$. In the following, we report the Algorithm 3 proposed by Basoglu et al. [2018], which computes the estimates of the risk metrics using cross-entropy and geometric shortcut methods.

Algorithm 3 Tail loss probability and conditional excess with CEGS simulation

determine D', then solve for $\mathbf{v}$ using algorithm 2
for *replication* $k = 1, ..., N$ **do**
    generate Ψ_d from $f(\dots, \boldsymbol{v})$, d=1,..., D independently
    calculate $\rho^{(k)}$ as in 3.30
    compute $p_j(\boldsymbol{\Psi}),\ j = 1, ..., J$
    construct loss vector L_{in} of size $N_{in} = \min\left([1/\bar{p}_j(\boldsymbol{\Psi})], J\right)$
    for obligors $j = 1, ..., J$ **do**
        inizialize λ to zero and *cont* to false
        while *cont* = *true* **do**
            generate the $U(0,1)$ random variate U
            set $\lambda = \lambda +$ ceil$(\log(1-U)/log(1-p_j())$
            if $(\lambda > N_{in})$ **then**
                cont=true
            elseset $L_{in}^{\lambda} = L_{in}^{\lambda} + c_j$
            end if
        end while
    end for
    compute $\bar{p}_{in}^{(k)} = \frac{1}{N_{in}} \sum_{l=1}^{N_{in}} \mathbf{1}_{\{L_{in}^{l} > \tau\}}$, as in 3.29
    compute $\bar{L}_{in}^{(k)} = \frac{1}{N_{in}} \left(\sum_{l=1}^{N_{in}^{(k)}} L^{(k,l)} \mathbf{1}_{\{L^{(k,l)} > \tau\}} \right)$, as in 3.33
end for
return $\hat{y}^{CEGS}$ as in 3.31 and $\hat{r}^{CEGS}$ as in 3.34

Chapter 4

Models settings and simulations implementation

In the previous chapters, we faced all the theoretical concepts needed to fully understand how the simulations of tail loss probability and conditional excess work. Starting from credit risk definitions, we gave an overview of the models, used in this work, to simulate the default of obligors. In Chapter 3 are presented all elements required to address the new method as opposed to the Naive one. Now in this Chapter, we explain how to implement all the models and the simulations giving the related codes. We decide to use MATLAB software (produced by MathWorks [2022]), with the support of the toolboxes: *Financial Toolbox*, *Optimization Toolbox*, *Statistics and Machine learning Toolbox* and *Symbolic Math Toolbox*. For reproducibility reasons, we take the same data used in the Basoglu et al. [2018], in particular, the parameters of models come for others papers on the Credit risk topic. They are considered the common test values for credit risk simulations. The last two sections of the chapter report the code of the related method and the results.

4.1 Parameters setting

4.1.1 CreditRisk+ parameters

For the CreditRisk+ model we use the same parameters as proposed in Glasserman and Li [2004]. Recalling the model formula in 2.15 and the total portfolio loss 2.10 we consider a portfolio of:

- $J = 1000$ obligors;
- each obligor has an exposure value $c_j = 0.004 + 0.00196j$ for $j = 1, ..., J$;
- a_{j0}, the weight of the first systematic factor is equal to 0.02;
- the remaining weights of factors are set to $a_{jd} = 0.002$ for $d = 1, ..., D = 10$;
- assume $\sigma_d = 9$ for $d = 1, ..., D$.

Here we propose the MATLAB function that computes the conditional default probability 2.15 for a single obligor j and the code that set the parameter for this model.

```
function [p_plus,psi] = creditriskplus(d,sigma,a,a0)
% CREDITRISK+
alpha = sigma.^(-2);
beta  = sigma.^2;
psi = random('Gamma',alpha,beta,d,1);
p_plus = 1- exp(-a0 -a*psi);
end
```

```
%% Data creditrisk+ [p_plus] = creditriskplus(d,sigma,a,a0)
j= 1000;
d= 10;
c= 0.04 + 0.00196*[1:j]';
a0= 0.002;
a= 0.0002*ones(j,d);
sigma= 9;
p_criskplus=creditriskplus(d,sigma,a,a0);
```

Notice that in the CreditRisk+ model, all parameters are deterministic, therefore they are not a source of randomness, especially referring to the weight of systematic factors. As we will see in the next models we will relax this assumption.

4.1.2 Gaussian copula model parameters

Following Basoglu et al. [2018], the authors take data from Glasserman [2005] for the Gaussian copula model. We consider a portfolio of:

- J=1000 obligors;
- $d = 1, ..., D = 10$, thus it is a 10-factor model;
- the marginal default probabilities are $p_j = 0.01(1 + \sin(16\pi j/J))$ for $j = 1, ..., J$. Notice that $p_j \in [0, 2\%]$;
- obligors have an exposure value $c_j = (\lceil 5j/J \rceil)^2$ where $\lceil . \rceil$ represent the ceiling function;
- the factor loadings a_j are generated independently and uniformly form the interval $(0,1/\sqrt{10})$ [1]; as Basoglu et al. [2018] report, this choice aims to ensure that the Condition 2.26 will be satisfied. The upper limit of the interval implies that, for some of the obligors, the sum of the squares of elements of a_j is close to 1: this credit portfolio contains strongly correlated obligors. We obtain the b_j coefficents inverting the Condition 2.26.

The MATLAB function that computes the conditional default probability of the Gaussian copula model 2.28 and the parameters of the portfolio are here proposed. In the code, we use the same notation adopted in Chapter 2.

```
function [p_gausscop,Zbig,z] = gaussiancopulamodel(j,d,p,a,b)
% GAUSSIANCOPULAMODEL
% info a must be (j,d)
% info b must be (j,1)
eps = randn(j,1); %idiosincratic risk factor
psi = randn(d,1); % systematic risk factor
z= icdf('Normal',1-p,0,1);
Zbig= b'.*eps + a* psi;
p_gausscop = cdf("Normal", a*psi + icdf('Normal',p,0,1).*(1./
   b),0,1);
```

[1] as anticipated, opposed to CreditRisk+, the factor loadings in Gaussian copula model are not fixed, but random.

```
end
```

```
%% Data gaussian copula model [p_gausscop,Zbig,z] =
   gaussiancopulamodel(j,d,p,a,b)
j= 1000;
d= 10; %10 factor model
p=0.01*(1+sin(16*pi/j*[1:j]'));
c= (ceil(5/j*[1:j]')).^2;
a = random('Uniform',0,1/sqrt(10),j,d);  %rows are obligors
   and columns are systematic factor
b = sqrt(ones(j,1)-sum(a.^2,2));
p_gausscop=gaussiancopulamodel(j,d,p,a,b);
```

4.1.3 t-copula model parameters

In the t-copula model, Basoglu et al. [2018] take the same data from Sak and Hörmann [2012]. The model considers a portfolio of:

- $J = 1200$ obligors;
- $D = 5$ factors;
- the marginal default probabilities p_j, for each obligor, are generated independently and uniformly from the interval [0,0.02];
- the exposure level of each obligor is $c_j = (\lceil 20j/J \rceil)^2$ for $j = 1, ..., J$, where $\lceil \cdot \rceil$ represent the ceiling function;
- the factor loadings values follow the example in Sak and Hörmann [2012]. Assume to divide all obligors into 6 segments of size 200. For each segment, the factors follow a uniform distribution $U(0, max)$, according to the table in Fig. 4.1.
- Recalling that Ψ_D is a Chi-square random variable, we set $\eta = 5$, the number of degree of freedom;

As usual they follow the MATLAB function of the conditional default probability 2.33 and the code that imports parameters data.

Segment	Obligor j	$a_{j,1}$	$a_{j,2}$	$a_{j,3}$	$a_{j,4}$	$a_{j,5}$
$1A$	$1-200$	U(0,0.5)	$U(0,0.5)$	$U(0,0.1)$		
$1B$	$201-400$	$U(0,0.5)$	$U(0,0.1)$	$U(0,0.5)$		
$2A$	$401-600$	$U(0,0.4)$		$U(0,0.3)$	$U(0,0.1)$	
$2B$	$601-800$	$U(0,0.4)$		$U(0,0.1)$	$U(0,0.3)$	
$3A$	$801-1000$	$U(0,0.5)$			$U(0,0.4)$	$U(0,0.3)$
$3B$	$1001-1200$	$U(0,0.5)$			$U(0,0.3)$	$U(0,0.4)$

Figure 4.1: Distribution used to generate the factor loadings of t-copula model

```
function [p_cond,T,t] = tcopulamodel(j,d,p,c,b,a_t,v)
eps = randn(j,1);
psi_t  = randn(d-1,1);
psi_D =  random('Chisquare',v);
T= (b.*eps + a_t*psi_t)*((psi_D/v).^(-0.5)); % latent
   variables
t = icdf('T',1-p,v); %threshold
p_cond = cdf('Normal',(a_t*psi_t - sqrt(psi_D/v)*icdf('T',1-p
   ,v)).*(1./b),0,1);
end
```

```
%% Data t copula model [p_cond,T,t] = tcopulamodel(j,d,p,c,b,
   a_t,v)
j=1200; % total obligors
d=5; % 5 factor model;
p= random('Uniform',0,0.02,j,1); % marginal default
   probability
c= (20/j*[1:j]').^2; % obligors exposure
a = [random("Uniform",0,0.5,200,2) random("Uniform
   ",0,0.1,200,1) zeros(200,2);
       random("Uniform",0,0.5,200,1) random("Uniform
          ",0,0.1,200,1) random("Uniform",0,0.5,200,1) zeros
          (200,2);
       random("Uniform",0,0.4,200,1) zeros(200,1) random("
          Uniform",0,0.3,200,1) random("Uniform",0,0.1,200,1)
           zeros(200,1);
       random("Uniform",0,0.4,200,1) zeros(200,1) random("
          Uniform",0,0.1,200,1) random("Uniform",0,0.3,200,1)
           zeros(200,1);
       random("Uniform",0,0.5,200,1) zeros(200,2) random("
          Uniform",0,0.4,200,1) random("Uniform",0,0.3,200,1)
          ;
```

```
        random("Uniform",0,0.5,200,1) zeros(200,2) random("
           Uniform",0,0.3,200,1) random("Uniform",0,0.4,200,1)
           ];
a_t = a(:,1:d-1);
b = sqrt(ones(j,1)-sum(a_t.^2,2));
v=5; %degree of freedom of Psi_D
p_tcop = tcopulamodel(j,d,p,c,b,a_t,v);
```

4.2 Naive Monte Carlo (NV) implementation

We give in this section the code that implements the naive Monte Carlo simulation. We set the number of replication $N = 10000$ valid for all models, but we change the threshold values τ according to Basoglu et al. [2018]:

- CreditRisk+ : $\tau = [8, 13.8, 19.7]$;
- Gaussuan copula model: $\tau = [250, 950, 2000]$;
- t copula model: $\tau = [4500, 16500, 34000]$;

Using the previous code of Section 4.1 we can implement the following code comparing with the Algorithm 1. We divide the code into sections for each model. Notice at the end of the sections we compute the risk metrics: the variable `yNV` followed by the name of the model is the estimate of tail loss probability and `rNV` is the estimate of conditional excess.

```
%% Naive simulation of tail loss probability and conditional
   excess for CREDITRISK+
% remeber first run the section of data of model
N = 10000; %number of replication
k =0;
L = zeros(N,1);
tau_crplus = [8 13.8 19.7];
for k=1:N
    p_criskplus=creditriskplus(d,sigma,a,a0);
    Y = random("Binomial",1,p_criskplus);
    L(k)= c'*Y;
end

yNV_crplus = 1/N* (sum(L>tau_crplus,1))
rNV_crplus = (L'*(L>tau_crplus))./(sum(L>tau_crplus,1))
```

```
15
16
17 %% Naive simulation of tail loss probability for GAUSSIAN
       COPULA MODEL
18 % remeber first clear and close then run the section "Data
       gaussian
19 % copula model" and finally the following section
20 N = 10000; %number of replication
21 k =0;
22 L = zeros(N,1);
23 tau_gausscop = [250 950 2000];
24 for k=1:N
25     p_gausscop=gaussiancopulamodel(j,d,p,a,b);
26     Y = random("Binomial",1,p_gausscop);
27     L(k)= c'*Y;
28 end
29
30 yNV_gausscop = 1/N* (sum(L>tau_gausscop,1))
31 rNV_gausscop = (L'*(L>tau_gausscop))./(sum(L>tau_gausscop,1))
32
33 %% Naive simulation of tail loss probability and conditional
       excess for T COPULA MODEL
34 % remeber first clear and close then run the section "Data t
       copula model"
35 % and finally the following section
36 N = 10000; %number of replication
37 k =0;
38 L = zeros(N,1);
39 tau_tcop = [4500 16500 34000];
40 for k=1:N
41     p_tcop = tcopulamodel(j,d,p,c,b,a_t,v);
42     Y = random("Binomial",1,p_tcop);
43     L(k)= c'*Y;
44 end
45
46 yNV_tcop = 1/N* (sum(L>tau_tcop,1))
47 rNV_tcop = (L'*(L>tau_tcop))./(sum(L>tau_tcop,1))
```

4.3 CEGS implementation

In this Section are proposed the implementations of the cross-entropy geometric shortcut method for the three models. For each model, we repropose the import of parameters

because we cannot use the functions introduced before. This is motivated by the software that we are using: MATLAB is more efficient with vectorized code, therefore the abuse of `for` loops can surely destroy the simulation. The Algorithm 3 can not be implemented with the same structure and has to be adapted to the coding environment. Most random generations are obtained by a single line code without performing `for` loops.

4.3.1 CreditRisk+

We divide the code into three parts. In the first, we vectorize the parameters setting discussed in Section 4.1. Then we perform the cross-entropy algorithm to compute the importance sampling parameters `v`. In the third part, there is the simulation equipped with the geometric shortcut technique. The main result that we have to reach in the first codes section (up to line 22) is the computation of conditional default probability, i.e `p_criskplus_matrix`. We start the code by setting the number of total replications N and the replications useful to reach the optimal parameters in cross-entropy algorithm M. A *priori* we need to decide what D' is, depending on the model. We can take the result discussed in Section 3.2.2. In the second section, we solve the optimization problem 3.23 that we rewrite in minimized sense. In the case of the CreditRisk+ model, the optimization is unconstrained and one single optimization variable is involved: at first glance, can be used the simple solver `fminunc`. Nevertheless, we are neglecting that the objective function 3.27 is highly nonlinear, therefore a sensible solution is to perform optimization with *genetic algorithm*[2]. This choice allows us to overtake the problem with a proper feasible initial starting point. The code also reported a nice plot in Figure 4.2, using *Symbolic Math Toolbox*, of the objective function. It is useful to have a visual check of convexity and consequently to observe directly where the minimum of the function is. Obtaining the importance sampling parameter we can start with the simulation following the conceptual step in the previous Flowchart 3.2. In this part, we take the advantage of the geometric shortcut method using the inner replication N_{in} and simulating the default event from geometric distribution. Notice that the $53th$ and $54th$ lines of the code are

[2] `ga` function in MATLAB

responsible for the simulation. Finally, we compute the estimate of tail loss probability, `yCEGS_crplus` in the code, and the estimate of conditional excess `rCEGS_crplus`.

```
%% CEGS implementation for CREDITRISK+
clc
clear
close
j= 1000;
d= 10;
c= 0.04 + 0.00196*[1:j]';
a0= 0.002;
a= 0.0002*ones(j,d);
sigma= 9;
M = 1000; % replication for optimization CE
N= 10000; % outer replication for simulation

alpha = sigma.^(-2);
beta  = sigma.^2;
tau_crplus = [8 13.8 19.7];

% compute the psi_d^k matrix
% rng('default')
psi= random('Gamma',alpha,beta,d,M);

p_criskplus_matrix= 1- exp(-a0 -a*psi);

%% Cross entrpy CE method to compute v importance sampling
   paramters
% compute first part of objective function
obj_part1  = 1-cdf("Normal", (tau_crplus(1)-c'*
   p_criskplus_matrix)./sqrt((c'.^2)*(p_criskplus_matrix-
   p_criskplus_matrix.^2)),0,1);

fun = @(theta) -obj_part1*(log(gampdf(psi,((theta*(a'*c))
   .^-2).*ones(d,M),((theta*(a'*c)).^2).*ones(d,M)))'*ones(d
   ,1))
fplot(fun,[20 70]) % plot of objective function
nvars=1;
options = optimoptions('ga','ConstraintTolerance',1e-9);
[opt_theta_ga ,fval]= ga(fun,nvars,[],[],[],[],1,[],[],
   options)
v= opt_theta_ga*(a'*c);

%% CEGS simulation
N= 10000;
```

```
psi= random("Gamma",(v.^-2).*ones(d,N),(v.^2).*ones(d,N),d,N)
    ;
f_u=gampdf(psi,alpha,beta);
f_v=gampdf(psi,(v.^-2).*ones(d,N),(v.^2).*ones(d,N));
rho= prod(f_u ./ f_v,1);
p_criskplus_matrix= 1- exp(-a0 -a*psi); %it is a matrix where
    the columns contain k=1...N replication infact #col is
   =10000
N_in= round(min(1./(mean(p_criskplus_matrix,1)),j*ones(1,N)))
    ;
L_in={};
for  i=1:length(N_in)
    L_in(i,:)={zeros(1,N_in(i))};
end
% test the loop
for k=1:N
    for i =1:j
        lambda=0;
        cont=false;
        while cont==false
            U=random("Uniform",0,1);
            lambda= lambda + ceil(log(1-U)./log(1-
               p_criskplus_matrix(i,k)));
            if lambda>N_in(k)
                cont=true;
            else
                if i<=length(L_in{k,1})
                    L_in{k,1}(i)= L_in{k,1}(i)+c(i);
                else
                    break
                end
            end
        end
    end
end

k=0;
for k=1:N
    p_in(k)= 1/N_in(k) * sum(L_in{k,1}>tau_crplus(1));
end
k=0;
for k=1:N
    L_bar_in(k)= 1/N_in(k) * (L_in{k,1}*(L_in{k,1}>tau_crplus
       (1))');
end
```

```
76
77 rho(isnan(rho))=0;
78 yCEGS_crplus= 1/N*(rho*p_in')
79 rCEGS_crplus= (rho*L_bar_in')/(rho*p_in')
```

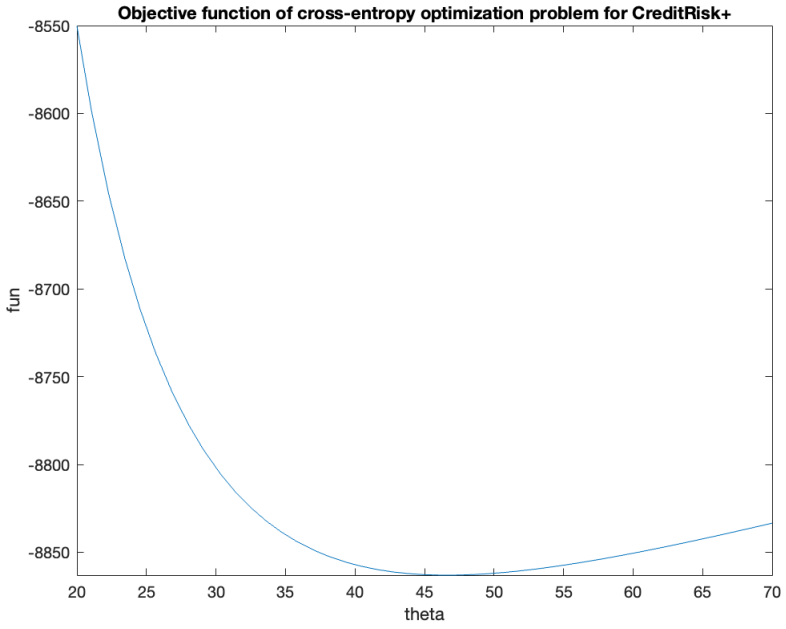


Figure 4.2: Objective function of cross-entropy optimization problem for CreditRisk+ model

4.3.2 Gaussian copula model

Also for this model, we follow the same structure adopted in the code of the CreditRisk+ simulation. We fix to $M = 10000$ the number of replication required to have good estimates of the importance sampling parameters. We vectorize properly the simulation rearranging the conditional probability of default 2.28 in a matrix form, that figure in the code as `p_gauss_matrix`. Referring to the section 3.2.2, we set the value of D', noticing that also in this case one optimization variable is involved. This results in the simplification of the following section. In the second section of the code we perform the

optimization of importance sampling parameters, that instead of using the genetic algorithm as in CrediRisk+, we can use the function `fminsearch`. Figure 4.3 illustrate the objective function of the cross-entropy optimization problem. In the last section, we simulate the default of obligors that constitute our portfolio, in the same manner, done in CreditRisk+. We put more attention to that we introduced in the row $46th$ the control on the value of probability, in order to avoid numerical problems in the computation of likelihood ratio.

```
1  %% CEGS implementatin for Gaussian copula model
2  clc
3  clear
4  close
5  j= 1000;    % obligors
6  d= 10;      % 10 factor model
7  p=0.01*(1+sin(16*pi/j*[1:j]'));
8  c= ceil(5/j*[1:j]').^2;
9  M=10000;
10
11 a = random('Uniform',0,1/sqrt(10),j,d);  %rows are obligors
       and columns are systematic factor
12 b = sqrt(ones(j,1)-sum(a.^2,2));
13 psi = randn(d,M);
14 p_gauss_matrix= cdf("Normal",(a*psi + icdf("Normal",p,0,1).*
       ones(j,M)).*(b.^-1),0,1);
15
16 sigma=1;
17 tau_gaus=[250 950 2000];
18
19 %% Cross entrpy CE method to compute v importance sampling
       paramters
20 obj_part1 = 1- cdf("Normal", (tau_gaus(1)-c'*p_gauss_matrix)
       ./sqrt((c.^2)'*(p_gauss_matrix-p_gauss_matrix.^2)),0,1);
21 fun_gauss = @(theta) -obj_part1*(log(1/(sigma*sqrt(2*pi))*exp
       (-0.5*(1/sigma*(psi-(theta*(a'*c).*ones(d,M)))).^2))'*ones
       (d,1));
22
23 fplot(fun_gauss) % plot of objective function
24 title('Objective function of cross-entropy optimization
       problem for Gaussian copula model')
25 xlabel('theta')
26 ylabel('fun gauss')
27 options = optimset('TolX',1e-8,'TolFun',1e-8);
```

```
28
29 theta_gauss_opt=fminsearch(fun_gauss,0.01,options)
30 v_gauss=theta_gauss_opt*(a'*c);
31
32 %% CEGS simulation
33 N= 10000;
34 psi= random("Normal",v_gauss.*ones(d,N),ones(d,N),d,N);
35 f_u=normpdf(psi);
36 f_v=normpdf(psi,v_gauss.*ones(d,N),ones(d,N));
37 rho= prod(f_u ./ f_v,1);
38 p_gauss_matrix= cdf("Normal",(a*psi + icdf("Normal",p,0,1).*
      ones(j,M)).*(b.^-1),0,1);
39 N_in= round(min(1./(mean(p_gauss_matrix,1)),j*ones(1,N)));
40 L_in={};
41 for  i=1:length(N_in)
42     L_in(i,:)={zeros(1,N_in(i))};
43 end
44
45 % to avoid numerial problems
46 p_gauss_matrix(p_gauss_matrix<1e-16)=2e-9;
47
48 for k=1:N
49     for i =1:j
50         lambda=0;
51         cont=false;
52         while cont==false
53             U=rand();
54             lambda= lambda + round(log(1-U)./log(1-
                  p_gauss_matrix(i,k)));
55             if lambda>N_in(k)
56                 cont=true;
57             else
58                 if i<=length(L_in{k,1})
59                     L_in{k,1}(i)= L_in{k,1}(i)+c(i);
60                 else
61                     break
62                 end
63             end
64         end
65     end
66 end
67
68 k=0;
69 for k=1:N
70     p_in(k)= 1/N_in(k) * sum(L_in{k,1}>tau_gaus(1));
```

```
71 end
72 k=0;
73 for k=1:N
74     L_bar_in(k)= 1/N_in(k) * (L_in{k,1}*(L_in{k,1}>tau_gaus
       (1))');
75 end
76
77 rho(isnan(rho))=0; %to avoid numerical problem
78 yCEGS_gauss= 1/N*(rho*p_in')
79 rCEGS_gauss= (rho*L_bar_in')/(rho*p_in')
```

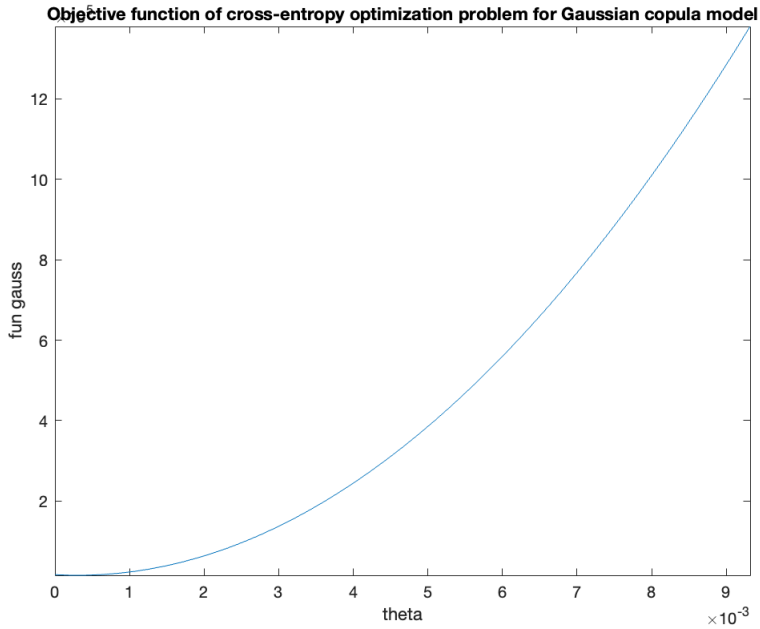


Figure 4.3: Objective function of cross-entropy optimization problem for Gaussian copula model

4.3.3 t-copula model

In this last section, we present the remaining model: the t-copula model. Since the implementation of the code follows the same steps as before, we can skip to the main focal point. Recall that this model differs from the other since the D' is equal to $D-1$, therefore we can combine $D-1$ identical factors into one optimization variable θ and also another

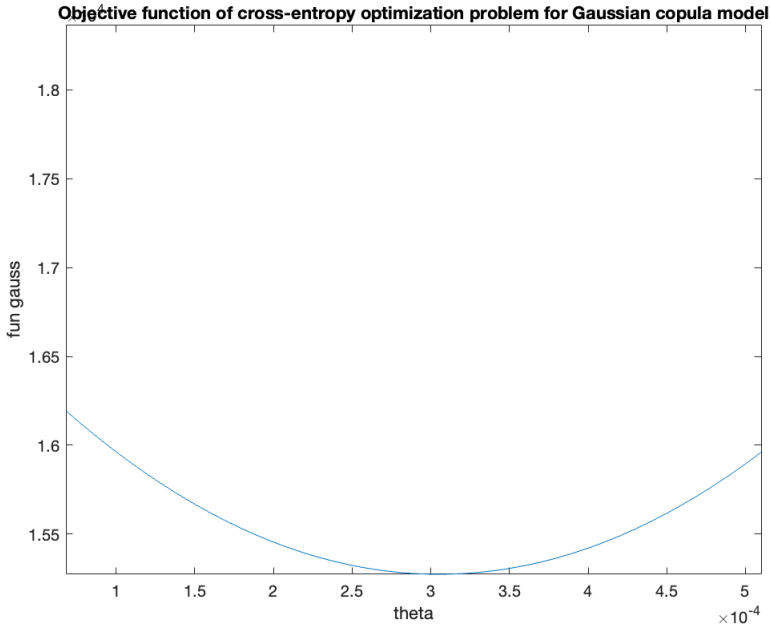


Figure 4.4: Detail on convexity of objective function fro Gaussian copula model

variable v_D constitutes the second optimization variable. At first glance, the problem becomes more complicated however, we notice that the same variables have no interaction. By exploiting this property, we can divide the problem into more easy subproblems, improving performance. Instead of using a more complex solver as the *genetic algorithm*, thanks to this decomposition we can use the function `fminunc` that uses by default the *quasi-newton algorithm*. As usual, we set $M = 1000$ as the number of replications to search the importance sampling parameters. Also, in this case, we recast the maximization problem 3.27 into minimization one. Firstly, we report the full objective surface (Fig. 4.5) in which we indicate the global minimum. Then, we apply the division into subproblems. In Figures 4.6 and 4.7 are plotted respectively the cross-entropy objective function for importance sampling parameters related to θ and the objective function of parameter v_D. In the last section, where the simulation equipped with geometric shortcut starts, we adopted the same structure used in the Gaussian copula model implementation. To avoid numerical problems, at 71-th and 102-th, we introduce a control function on the values of new conditional default probabilities.

```
%% CEGS implementatin for t-copula model
clc
clear
close

j=1200; % obligors
d=5;    % 5 factor model;
p= random('Uniform',0,0.02,j,1);
c= ceil(20/j*[1:j]').^2;
rng('default')
a = [random("Uniform",0,0.5,200,2) random("Uniform
   ",0,0.1,200,1) zeros(200,2);
     random("Uniform",0,0.5,200,1) random("Uniform
        ",0,0.1,200,1) random("Uniform",0,0.5,200,1) zeros
        (200,2);
     random("Uniform",0,0.4,200,1) zeros(200,1) random("
        Uniform",0,0.3,200,1) random("Uniform",0,0.1,200,1)
        zeros(200,1);
     random("Uniform",0,0.4,200,1) zeros(200,1) random("
        Uniform",0,0.1,200,1) random("Uniform",0,0.3,200,1)
        zeros(200,1);
     random("Uniform",0,0.5,200,1) zeros(200,2) random("
        Uniform",0,0.4,200,1) random("Uniform",0,0.3,200,1);
     random("Uniform",0,0.5,200,1) zeros(200,2) random("
        Uniform",0,0.3,200,1) random("Uniform",0,0.4,200,1)];
a_t = a(:,1:d-1);
b = sqrt(ones(j,1)-sum(a_t.^2,2));
v=5;
M=10000;
tau_t=[4500 16500 34000];
psi_t = randn(d-1,M);
psi_D = random('Chisquare',v,1,M);
psi   = [psi_t;psi_D];   %general matrix of psi
p_t_matrix = cdf('Normal',(a_t*psi_t - sqrt(psi_D/v).*icdf('T
   ',1-p,v)).*(1./b),0,1);

%% Cross entrpy CE method to compute v importance sampling
   paramters
% calculate the first part of objective function
obj_part1 = 1- cdf("Normal", (tau_t(1)-c'*p_t_matrix)./sqrt((
   c.^2)'*(p_t_matrix-p_t_matrix.^2)),0,1);

% optimization for theta
```

```
fun_t= @(theta) -obj_part1*(log(normpdf(psi_t,theta*(a_t'*c)
   .*ones(d-1,M),1))'*ones(d-1,1));
[theta_t_opt, fval_fun_t]= fminunc(fun_t,0)
fplot(fun_t,[0 0.0001])

% optimization for vD
fun_t= @(vD) -obj_part1*(log(chi2pdf(psi_D,vD*ones(1,M)))');
[vD_opt, fval_fun_t]= fminunc(fun_t,0.0001)
fplot(fun_t,[0 7])

%% plot of full objective function
fun = @(theta,vD) -obj_part1*(log(normpdf(psi_t,theta*(a_t'*c
   ).*ones(d-1,M),1))'*ones(d-1,1)) -obj_part1*(log(chi2pdf(
   psi_D,vD*ones(1,M)))');
fsurf(fun,[0 0.0001 0 7])
xlabel('theta')
ylabel('vD')
hold on
fsurf(theta_t_opt,vD_opt,fun(theta_t_opt,vD_opt),'Marker','o'
   ,'MarkerFaceColor','r','MarkerSize',15) % minimum value of
    obj function
hold off

%% CEGS simulation
v=theta_t_opt*(a_t'*c);
vD_opt= round(vD_opt);
N= 10000;

psi_t= random("Normal",v.*ones(d-1,N),1);
psi_D= random("Chisquare",vD_opt*ones(1,N));

% compute f_u and f_v
f_u =[normpdf(psi_t); chi2pdf(psi_D,5)]; % 5 are the initial
   degree of freedom
f_v =[normpdf(psi_t,v.*ones(d-1,N),1); chi2pdf(psi_D,vD_opt)
   ];
rho= prod(f_u ./ f_v,1);

p_t_matrix = cdf('Normal',(a_t*psi_t - sqrt(psi_D/vD_opt).*
   icdf('T',1-p,vD_opt)).*(1./b),0,1); %it is a matrix where
   the columns contain k=1...N replication infact #col is
   =10000

N_in= round(min(1./(mean(p_t_matrix,1)),j*ones(1,N)));
```

```
66  L_in={}; % construct a loss vector of different size for each
        inner replication
67  for  i=1:length(N_in)
68      L_in(i,:)={zeros(1,N_in(i))};
69  end
70
71  p_t_matrix(p_t_matrix<1e-16)=2e-9; % to avoid numerial
        problems
72
73  for k=1:N
74      for i =1:j
75          lambda=0;
76          cont=false;
77          while cont==false
78              U=rand();
79              lambda= lambda + ceil(log(1-U)./log(1-p_t_matrix(
                    i,k)));
80              if lambda>N_in(k)
81                  cont=true;
82              else
83                  if i<=length(L_in{k,1})
84                      L_in{k,1}(i)= L_in{k,1}(i)+c(i);
85                  else
86                      break
87                  end
88              end
89          end
90      end
91  end
92
93  k=0;
94  for k=1:N
95      p_in(k)= 1/N_in(k) * sum(L_in{k,1}>tau_t(1));
96  end
97  k=0;
98  for k=1:N
99      L_bar_in(k)= 1/N_in(k) * (L_in{k,1}*(L_in{k,1}>tau_t(1))
            ');
100 end
101
102 rho(isnan(rho))=0; %to avoid numerical problems
103 yCEGS_tcop= 1/N*(rho*p_in')
104 rCEGS_tcop= (rho*L_bar_in')/(rho*p_in')
```

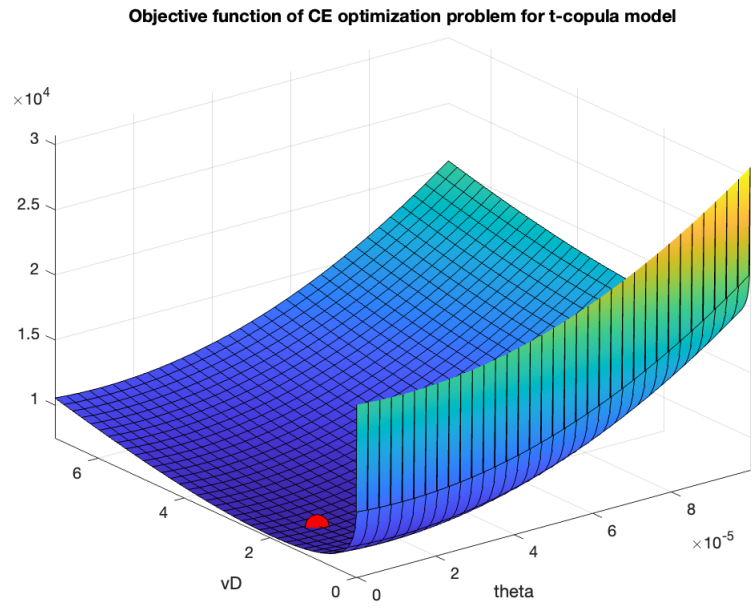


Figure 4.5: Objective function of cross-entropy optimization problem for t-copula model

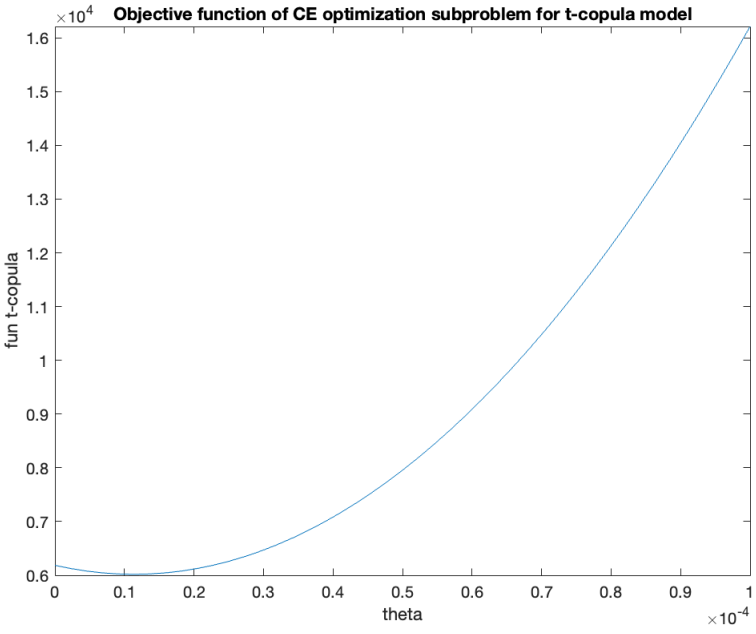


Figure 4.6: t-copula model: objective function for theta optimization subproblem

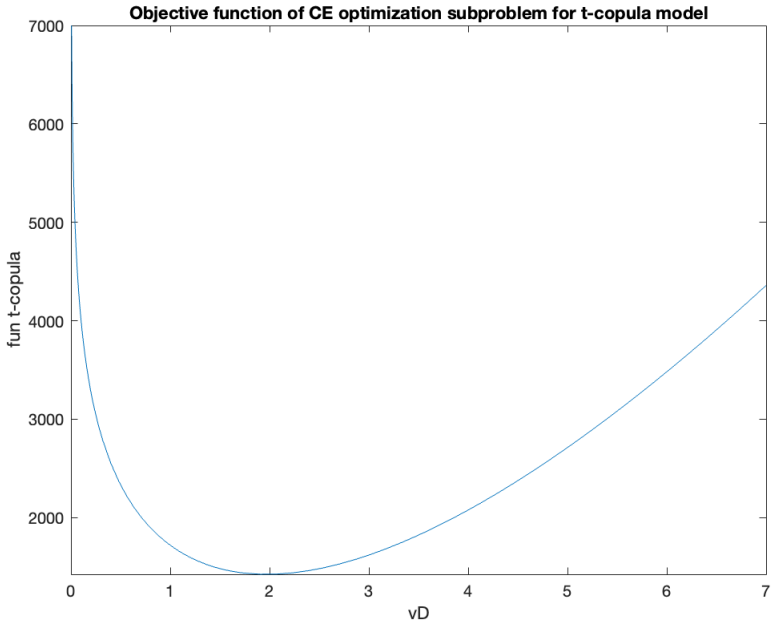


Figure 4.7: t-copula model: objective function for v_D optimization subproblem

4.4 Experimental result

All implementations discussed end up with the two main risk metrics: tail loss probability and conditional excess. All these metrics derive from the simulated loss distribution. We recall that our models are set on different portfolios with different parameters as discussed in Section 4.1 and as done in Basoglu et al. [2018]. This choice is taken in order to evaluate the new method based on Cross entropy and Geometric shortcut on different portfolios and models: it's a kind of test of the robustness of the method. For readability reasons, it is useful to remember that the CreditRisk+ model refers to the parameters described in Section 4.1.1; for the Gaussian copula model notice the Section 4.1.2 and finally for the t-copula model, the Section 4.1.3. We propose one first analysis that compares for every single model, the Naive versus the Cross-entropy Geometric shortcut simulation. This analysis is based on the computation of VaR (Value at Risk), a common risk metric introduced by JP Morgan investment banking, and recognized as a valid risk metric in the second pillar of Basel II. It is defined as the maximum dollar amount expected to

be lost over a given time horizon, at a pre-defined confidence level. For our case, it corresponds to the quantile of simulated loss distribution at 95% level of confidence. In Figure 4.8 there are the simulated loss distributions for all three models, and we mark with a vertical red line the value of VaR. Notice that the distributions have a long and thin tail, which means that the naive Monte Carlo method, especially when dealing with a value of probability near 0, does not apport significant information about the tail distribution. This is a collateral effect of the *rare-events* simulation. The new method implemented in this work helps us to increase the quality of the simulation in the tail of the distribution. Figure 4.9 show the distributions of losses for the three models simulated with the cross-entropy geometric shortcut method. From a qualitative point of view, it's evident the difference between the two distributions of the CreditRisk+ model. With the importance sampling based on cross-entropy, the distribution assumes a completely different shape, providing a relevant number of defaults. This justifies the fact that the distribution mode is shifted to the left, near the VaR. For the Gaussian copula model, we can draw the same conclusion, appreciating the improvements on the tail side simulation. Further clarifications can be made on the t-copula model: although we reduced the variability in the tail, the effect of importance sampling is less evident. By complexity of this model, arise the difficulties in the performance. Despite this, the frequency observed in the tail for CEGS simulation increased. To conduct a more strong quantitative analysis, VaR is not the most recommended risk measure. We can only observe that the VaR in the CEGS simulation is always greater than the Naive VaR, meaning that the method is working properly and there is an increase in the probability of observing a higher loss. The main problem is that it is strongly dependent on the obligor's exposure. In our case, we used VaR only to compare a given model across the two simulations, not across models. We aim to get an overall evaluation characterized by model independence property, to quantify the improvements of the new method discussed in this work.

The target risk metrics, tail loss probability, and conditional excess computed for every model, end to themselves if we limit to get their value. We have to adopt a measure, strictly related to this quantity, that compares the efficiency of Cross-entropy with Geometric shortcut simulation versus the Naive one. As in Basoglu et al. [2018], we use the half

length of the 95% confidence interval, to evaluate the goodness of estimates of tail loss probability and conditional excess. The total sample size $N = 10000$ for each method, help us to use, thanks to the central limit theorem, the common structure of confidence interval. In Tables 4.1 and 4.2 we present the results.

Table 4.1 reports the results for the tail loss probabilities. For each threshold value in the second column there are the estimates of the tail loss probabilities $P(L > \tau)$ with Naive and CEGS simulations, and the related confidence intervals [3]. For every model increasing the value of the threshold, decrease obviously the tail loss probability, and consequently gets worse its estimate. It is more evident looking at the half lengths of the confidence interval of the Naive estimates: higher the threshold τ, longer the half length of the confidence interval. Given a certain threshold value, if we compare the two estimates of the risk measure they have the same order of magnitude, indicating a correct working of the two types of simulation, however, there is a significant difference in the confidence interval. The estimate obtained with CEGS simulation always has a shorter semi-length of the confidence interval. The more evident case, for every model, appears with the last threshold value. E.g. for the Gaussian copula model, with threshold $\tau = 2000$, the semi-confidence interval for the naive simulation is 22.33 approximately 20 times longer than the one computed with CEGS simulation, equal to 1.00. Focusing only on the last two columns we can notice that for the naive simulation, there is high variability in the confidence intervals, for every model. The same variability is killed by the cross-entropy method that limits the semi-lengths of confidence intervals. This analysis is still valid for the second Table 4.2 that presents the estimates and the confidence intervals for the conditional excess $E[L|L > \tau]$. The table presents the same structure as before, for easy reading. Additional analysis can be made on the computational time required for the simulations. In the Table 4.3 are given all times for all simulations run. It is done thanks to the `tic-toc` MATLAB command. Notice, as expected, that changing the threshold value does not affect the computational time. Instead, the complexity of the model and the type of simulation significantly affect the time elapsed. E.g the t-copula model shows

[3] In the table the text *confint* stands for the confidence interval.

Model	Threshold τ	NV estimate	CEGS estimate	NV confint	CEGS confint
CreditRisk+	8.0	0.1068	0.1014	2.2	0.40
	13.8	0.0512	0.02	7.3	0.66
	19.7	0.0286	0.001	22.62	0.97
Gaussinan copula	250.0	0.0533	0.10	1.77	0.59
	950.0	0.0015	0.0863	6.27	0.84
	2000.0	0.0002	0.0009	22.33	1.00
t- copula	4,500.0	0.1027	0.108	1.87	0.56
	16,500.0	0.0209	0.0487	6.10	0.79
	34,000.0	0.0042	0.005	19.30	0.90

Table 4.1: Experimental results for tail loss probabilities $P(L > \tau)$, with estimates and half lengths of confidence intervals at 95% level.

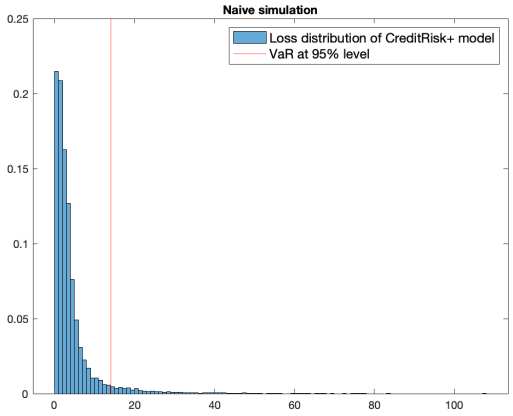
Model	Threshold τ	NV estimate	CEGS estimate	NV confint	CEGS confint
CreditRisk+	8.0	17.820	8.8419	0.46	0.07
	13.8	25.318	27.54	0.9	0.05
	19.7	32.495	35.21	2.69	0.06
Gaussinan copula	250.0	428.1	508.45	1.21	0.33
	950.0	1385.8	1370.3	2.00	0.21
	2000.0	2477.0	2478.45	4.04	0.15
t- copula	4,500.0	12,190	9,500	1.29	0.33
	16,500.0	27,608	24,218	2.04	0.21
	34,000.0	45,601	40,365	3.99	0.14

Table 4.2: Experimental results for conditional excess $E[L|L > \tau]$, with estimates and half lengths of confidence intervals at 95% level.

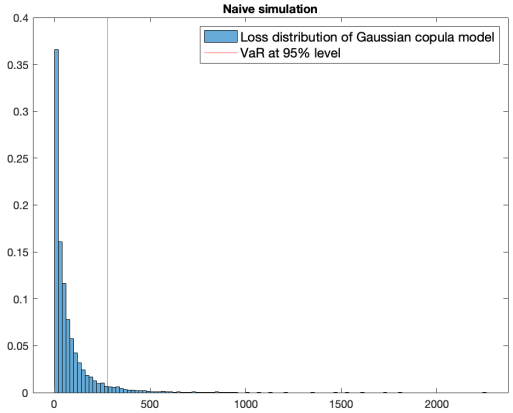
as the most complex model in computational time and estimates of risk measure as well. The CEGS simulation should also improve the computational time with respect to the naive simulation: it is out of our results. This is due to the choice of MATLAB software, especially in the vectorization of the code. The main simulation part of each model, cannot be properly vectorized and this reflects in slower simulations.

Model	Threshold τ	NV	CEGS
CreditRisk+	8.0	8.24	12
	13.8	8.24	12
	19.7	8.24	12
Gaussinan copula	250.0	11	13.4
	950.0	11	13.4
	2000.0	11	13.4
t- copula	4,500.0	12.3	15.7
	16,500.0	12.3	15.7
	34,000.0	12.3	15.7

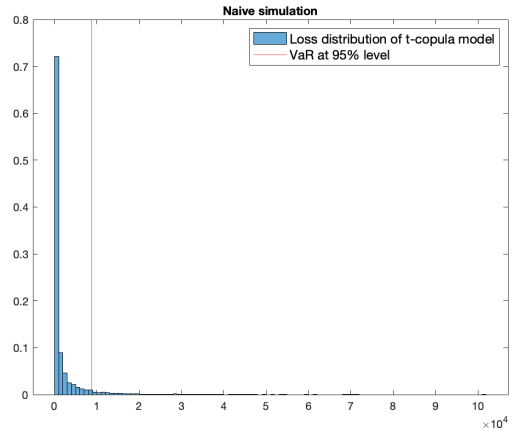
Table 4.3: Computational times of the simulations for the three models



(a) Naive simulation for CreditRisk+ model

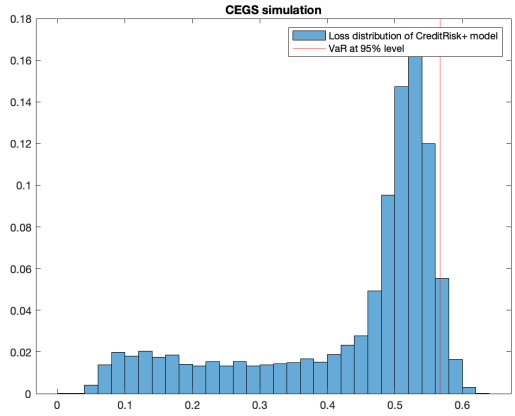


(b) Naive simulation for Gaussian copula model

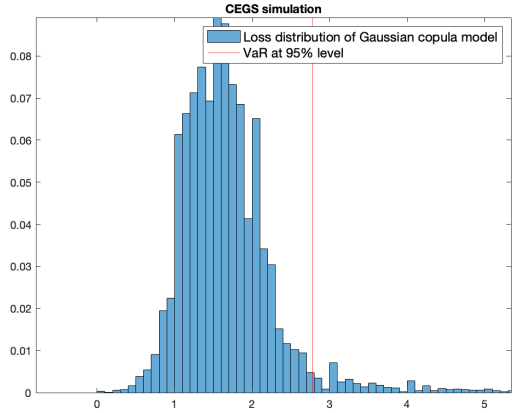


(c) Naive simulation for t-copula model

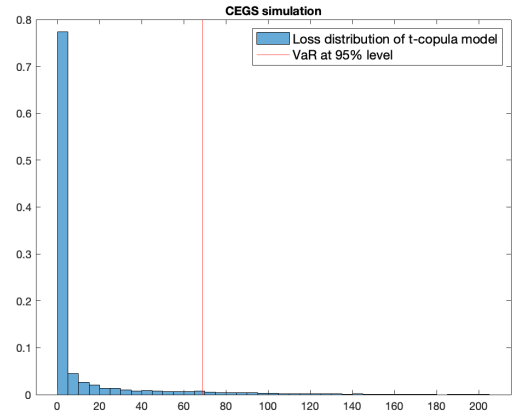
Figure 4.8: Distributions of losses and VaRs at 95% for the Naive Monte Carlo simulations



(a) CEGS simulation for CreditRisk+ model



(b) CEGS simulation for Gaussian copula model



(c) CEGS simulation for t-copula model

Figure 4.9: Distributions of losses and VaRs at 95% for the CEGS Monte Carlo simulations

Chapter 5

Conclusions and future works

This work aimed to implement a new efficient method that improves the estimates of the tail loss probabilities and conditional excess for a simulated portfolio's loss distribution. We use three different models with the related three portfolios of obligors. The new approach aspired to overcome the main difficulties of the Naive Monte Carlo simulation, improving the quality of the estimate and reducing the computational time. According to the results obtained in Section 4.4, we satisfied the target requirements in all models analyzed: the Monte Carlo simulation equipped with the Cross-Entropy and the Geometric Shortcut methods always give better estimates of the risk measures. This is valid as long as we use the literature parameters taken into account in the model-building phase. The exposure of each obligor and the marginal default probability of default, if they are changed, can give the opposite results, thus different conclusions. There aren't improvements in the computational time. The Naive Monte Carlo simulation, in our analysis, is a bit faster than the CEGS simulation, and this is due to the limit of software in the cycle iteration. From this point, potential future works can be developed. If we implement the simulation in python, we can deal easily with `for` loops, without vectorizing the code. Furthermore, this work can be improved by implementing another variance reduction method: stratified sampling. The final product formalized by Basoglu et al. [2018] is called stratified cross-entropy geometric shortcut simulation (STCEGS), which includes

stratified sampling across obligors. This sampling regards principally the systematic factors $\mathbf{\Psi}$, and consists in identifying equiprobable subset from the multivariate distribution of systematic factors. In all models considered, we use the literature values of parameters, thus defined *a priori*. A good challenge is to implement the same procedures discussed in this work with parameters obtained from a previous statistical analysis. Given a dataset of obligors and default, we can make inferences on parameters obtained from a multivariate regression. Nowadays many machine-learning techniques are used in credit risk assessment, especially for the computation of *scores* on obligors. This evaluation can be adapted to our model looking for more realistic exposures c_j of obligors and reliable weight of systematic factor a_{jd}.

Bibliography

P. Embrechts A. J. McNeil, R. Frey. *Quantitative Risk Management*. Princeton universty press, 2005.

Ismail Basoglu, Wolfgang Hormann, and Halis Sak. Efficent simualtion for a bernuolli mixture model of portfolio credit risk. *Annals of Operations Research*, 2018.

CreditSuisse. Organizational guidelines and regulations, 2023. URL https://www.credit-suisse.com/media/assets/about-us/docs/our-company/our-governance/organizational-guidelines-and-regulations-en.pdf.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005. doi: 10.1007/s10479-005-5724-z. URL https://doi.org/10.1007/s10479-005-5724-z.

Kenneth R. French. Kenneth r. french - data library, 2023. URL https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

Paul Glasserman. Measuring marginal risk contributions in credit portfolios. *FDIC Center For Financial Research*, 51, 01 2005. doi: https://dx.doi.org/10.2139/ssrn.681227.

Paul Glasserman and Jingyi Li. Importance sampling for portfolio credit risk. *Management Science*, 51:1643–1656, 02 2004. doi: 10.1287/mnsc.1050.0415.

MathWorks. Matlab documentation center, 2022. URL http://www.mathworks.it/it/help/matlab/.

Halis Sak and Wolfgang Hörmann. Fast simulations in credit risk. *Quantitative Finance*, 12(10):1557–1569, 2012. doi: 10.1080/14697688.2011.564199. URL https://doi.org/10.1080/14697688.2011.564199.