EFFICIENT SIMULATIONS IN FINANCE

by

Halis Sak

B.Sc., in Mechanical Engineering, Middle East Technical University, 2000

M.Sc., in Industrial Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2008

ACKNOWLEDGEMENTS

I am grateful to my advisor, Assoc. Prof. Wolfgang Hörmann for introducing me to the field of simulation. Without his guidance and patience, I would have probably being lost in this study. I am also grateful to the members of examining committee, Prof. Süleyman Özekici and Prof. Refik Güllü, for their helpful comments during my study. It was a great pleasure for me to do my research under supervision of these three researchers.

I had been working as a teaching assistant at İstanbul Kültür University during most of the time of my thesis study. I want to thank Prof. Tülin Aktin for her support as an employer during this period of time.

I have been partially supported from the scholarship program BİDEP 2211 of The Scientific and Technological Research Council of Turkey (TÜBİTAK) during my research. Many thanks to them for giving the feeling of financially secure for just doing what we meant to do.

I have been working as a research assistant in a project for four months in the department of Statistics and Mathematics, Vienna University of Economics and Business Administration. I am grateful to Prof. Josef Leydold, Prof. Kurt Hornik, Karin Haupt, Stefan Theußl and others for their support and hospitality during my time in Vienna.

I want to thank my brothers, parents and to all who really care about me. Finally, I want to specifically thank Haşim Sak and Sibel Tombaz for their support and love.

ABSTRACT

EFFICIENT SIMULATIONS IN FINANCE

Measuring the risk of a credit portfolio is a challenge for financial institutions because of the regulations brought by the Basel Committee. In recent years lots of models and state-of-the-art methods, which utilize Monte Carlo simulation, were proposed to solve this problem. In most of the models factors are used to account for the correlations between obligors. We concentrate on the the normal copula model, which assumes multivariate normality of the factors. Computation of value at risk (VaR) and expected shortfall (ES) for realistic credit portfolio models is subtle, since, (i) there is dependency throughout the portfolio; (ii) an efficient method is required to compute tail loss probabilities and conditional expectations at multiple points simultaneously. This is why Monte Carlo simulation must be improved by variance reduction techniques such as importance sampling (IS). Optimal IS probabilities are computed and compared with the "asymptotically optimal" probabilities for credit portfolios consisting of groups of independent obligors. Then, a new method is developed for simulating tail loss probabilities and conditional expectations for a standard credit risk portfolio. The new method is an integration of IS with inner replications using geometric shortcut for dependent obligors in a normal copula framework. Numerical results show that the new method is better than naive simulation for computing tail loss probabilities and conditional expectations at a single x and VaR value. Furthermore, it is clearly better than two-step IS in a single simulation to compute tail loss probabilities and conditional expectations at multiple x and VaR values. Then, the performance of outer IS strategies, which consider only shifting the mean of the systematic risk factors of realistic credit risk portfolios are evaluated. Finally, it is shown that compared to the standard t statistic a skewness-correction method of Peter Hall is a simple and more accurate alternative for constructing confidence intervals.

ÖZET

VERİMLİ FİNANSAL SİMÜLASYONLAR

Basel komitesinin düzenlemeleri finansal enstitüler için zor bir iş olan kredi portföy riskinin hesaplanmasını zorunlu kılar. Son yıllarda bir çok model ve Monte Carlo simülasyonunu kullanan metodlar geliştirilmiştir. Bu modellerin çoğunda yükümlüler arası korrelasyonu sağlamak için faktörler kullanılır. Biz çok-faktörlü normalliğin kabul edildiği normal kapula modeli üzerinde yoğunlaşırız. Gerçekci kredi potföy modelleri için riskteki değer (VaR) ve beklenen kuyruk kaybı (ES) hesaplanması kompleks bir iştir, çünkü; (i) portföy içinde bağımlılık vardır; (ii) kuyruk kayıp olasılıklarını ve şartlı beklentileri çoklu noktalarda aynı anda verebilecek verimli bir metoda gereksinim duyulur. Bu yüzden Monte Carlo simülasyonu önemli örnekleme (IS) gibi varyasyon azaltma teknikleri ile geliştirilmelidir. Bağımsız ve küçük guruplara bölünebilen yükümlülerden oluşan kredi portföyleri için en iyi IS olasılıkları hesaplanır ve bunlar "en iyi asimtotik" olasılıklarla karşılaştırılır. Daha sonra standart kredi portföyleri için kuyruk kayıp olasılıklarını ve şartlı beklentileri simüle edecek yeni bir metod geliştirilir. Yeni metod normal kopula kapsamındaki bağımlı yükümlüler için IS'in geometrik kısa yolu kullananan içsel replikasyonlarla birleşimidir. Numeriksel sonuçlar, yeni metodun tek bir x ve VaR değeri için standart simülasyondan daha iyi olduğunu ortaya koyar. Buna ek olarak, tek bir simülasyonda birden fazla x ve VaR değerleri için kuyruk kayıp olasılıklarının ve şartlı beklentilerinin hesaplanmasında iki-aşamalı IS'den açık bir şekilde daha iyidir. Daha sonra, gerçekçi kredi portföyleri üzerinde sadece sistematik risk faktörlerinin ortalamasını arttıran dışsal IS stratejileri incelenir. En sonunda, standart t istatistiğiyle karşılaştırıldığında Peter Hall'un yamukluk düzeltme metodunun güven aralığı oluşturulmasında kolay ve daha kesin bir alternatif olduğu gösterilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a_{jl}	Factor loading of jth obligor for risk factor l
A	First group of obligors
B	Second group of obligors
c	Loss resulting from default of an obligor or homogenous loss
c_A	Loss resulting from default of an obligor in group A
c_B	Loss resulting from default of an obligor in group B
c_j	Loss resulting from default of jth obligor
c^∞	$100(1-\alpha)\%$ quantile of L_d^∞
$Cov(Z_k, Z_l)$	Covariance between Z_k and Z_l
d	Number of systematic risk factors
g_j	Weight for obligor j
$E[T]$	Expectation of a random variable T
$E[T\|S]$	Expectation of a random variable T conditional on an event S
E_θ	Expectation under exponential twisting with parameter θ
$erfc(z)$	Complementary error function of z
$exp(z)$	Exponential function of z
I	Identity matrix
L	Total loss of portfolio
L_i	ith possible (having probability greater than zero) loss value in the portfolio loss distribution
L_d^∞	Infinite homogenous portfolio loss function
$L^{(k)}$	Total loss of portfolio for replication k
m	Number of obligors in portfolio
m_A	Number of obligors in group A
m_B	Number of obligors in group B
$M_2(x, \theta)$	Second moment of estimator for $P(L > x)$ for exponential twisting with parameter θ
n	Number of replications

n_{in}	Number of inner replications
$N_{\mu,\sigma}$	Normal density function with mean μ and variance σ
p	Default probability of an obligor or homogenous default probability
$\mathcal{P}$	Probability measure
p_A	Default probability of an obligor in group A
p_B	Default probability of an obligor in group B
p_j	Marginal default probability of jth obligor
p_{new}	IS default probability
$p_{new,A}$	IS default probability for obligors in group A
$p_{new,B}$	IS default probability for obligors in group B
$P(S)$	Probability of an event S
$\bar{p}_Z$	Average of the conditional default probabilities for obligors
$\mathcal{Q}$	Importance Sampling probability measure
r_s	Speed up ratio
s	Scaling factor to normalize weighted sum of factor loadings
$Var[T]$	Variance of a random variable T
w	Likelihood ratio
$w^{(k)}$	Likelihood ratio for replication k
X	Multivariate normal vector of latent variables
Y	Default indicator for an obligor (1 if default, 0 otherwise)
Y_j	Default indicator for jth obligor (1 if default, 0 otherwise)
Z	Multivariate standard normal random variable for systematic risk factors
Z_l	Standard normal random variable for systematic risk factor l
$z_{\delta/2}$	$(1-\delta/2)$ percentile of the standard normal distribution
$1-\alpha$	Probability level at which confidence interval is given
ϵ_j	Idiosyncratic risk factor for obligor j
θ	Exponential twisting parameter
μ	Mean shift vector in IS
ϕ	Density function for $\mathcal{P}$

ψ	Density function for $\mathcal{Q}$
$\psi_L(\theta)$	Cumulant generating function of L given θ
Φ	Standard normal cumulative distribution function
Ψ	Weighted sum of factor loadings
CI	Confidence Interval
ES	Expected Shortfall
ET	Exponential twisting
ETP	Exponential twisting probability
GSL	GNU Scientific Library
HA	Homogenous approximation
hl	Half length of confidence interval
IS	Importance Sampling
max	Maximize
NA	Normal approximation
O	Optimal IS
OP	Optimal IS probability
prob.	Probability
skw	Skewness
sd	Standard deviation
sup	Supremum (least upper bound)
TBA	Tail bound approximation
var	Variance
VaR	Value at Risk
VaR_α	Value at Risk at $100(1-\alpha)$ percent confidence level
VR	Variance Ratio

1. INTRODUCTION

Financial institutions are subject to a wide range of risks. [6] classifies them as market risk, credit risk, liquidity risk, operational risk and systematic risk. This thesis study considers only credit risk, which can be defined as risks associated with default of obligors to make payments or changes in their credit quality.

Measuring the credit portfolio risk is a challenge for financial institutions because of the regulations brought by Basel Committee [3]. In recent years lots of models (see [4]) and state-of-the-art methods, which utilize Monte Carlo simulation, (see [11]) were proposed to solve this problem. The proposed models are with the exception of CreditPortfolio View [27] factor based models. They use factors to account for the correlations in the defaults of obligors. This ease the calibration of the models and decrease computational effort for correlated losses. However, [20] report that multi-factor models should be adjusted to achieve Basel II-consistent results and they show how to do that.

We concentrate on the the normal copula model of [19], a Merton [29] type model, to capture the dependence across obligors. Underlying risk factors are assumed to be normal in this model. However, we are aware of the risks of using this model because of weak tail dependence (see [8]).

Value at Risk (VaR) has been quite popular as a risk measure to be used in credit risk models because of being conceptually simple and easy-to-compute. It is simply the quantile of portfolio loss distribution. [2] defines VaR at $100(1-\alpha)$ percent confidence level ($\mathrm{VaR}_\alpha(X)$) as

$$\mathrm{VaR}_\alpha(X) = \sup\{x | P[X \geq x] > \alpha\}$$

where $\sup\{x|A\}$ is the upper limit of x given event A.

Nevertheless, there are some shortcomings of VaR (see [1] and [2]). To summarize, it ignores losses beyond VaR and is not coherent. [1] proposes Expected Shortfall (ES) to solve the problems of VaR. ES is simply the expected amount of losses that are larger than the VaR level. Thus, ES is defined as:

$$\mathrm{ES}_\alpha(X) = E[X|X \geq \mathrm{VaR}_\alpha(X)]$$

for the $100(1-\alpha)$ percent confidence level.

Computation of VaR and ES for realistic portfolio models is subtle, since, (i) there is dependency throughout the portfolio; (ii) an efficient method is required to compute tail loss probabilities and conditional expectations at multiple points simultaneously. This is why Monte Carlo simulation is a widely used tool here. But, simulation is not an easy option either. We need to apply variance reduction techniques such as IS, control variate, etc. (see [11], Chapter 4) for rare-event simulations, simulating tail loss probabilities and conditional expectations.

Variance reduction techniques for simulating tail loss probabilities and conditional expectations in the normal copula framework are "hot topics". Previous studies in importance sampling (IS), in the the normal copula model, can be summarized as:

(a) outer IS: applying IS to common factors (see [7, 25]);
(b) inner IS: applying IS on the resulting conditional default probabilities (see [28]);
(c) two-step IS: applying outer IS first, then inner IS (see [12, 13, 18]).

These papers apply different but similar methodologies to slightly different problems. Note that all these works were finished in the last 4 years.

[13] and [28] apply an importance sampling (IS) technique which utilizes exponentially twisted Bernoulli mass functions discussed in [31] for computing $P(L > x)$. Furthermore, [18] applies IS approach of [13], to the integrated market and credit portfolio model described in [17].

[13] and [28] not only apply exponential twisting, but also show that it is "asymptotically optimal"[1] for loss exceedance probabilities. However, [15] shows that asymptotically optimality of IS based on exponential twisting is true only in one point. Moreover, [7] report that for practical purposes, asymptotically optimal IS methods may fail. Even, they may perform worse than naive simulation for a typical medium-sized portfolio. And, they propose an IS approach that is based on adaptive stochastic approximation of Robbins-Monro.

It is well known that whenever we have a large skewness in a simulation, standard method of using t statistc for the confidence interval construction for the mean does not give robust results (see [11]). However, there are studies on how to decrease the effect of skewness of the population when doing tests on the mean such as confidence interval construction and hypothesis testing. [23] propose a transformation (Johnson's transformation) on the t variable to correct for the bias on the mean, but [21] reports that it fails to correct for skewness. [21] propose another transformation (Hall's transformation) that has desirable characteristics of monotonicity and invertibility to correct for both bias and skewness from the distribution of t statistic. Furthermore, [34] compares the performance of Johnson's and Hall's transformations and their bootstrap versions by looking at the coverage accuracy of one-sided confidence intervals.

The organization of the thesis is as follows.

In Chapter 2 we first describe the widely used dependency model across obligors in a credit portfolio called here the normal copula model (see Section 2.1). In that section we also introduce the notation, which will be used throughout the thesis. In Section 2.2 we talk about simulation and one of the classical variance reduction techniques, IS. Finally, we describe the IS approach of [13] for the normal copula model.

In Chapter 3 we show how to compute the optimal importance sampling probabilities for credit portfolios consisting of groups of independent obligors both for tail

[1] asymptotically optimality refers to the case where the second moment decreases at the fastest possible rate among all unbiased estimators as the event becomes rare (see [13]).

loss probability and expected shortfall simulation. In Section 3.1 we show that we can apply exponential twisting directly on the binomial distribution to simulate tail loss probabilities. In Section 3.2 we compare the optimal IS probabilities with the "asymptotically optimal" ones for groups of obligors. In Section 3.3 we show how our methodology which is used throughout the chapter can be used to compute optimal probabilities for two small financial examples.

In Chapter 4 we propose a new efficient simulation method for computing tail loss probabilities and conditional expectations in the normal copula framework. We replace inner IS by inner replications implemented by a geometric shortcut in the two-step IS of [13]. Section 4.1 analyzes the geometric shortcut for independent obligors. In section 4.2 we consider dependent obligors and introduce the geometric shortcut for inner replications. In Section 4.3 we add outer-IS to the inner replications of the geometric shortcut for a portfolio having dependent obligors. While Sections 4.1-4.3 concentrate on the efficient simulations for computing tail loss probabilities, Section 4.4 applies our new method to the simulation of conditional expectations. We summarize our numerical results in Section 4.5. Finally, we conclude in Section 4.6.

In Chapter 5 we evaluate outer IS strategies, which consider only shifting the mean of the systematic risk factors in the numerical examples of Chapter 4. In Section 5.1 we describe the tail bound approximation used in [13] and the normal approximation used in Chapter 4, which are then used to find the mode of the zero-variance IS distribution. In Section 5.2 we describe the homogenous portfolio approximation of [25]. In Section 5.3 we assess the performance of these three ways to calculate the mean shifts for the numerical examples of Chapter 4.

In Chapter 6 we evaluate the performance of Hall's transformation in correcting the confidence intervals for financial simulations that include IS. In Section 6.1 we give the details of Hall's transformation. In Section 6.2.1 and 6.2.2 we describe a portfolio risk and an option pricing example and compare the coverage accuracy of one-sided confidence intervals using ordinary t statistic and Hall's transformation. For both of the problems we have analytical solutions. In Section 6.3 we compare the performance

of Hall's condence intervals and standard t intervals for the numerical examples of Chapter 4.

We conclude in Chapter 7.

2. THE CREDIT RISK MODEL

2.1. The Normal Copula Model

We are interested in the the normal copula model of CreditMetrics (see [19]) for the dependence structure across obligors. We first of all give the notation used throughout the thesis which is similar to [13].

m: number of obligors in portfolio

Y_j: default indicator for jth obligor (equal to 1 if default occurs, 0 otherwise)

c_j: loss resulting from the default of jth obligor

p_j: marginal default probability of jth obligor

$L = \sum_{j=1}^{m} c_j Y_j$: total loss of portfolio

n: number of replications in a simulation

Our aim is to assess the distribution of tail loss probability and ES over a fixed horizon. Values of c_j's are known and constant over the fixed horizon. Furthermore, we assume that the marginal default probabilities (p_j's) are known.

The normal copula model introduces a multivariate normal vector $(X_1, ..., X_m)$ of latent variables to obtain dependence across obligors. Relationship between the default indicators and the latent variables are represented by

$$Y_j = \mathbf{1}_{\{X_j > x_j\}}, \quad j = 1, ..., m,$$

where X_j has standard normal distribution and $x_j = \Phi^{-1}(1 - p_j)$, with Φ^{-1} inverse of the standard normal cumulative distribution function. Obviously, the threshold value of x_j is chosen such that $P(Y_j = 1) = p_j$.

The correlations among X_j are modeled as

$$X_j = b_j \epsilon_j + a_{j1} Z_1 + ... + a_{jd} Z_d, \quad j = 1, ..., m, \tag{2.1}$$

where ϵ_j and $Z_1, ..., Z_d$ are independent standard normal random variables with $b_j^2 + a_{j1}^2 + ... + a_{jd}^2 = 1$. While, $(Z_1, ..., Z_d)$ are systematic risk factors affecting all of the obligors, ϵ_j is the idiosyncratic risk factor affecting only obligor j. Furthermore, $a_{j1}, ..., a_{jd}$ are constant and nonnegative factor loadings, assumed to be known. Thus, given the vector $Z = (Z_1, ..., Z_d)$, we have the conditionally independent default probabilities

$$p_j(Z) = P(Y_j = 1|Z) = \Phi\left(\frac{a_j Z + \Phi^{-1}(p_j)}{b_j}\right), \quad j = 1, ..., m, \tag{2.2}$$

where $a_j = (a_{j1}, ..., a_{jd})$.

2.2. Simulation

To evaluate the integrals Monte Carlo simulation is a widely used tool in many fields such as option pricing, risk management, communication engineering, etc.. Since, problems in risk management like in other fields can be written as an expectation under a probability measure, we could use Monte Carlo simulation to solve these problems. It has the advantage of giving an error bound on the final result but has the disadvantage of being rather slow compared to other numerical methods. Nevertheless, it is the fastest alternative for high dimensional integrals.

2.2.1. Naive Monte Carlo Simulation

If the integral we want to evaluate is

$$E_{\mathcal{P}}[g(x)] = \int g(x)\phi(x)dx$$

where $\mathcal{P}$ is the probability measure with density function $\phi(x)$ then the above expectation is computed by

$$\frac{1}{n}\sum_{k=1}^{n} g(x)$$

where x are random numbers generated from density $\phi(x)$ and n is the number of replications.

2.2.2. Importance Sampling

The confidence intervals produced by naive Monte Carlo simulations are too wide to give robust results for rare event simulations. This is why we need variance reduction methods such as Importance Sampling (IS), control variate, etc.. (see Chapter 4 of [11] for a good description of the methods and examples)

The fundamental idea of importance sampling is to modify the probability density in such a way as to reduce variance. If for a probability measure, $\mathcal{P}$; we have density ϕ; we can introduce a measure, $\mathcal{Q}$; with new density, ψ; and rewrite the expectation via

$$E_{\mathcal{P}}[g(x)] = \int g(x)\phi(x)dx = \int g(x)\frac{\phi(x)}{\psi(x)}\psi(x)dx = E_{\mathcal{Q}}[g(x)w(x)]$$

where $w(x) = \frac{\phi(x)}{\psi(x)}$ is called likelihood ratio for IS weight. Our aim is to find a new density ψ such that $g(x)w(x)$ under measure $\mathcal{Q}$ has a lower variance than $g(x)$ under measure $\mathcal{P}$. However, we shoud be careful about the distribution of the likelihood ratio (see [10] and [22]).

Since, we are concerned with default risks of obligors in credit risk, we want more defaults to occur in our IS simulation. Thus, we simply increase the default probabilities of obligors. But, the problem is how much to increase the default probabilities to obtain minimal variance.

In IS simulation, the above expectation is computed by

$$\frac{1}{n}\sum_{k=1}^{n} g(x)w(x)$$

where x are random numbers generated from density ψ instead of ϕ. Thus, the variance of the new estimator is

$$\frac{1}{n}\left(E_{\mathcal{Q}}[g(x)^2 w(x)^2] - \left(E_{\mathcal{P}}[g(x)]\right)^2\right). \tag{2.3}$$

However, the variance formula is as difficult as the original integral. So, for real world problems, the minimization is intractable. But, we can find optimal IS probabilities, minimizing (2.3) in special cases of credit risk problems (see Chapter 3) or we can apply the minimization on the upper bound of this equation in some cases (see below).

2.2.3. The Algorithm of Glasserman & Li

Here we describe the IS approach of Glasserman & Li [13] for simulating tail loss probability. [13] considers the simpler case of independent obligors before dependent obligors. In this case $Y_1, ..., Y_m$ are independent Bernoulli random variables with parameters $p_1, ..., p_m$. For the IS approach, they simply increase each default probability from p_j to some larger value q_j to make large losses more likely. Thus, the estimator of $P(L > x)$ is the product of the indicator $\mathbf{1}_{\{L>x\}}$ and the likelihood ratio

$$\prod_{j=1}^{m} \left(\frac{p_j}{q_j}\right)^{Y_j} \left(\frac{1-p_j}{1-q_j}\right)^{1-Y_j}.$$

Exponential twisting suggest to use a one-parameter family of the form

$$p_j(\theta) = \frac{p_j e^{\theta c_j}}{p_j e^{\theta c_j} + (1-p_j)} \tag{2.4}$$

for the IS probabilities. Here, θ is the exponential twisting parameter and if we choose a $\theta > 0$, we increase each default probability.

Under this condition, the likelihood ratio can be written as $\exp(-\theta L + \psi_L(\theta))$ where

$$\psi_L(\theta) = \log E\left[\exp(\theta L)\right] = \sum_{j=1}^{m} \log\left(1 + p_j(e^{c_j\theta} - 1)\right)$$

is the cumulant generating function of L.

The next step is to optimize the parameter θ to minimize the variance of the simulation. To accomplish that it is enough to consider the second moment of the estimator, given by

$$\begin{aligned} M_2(x,\theta) &= E_\theta\left[e^{-2\theta L + 2\psi_L(\theta)}\mathbf{1}_{\{L>x\}}\right] \\ &\leq \exp(-2\theta x + 2\psi_L(\theta)) \end{aligned} \tag{2.5}$$

where E_θ is the expectation under exponential twisting distribution with parameter θ. Since, it is a difficult problem to optimize the second moment, [13] suggest to optimize the upper bound (2.5).

The minimizer θ_x is the unique solution to

$$\psi_L^{'}(\theta_x) = x \tag{2.6}$$

for the case where $E[L] = \sum_{j=1}^{m} p_j c_j < x$ and equal to 0 otherwise. Note that, a standard property of exponential twisting is that the $E_\theta[L] = \psi_L^{'}(\theta_x)$ when $\theta_x > 0$ so that the expectation under the new measure is always greater than or equal to x. Algorithm given in Figure 2.1 shows the required steps of the method in more detail.

2.2.3.1. Two-step IS of Glasserman & Li: Dependent Obligors. In this secton, we consider the more interesting case where Y_j's are dependent. We consider dependence introduced through a normal copula. [13] applies "inner" IS as in the independent case, but they do so conditional on the common factors $Z = (Z_1, ..., Z_d)^T$. Conditional

1. Compute θ using (2.6) if $E[L] = \sum_{j=1}^{m} p_j c_j < x$ otherwise set $\theta = 0$.
2. Calculate $p_j(\theta), j = 1, ..., m$ as in (2.4).
3. Repeat for replications $k = 1, ..., n$:
   1. generate $Y_j, j = 1, ..., m$, from $p_j(\theta)$;
   2. calculate total loss for replication k, $L^{(k)} = \sum_{j=1}^{m} c_j Y_j$;
   3. calculate likelihood ratio for replication k, $w^{(k)} = \exp(-\theta L^{(k)} + \psi_L(\theta))$;
4. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}} w^{(k)}$.

Figure 2.1. Tail loss probability computation using exponential twisting for independent obligors.

independent probabilities are given in (2.2).

It is possible to compute the conditional cumulant generating function from the conditional independent default probabilities as

$$\psi_{L|Z}(\theta) = \log E[\exp(\theta L)|Z] = \sum_{j=1}^{m} \log(1 + p_j(Z)(e^{c_j\theta} - 1))$$

and solve for the parameter $\theta_x(Z)$ that will minimize the upper bound of the second moment of the estimator as in the independent case. That is, if $E[L|Z] = \sum_{j=1}^{m} p_j(Z)c_j < x$ then $\theta_x(Z)$ is the unique solution to

$$\psi'_{L|Z}(\theta_x(Z)) = x. \tag{2.7}$$

otherwise $\theta_x(Z) = 0$.

[13] then define new conditional default probabilities

$$p_{j,\theta_x(Z)}(Z) = \frac{p_j(Z)e^{\theta_x(Z)c_j}}{p_j(Z)e^{\theta_x(Z)c_j} + (1 - p_j(Z))}, \quad j = 1, ..., m. \tag{2.8}$$

The IS procedure now generates default indicators $Y_1, ..., Y_m$ from the twisted condi-

tional independent default probabilities $p_{j,\theta_x(Z)}(Z), j = 1, ..., m$.

Since L is equal to the sum of the $Y_j c_j$'s,

$$e^{-\theta_x(Z)L+\psi_{L|Z}(\theta_x(Z))}\mathbf{1}_{\{L>x\}} \tag{2.9}$$

is the new IS estimator. Its conditional expectation is $P(L > x|Z)$ and its unconditional expectation is therefore $P(L > x)$.

[13] reports that for highly dependent obligors exponential twisting is not enough for reducing the variance because large losses occur primarily when we have large outcomes of Z. To further reduce the variance, they apply a second step of "outer" importance sampling to Z. For this they consider shifting the mean of Z from the origin to some point μ. See Chapter 5 for mean shift methods proposed in literature and their comparison. We call shifting the mean outer IS and twisting the conditional default probabilities inner IS. The likelihood ratio for the outer IS is

$$e^{-\mu^T Z+\frac{1}{2}\mu^T\mu}.$$

When multiplied by (2.9) this yields the two-step IS estimator

$$\exp(-\mu^T Z + \frac{1}{2}\mu^T\mu - \theta_x(Z)L + \psi_{L|Z}(\theta_x(Z))\mathbf{1}_{\{L>x\}}$$

in which Z is sampled from $N(0, \mu)$ and then Y_j's are generated from the twisted conditional independent default probabilities $p_{j,\theta_x(Z)}(Z), j = 1, ..., m$ to generate L.

What is left is to decide on the magnitude of μ. [14] suggests choosing μ by solving

$$\mu = \max_z P(L > x|Z = z)e^{-\frac{1}{2}z^T z} \tag{2.10}$$

which is the mode of the zero-variance IS distribution for Z that would reduce variance in estimating the integral of $P(L > x|Z)$ with respect to the density of Z.

Solving for μ in (2.10) is not easy because we are indeed simulating to compute $P(L > x|Z = z)$. But, we can use some approximations for $P(L > x|Z = z)$ to compute μ (see Chapter 5). [13] uses a tail bound approximation in which

$$P(L > x|Z = z) \leq e^{-\theta_x(z)x+\psi_{L|Z}(\theta_x(z))}.$$

They use this upper bound in (2.10) to solve for an approximate μ.

Algorithm given in Figure 2.2 shows the required steps of the method in more detail.

---

1. Compute μ using (2.10).
2. Repeat for replications $k = 1, ..., n$:
   1. generate $z_l \sim N(\mu_l, 1), l = 1, ..., d$, independently;
   2. calculate $p_j(Z), j = 1, ..., m$ as in (2.2);
   3. compute $\theta_x(Z)$ using (2.7) if $E[L|Z] = \sum_{j=1}^{m} p_j(Z)c_j < x$ otherwise set $\theta = 0$;
   4. calculate $p_{j,\theta_x(Z)}(Z), j = 1, ..., m$ as in (2.8);
   5. generate $Y_j, j = 1, ..., m$, from $p_{j,\theta_x(Z)}(Z)$;
   6. calculate total loss for replication k, $L^{(k)} = \sum_{j=1}^{m} c_j Y_j$;
   7. calculate likelihood ratio for replication k, $w^{(k)} = \exp(-\mu^T Z + \frac{1}{2}\mu^T\mu - \theta_x(Z)L + \psi_{L|Z}(\theta_x(Z))$;
3. Return $\frac{1}{n}\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)}>x\}} w^{(k)}$.

Figure 2.2. Tail loss probability computation using two-step IS of [13] for dependent obligors.

[13] theoretically shows that exponential twisting for independent obligors and two-step IS for dependent obligors are "asymptotically optimal". This means that the second moment of the IS estimator decreases at the fastest possible rate as the event we simulate becomes rarer.

3. OPTIMAL IS FOR INDEPENDENT OBLIGORS

In this chapter, we assume that obligors default independently. This independence assumption will allow the use use of the binomial distribution for the group of obligors having the same default probability under the simplification of $c_j = 1$ for $j = 1, ..., m$ for simulating the loss probabilities and expected shortfalls.

3.1. Exponential Twisting for Binomial Distribution

We can use the binomial distribution to simulate for the $P(L > x)$ given that obligors default independently, $p_j = p$ and $c_j = 1$ for $j = 1, ..., m$. But the question is: Can we use exponential twisting directly on this binomial distribution?

Since, $L = \sum_{j=1}^{m} c_j Y_j = \sum_{j=1}^{m} Y_j$, L is a random variable with binomial distribution. Thus

$$P(L > x) = \sum_{j=\lfloor x \rfloor + 1}^{m} \binom{m}{j} p^j (1-p)^{m-j}.$$

Applying exponential twisting as IS, the new probability of default becomes

$$p_\theta = \frac{pe^\theta}{1 + p(e^\theta - 1)},$$

and the likelihood ratio is $e^{(-\theta L + \psi(\theta))}$ where

$$\begin{aligned} \psi(\theta) &= \sum_{j=1}^{m} \log(1 + p(e^\theta - 1)) \\ &= m \log(1 + p(e^\theta - 1)) \\ &= \log(1 + p(e^\theta - 1))^m. \end{aligned}$$

So, the new estimator for $P(L > x)$ is

$$\mathbf{1}_{\{L>x\}} e^{-\theta L + \log(1+p_\theta(e^\theta-1))^m} = (1+p(e^\theta-1))^m \mathbf{1}_{\{L>x\}} e^{-\theta L}.$$

If we take the expectation of the new estimator under the new measure;

$$\begin{aligned}
&E_\theta[(1+p(e^\theta-1))^m \mathbf{1}_{\{L>x\}} e^{-\theta L}] \\
&= (1+p(e^\theta-1))^m e^{-\theta x} E_{p_\theta}[\mathbf{1}_{\{L>x\}} e^{-\theta(L-x)}] \\
&= (1+p(e^\theta-1))^m e^{-\theta x} E_{p_\theta}[\mathbf{1}_{\{L-x\}} e^{-\theta(L-x)}] \\
&= (1+p(e^\theta-1))^m e^{-\theta x} \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} p_\theta^j (1-p_\theta)^{m-j} e^{-\theta(j-x)} \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} p_\theta^j (1-p_\theta)^{m-j} e^{-\theta(j-x)} (1+p(e^\theta-1))^m e^{-\theta x} \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} p_\theta^j (1-p_\theta)^{m-j} e^{-\theta j} (1+p(e^\theta-1))^m \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} \left(\frac{pe^\theta}{1+p(e^\theta-1)}\right)^j \left(1-\frac{pe^\theta}{1+p(e^\theta-1)}\right)^{m-j} e^{-\theta j} (1+p(e^\theta-1))^m \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} \frac{p^j}{(1+p(e^\theta-1))^j} \left(\frac{1+p(e^\theta-1)-pe^\theta}{1+p(e^\theta-1)}\right)^{m-j} (1+p(e^\theta-1))^m \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} \frac{p^j}{(1+p(e^\theta-1))^j} \frac{(1-p)^{m-j}}{(1+p(e^\theta-1))^{m-j}} (1+p(e^\theta-1))^m \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} p^j (1-p)^{m-j} \frac{1}{(1+p(e^\theta-1))^m} (1+p(e^\theta-1))^m \\
&= \sum_{j=\lfloor x \rfloor+1}^{m} \binom{m}{j} p^j (1-p)^{m-j}.
\end{aligned}$$

It is nothing but $P(L > x)$. So, the new estimator is an unbiased estimator of $P(L > x)$.

Then, we try to find the θ that will minimize the second moment of the estimator

Table 3.1. Portfolio composition (one obligor case)

Exposure level (c)	Default rate (p)
103	0.01

for $P(L > x)$. The second moment of the new estimator is

$$\begin{aligned} M_2(x,\theta) &= E_\theta \left[e^{-2\theta L + 2\psi(\theta)} \mathbf{1}_{\{L>x\}} \right] \\ &\leq \exp(-2\theta x + 2\psi(\theta)). \end{aligned}$$

To make the optimization problem simple, we minimize the upper bound. Thus,

$$\psi(\theta)' = x. \tag{3.1}$$

The solution to (3.1) is $\theta = \log(\frac{x(1-p)}{p(m-x)})$. So, if $E[L] = mp < x$ then $\theta = \log(\frac{x(1-p)}{p(m-x)})$ otherwise $\theta = 0$.

3.2. Optimal IS Probabilities

We concentrate on a small number of groups of obligors to give optimal IS probabilities. But, this is not a weird assumption. For example, [17] make use of eight group of ratings in his model. We want to see the percentage difference of variances in using optimal and "asymptotically optimal" IS in this section.

3.2.1. One Obligor Case

We first of all consider the simplest case; there is only one obligor in our portfolio. Portfolio composition is given in Table 3.1. The loss distribution for this portfolio is given in Table 3.2. We should consider the case where $0 < x < 103$ so that $P(L > x)$ is different than zero and 1.

Table 3.2. Loss Distribution for the portflio (one obligor case)

i	Loss (L_i)	Probability of loss ($P(L_i)$)
1	103	0.01
2	0	0.99

3.2.1.1. Computing Tail Loss Probability. Our aim is to compute $P(L > x)$. In this simplest case of a portfolio we have an analytical solution

$$P(L > x) = \sum_{i=1}^{2} \mathbf{1}_{\{L_i > x\}} P(L_i) = p$$

where L_i is the loss distribution value which has probability greater than zero.

Naive and IS simulation algorithms are given in Figures 3.1 and 3.2. In the algorithm given in Figure 3.2, p_{new} is the IS default probability for the obligor.

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y from p;
   2. calculate total loss for replication k, $L^{(k)} = cY$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}$.

Figure 3.1. Naive simulation algorithm for $P(L > x)$ (one obligor case)

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y from p_{new};
   2. calculate total loss for replication k, $L^{(k)} = cY$;
   3. calculate likelihood ratio for replication k, $w^{(k)} = \frac{p}{p_{new}}$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}} w^{(k)}$.

Figure 3.2. IS algorithm for $P(L > x)$ (one obligor case)

We compute the expectations and variances of Naive and IS estimators;

$$\begin{aligned} E[\text{Naive Sim. Estimator}] &= \frac{np}{n} = p \\ Var[\text{Naive Sim. Estimator}] &= \frac{np(1-p)}{n^2} = \frac{p(1-p)}{n} \\ E[\text{IS Estimator}] &= \frac{np_{new}\frac{p}{p_{new}}}{n} = p \\ Var[\text{IS Estimator}] &= \left(\frac{p}{p_{new}}\right)^2 \frac{np_{new}(1-p_{new})}{n^2} = \frac{p^2}{n}\frac{1-p_{new}}{p_{new}} \end{aligned} \tag{3.2}$$

Our aim is to find the optimal value of p_{new} that will minimize the variance of the IS estimator. Thus, we take the derivative of 3.2 with respect to p_{new}

$$\frac{\partial}{\partial p_{new}}\left(\frac{p^2}{n}\frac{1-p_{new}}{p_{new}}\right) = \frac{p^2}{n}\frac{-p_{new}-(1-p_{new})}{p_{new}^2} = \frac{1}{n}p^2\frac{-1}{p_{new}^2} < 0.$$

So, optimal value of p_{new} is 1.

If we apply the IS methodology of [13] explained in section 2.2.3, the IS probability is equal to $\frac{x}{c}$ if $E[L] = pc < x$ otherwise equal to p. Thus, the IS probability given by (2.6) is quite different from the optimal one we have computed. And, indeed the optimal has a variance of zero.

3.2.1.2. Computing Conditional Expectation of Loss. Our aim is to compute $E[L|L > x]$. In this simplest case of a portfolio, we have an analytical solution of

$$E[L|L > x] = \frac{1}{P(L > x)}\sum_{i=1}^{2} L_i \mathbf{1}_{\{L_i > x\}} P(L_i) = c$$

where L_i are the loss distribution values having probabilities greater than zero.

Naive and IS simulation algorithms are given in Figures 3.3 and 3.4.

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y from p;
   2. calculate total loss for replication k, $L^{(k)} = cY$;
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}{\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)}>x\}}}$.

Figure 3.3. Naive simulation algorithm for $E[L|L > x]$ (one obligor case)

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y from p_{new};
   2. calculate total loss for replication k, $L^{(k)} = cY$;
   3. calculate likelihood ratio for replication k, $w^{(k)} = \frac{p}{p_{new}}$;
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}{\sum_{k=1}^{n} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}$.

Figure 3.4. IS algorithm for $E[L|L > x]$ (one obligor case)

We compute the expectations and variances of Naive and IS estimators;

$$\begin{aligned}
E[\text{Naive Sim. Estimate}] &= \frac{nc_1 p}{nP(L>x)} = \frac{c_1 p}{P(L>x)} \\
Var[\text{Naive Sim. Estimate}] &= \frac{nc_1^2 p(1-p)}{n^2 P(L>x)^2} = \frac{c_1^2 p(1-p)}{nP(L>x)^2} \\
E[\text{IS Estimate}] &= \frac{nc_1 p_{new} \frac{p}{p_{new}}}{nP(L>x)} = \frac{c_1 p}{P(L>x)} \\
Var[\text{IS Estimate}] &= \left(\frac{pc_1}{p_{new}}\right)^2 \frac{np_{new}(1-p_{new})}{n^2 P(L>x)^2} = \frac{c_1^2 p^2}{nP(L>x)^2} \frac{1-p_{new}}{p_{new}}. \qquad (3.3)
\end{aligned}$$

Our aim is to find the optimal value of p_{new} that will minimize the variance of the IS estimator. Thus, we take the derivative of 3.3 with respect to p_{new}

$$\frac{\partial}{\partial p_{new}} \left(\frac{p^2 c_1^2}{n} \frac{1-p_{new}}{p_{new}}\right) = \frac{p^2 c_1^2}{n} \frac{-p_{new} - (1-p_{new})}{p_{new}^2} = \frac{p^2 c_1^2}{n} \frac{-1}{p_{new}^2} < 0.$$

Thus, optimal value of p_{new} is 1.

Table 3.3. Portfolio composition (independent two obligors)

	Exposure level ($c_j, j = 1, 2$)	Default rate ($p_k, k = A, B$)
Obligor A	103	0.01
Obligor B	105	0.05

Table 3.4. Loss Distribution for the portio (independent two obligors)

i	Loss (L_i)	Probability ($P(L_i)$)
1	$c_1 + c_2 = 208$	$0.01 * 0.05 = 0.0005$
2	$c_2 = 105$	$(1 - 0.01) * 0.05 = 0.0495$
3	$c_1 = 103$	$0.01 * (1 - 0.05) = 0.0095$
4	0	$(1 - 0.01) * (1 - 0.05) = 0.9405$

[12] uses the same exponential twisting probabilities for expected shortfall contributions with tail loss probability computation. Thus, from the previous section we have seen that the exponential twisting probability is quite different from the optimal probability 1.

3.2.2. Two Obligors Case

In this section, we consider the two obligors case. They default independently. The portfolio composition is given in Table 3.3. The loss distribution for the portfolio is given in Table 3.4.

3.2.2.1. Computing Tail Loss Probability. Our aim is to compute $P(L > x)$. It has the analytical solution

$$P(L > x) = \sum_{i=1}^{4} \mathbf{1}_{\{L_i > x\}} P(L_i)$$

where L_i are the loss distribution values having probabilities greater than zero.

Naive and IS simulation algorithms are given in Figures 3.5 and 3.6 to compute $P(L > x)$.

---

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y_1 from p_A and Y_2 from p_B;
   2. calculate total loss for replication k, $L^{(k)} = c_1 Y_1 + c_2 Y_2$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}$.

Figure 3.5. Naive simulation algorithm for $P(L > x)$ (two independent obligors case)

---

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y_1 from $p_{new,A}$ and Y_2 from $p_{new,B}$;
   2. calculate total loss for replication k, $L^{(k)} = c_1 Y_1 + c_2 Y_2$;
   3. calculate likelihood ratio for replication k,
      $w^{(k)} = \left(\frac{p_A}{p_{new,A}}\right)^{Y_1} \left(\frac{1-p_A}{1-p_{new,A}}\right)^{1-Y_1} \left(\frac{p_B}{p_{new,B}}\right)^{Y_2} \left(\frac{1-p_B}{1-p_{new,B}}\right)^{1-Y_2}$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}} w^{(k)}$.

Figure 3.6. IS algorithm for $P(L > x)$ (two independent obligors case)

If $\max\{c_1, c_2\} < x < c_1 + c_2$, the only possibility of $L > x$ is the case where two of the obligors default at the same time. Since, the defaults are independent, the problem simplifies to the one company case where the default probability is the product of two default probabilities of the obligors. So, the multiplication of default probabilities in new measure should be equal to 1 for the optimal IS. Thus, $p_{new,A} = p_{new,B} = 1$ are the optimal values of the new default probabilities for this case.

If $c_1 \leq x \leq c_2$;

$$E[\text{Naive Sim. Estimator}] = \frac{n\left[(1-p_A)p_B + p_A p_B\right]}{n} = p_B$$

$$Var[\text{Naive Sim. Estimator}] = \frac{1}{n}\left[p_B - p_B^2\right]$$

$$E[\text{IS Estimator}] = \frac{n\left(p_{new,A}p_{new,B}\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}} + (1-p_{new,A})p_{new,B}\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}}\right)}{n}$$

$$= p_B.$$

$$
\begin{aligned}
Var[\text{IS Estimator}] &= \frac{n}{n^2}\left(p_{new,A}p_{new,B}\left(\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 + \right. \\
&\quad \left. (1-p_{new,A})p_{new,B}\left(\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 - p_B^2 \right) \\
&= \frac{1}{n}\left(\frac{p_A^2 p_B^2}{p_{new,A}p_{new,B}} + \frac{(1-p_A)^2 p_B^2}{(1-p_{new,A})p_{new,B}} - p_B^2 \right) \\
&= \frac{1}{n}\left(\frac{p_A^2 p_B^2(1-p_{new,A}) + (1-p_A)^2 p_B^2 p_{new,A}}{p_{new,A}p_{new,B}(1-p_{new,A})} - p_B^2 \right) \\
&= \frac{p_B^2}{n}\left(\frac{p_A^2(1-p_{new,A}) + (1-p_A)^2 p_{new,A}}{p_{new,A}p_{new,B}(1-p_{new,A})} - 1 \right) \\
&= \frac{p_B^2}{n}\left(\frac{p_A^2 - p_A^2 p_{new,A} + p_{new,A} - 2p_A p_{new,A} + p_A^2 p_{new,A}}{p_{new,A}p_{new,B}(1-p_{new,A})} - 1 \right) \\
&= \frac{p_B^2}{n}\left(\frac{p_A^2 + p_{new,A} - 2p_A p_{new,A}}{p_{new,A}p_{new,B}(1-p_{new,A})} - 1 \right)
\end{aligned}
$$

To minimize $Var[\text{IS Estimator}]$;

$$
\begin{aligned}
\frac{\partial Var[\text{IS Estimator}]}{\partial p_{new,B}} &= \frac{p_B^2}{n}\frac{-\left(p_A^2 + p_{new,A} - 2p_A p_{new,A}\right) p_{new,A}(1-p_{new,A})}{\left(p_{new,A}p_{new,B}(1-p_{new,A})\right)^2} \\
&= \frac{p_B^2}{n}\frac{-\left(p_A^2 + p_{new,A} - 2p_A p_{new,A}\right)}{p_{new,A}p_{new,B}^2(1-p_{new,A})}
\end{aligned}
$$

So, $\frac{\partial Var[\text{IS Estimator}]}{\partial p_{new,B}} \leq 0$ if $(p_A^2 + p_{new,A} - 2p_A p_{new,A}) \geq 0$ which necessitates $p_{new,A} \geq \frac{-p_A^2}{1-2p_A}$ for $p_A < 0.5$ and $p_{new,A} \leq \frac{-p_A^2}{1-2p_A}$ for $p_A > 0.5$. Since, these conditions are always satisfied, minimum is is attained for $p_{new,B} = 1$.

Then, if we put $p_{new,B} = 1$ into the equation and take the derivative with respect to $p_{new,A}$;

$$\frac{\partial Var[\text{IS Estimator}]}{\partial p_{new,A}} = \frac{p_B^2}{n}\left(\frac{2p_A-1}{p_{new,A}^2-p_{new,A}}+\frac{p_{new,A}-2p_{new,A}p_A+p_A^2}{p_{new,A}^3-p_{new,A}^2}+\frac{p_{new,A}-2p_{new,A}p_A+p_A^2}{p_{new,A}-2p_{new,A}^2+p_{new,A}^3}\right)=0.$$

Solution is: $\begin{cases} \left\{p_A, \frac{p_A}{2p_A-1}\right\} & \text{if} \quad p_A \neq \frac{1}{2} \\ \{p_A\} & \text{if} \quad p_A = \frac{1}{2} \end{cases}$.

To summarize, for all cases $p_A < \frac{1}{2}, p_A > \frac{1}{2}$ and $p_A = \frac{1}{2}$, the optimal solution is $p_{new,A} = p_A$ and $p_{new,B} = 1$.

Thus, the optimal solution is $p_{new,B} = 1$ and $p_{new,A} = p_A$. This is nothing but the variance of the one company case where the default probability is equivalent to p_B.

Note that $c_2 \leq x \leq c_1$ is the same as the previous case but this time we should only change default probability of company A instead of B. So, the optimal solution is $p_{new,A} = 1$ and $p_{new,B} = p_B$.

If $x < \min\{c_1, c_2\}$;

$$E[\text{Naive Sim. Estimator}] = \frac{n\left[1-(1-p_A)(1-p_B)\right]}{n} = p_A + p_B - p_A p_B = \widetilde{p}$$

$$Var[\text{Naive Sim. Estimator}] = \frac{1}{n}\left[\widetilde{p}-\widetilde{p}^2\right]$$

$$
\begin{aligned}
E[\text{IS Estimator}] &= \frac{n}{n}\Bigg[p_{new,A}p_{new,B}\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}} \\
&\quad +(1-p_{new,A})p_{new,B}\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}} \\
&\quad +(1-p_{new,B})p_{new,A}\frac{1-p_B}{1-p_{new,B}}\frac{p_A}{p_{new,A}}\Bigg] \\
&= \widetilde{p} \\
Var[\text{IS Estimator}] &= \frac{n}{n^2}\Bigg[p_{new,A}p_{new,B}\left(\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 \\
&\quad +(1-p_{new,A})p_{new,B}\left(\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 \\
&\quad +(1-p_{new,B})p_{new,A}\left(\frac{1-p_B}{1-p_{new,B}}\frac{p_A}{p_{new,A}}\right)^2-\widetilde{p}^2\Bigg] \\
&= \frac{1}{n}\Bigg[\frac{p_A^2p_B^2}{p_{new,A}p_{new,B}}+\frac{(1-p_A)^2p_B^2}{(1-p_{new,A})p_{new,B}} \\
&\quad +\frac{(1-p_B)^2p_A^2}{(1-p_{new,B})p_{new,A}}-\widetilde{p}^2\Bigg]. \qquad (3.4)
\end{aligned}
$$

There is no easy analytical solution for $p_{new,A}$ and $p_{new,B}$ that minimizes (3.4). But, we can solve it numerically.

If we apply exponential twisting, the optimal solution for the default probabilities are $p_{new,A}=\frac{p_Ae^{\theta c_1}}{1+p_A(e^{\theta c_1}-1)}$ and $p_{new,B}=\frac{p_Be^{\theta c_2}}{1+p_B(e^{\theta c_2}-1)}$ if $x > p_Ac_1+p_Bc_2$ where θ is the unique solution to (2.6) otherwise $p_{new,A}=p_A$ and $p_{new,B}=p_B$.

Let's see how exponential twisting perform with respect to the optimal probabilities in the numerical example of Table 3.3. We observe that exponential twisting probabilities are quite different than the optimal ones especially for large x values in Table 3.5. This makes great difference in standard deviations. Note that, results in this table are based on $1,000,000$ number of replications for all of the methods.

Table 3.5. Optimal (OP) and exponential twisting probabilities (ETP) and corresponding standard deviations (sd) to compute $P(L > x)$ (Two independent obligors case).

x	OP		sd (O)	ETP		sd (ET)	sd (naive)
	$p_{new,A}$	$p_{new,B}$		$p_{new,A}$	$p_{new,B}$		
120	1	1	$4.07 * 10^{-18}$	0.377	0.773	$7.80 * 10^{-7}$	$2.22 * 10^{-5}$
110	1	1	$4.08 * 10^{-18}$	0.324	0.729	$8.99 * 10^{-7}$	$2.22 * 10^{-5}$
104	0.010	1	$9.23 * 10^{-16}$	0.295	0.701	$4.96 * 10^{-5}$	$2.18 * 10^{-4}$
100	0.250	0.750	$4.76 * 10^{-5}$	0.276	0.681	$4.95 * 10^{-5}$	$2.37 * 10^{-4}$
90	0.250	0.750	$4.76 * 10^{-5}$	0.233	0.629	$5.09 * 10^{-5}$	$2.36 * 10^{-4}$

3.2.2.2. Computing Conditional Expectation of Loss. Our aim is to compute $E[L|L > x]$. It has an analytical solution of

$$E[L|L > x] = \frac{1}{P(L > x)} \sum_{i=1}^{4} L_i \mathbf{1}_{\{L_i > x\}} P(L_i)$$

where L_i are the loss distribution values which have probabilities greater than zero.

Naive and IS simulation algorithms are given in Figures 3.7 and 3.8 to compute $E[L|L > x]$.

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y_1 from p_A and Y_2 from p_B;
   2. calculate total loss for replication k, $L^{(k)} = c_1 Y_1 + c_2 Y_2$;
2. Return $\dfrac{\sum_{k=1}^{n} L^{(k)} \mathbf{1}_{\{L^{(k)} > x\}}}{\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}}$

Figure 3.7. Naive simulation algorithm for $E[L|L > x]$ (two independent obligors case)

If $\max\{c_1, c_2\} < x < c_1 + c_2$, the only possibility of $L > x$ is the case where two of the bonds default at the same time. Since, the defaults are independent, the problem

1. Repeat for replications $k = 1, ..., n$:
   1. generate Y_1 from $p_{new,A}$ and Y_2 from $p_{new,B}$;
   2. calculate total loss for replication k, $L^{(k)} = c_1 Y_1 + c_2 Y_2$;
   3. calculate $w^{(k)} = \left(\frac{p_A}{p_{new,A}}\right)^{Y_1} \left(\frac{1-p_A}{1-p_{new,A}}\right)^{1-Y_1} \left(\frac{p_B}{p_{new,B}}\right)^{Y_2} \left(\frac{1-p_B}{1-p_{new,B}}\right)^{1-Y_2}$
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}{\sum_{k=1}^{n} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}$.

Figure 3.8. IS algorithm for $E[L|L > x]$ (two independent obligors case)

simplifies to the one company case where default probability is the multiplication of two default probabilities of the obligors. So, the multiplication of default probabilities in new measure should be equal to 1 for the optimal IS. Thus, $p_{new,A} = p_{new,B} = 1$ are the optimal values of new default probabilities for this case.

If $c_1 \leq x \leq c_2$;

$$E[\text{Naive Sim. Estimator}] = \frac{c_2(1-p_A)p_B + (c_1+c_2)p_A p_B}{P(L>x)} = \frac{c_2 p_B + c_1 p_A p_B}{P(L>x)}$$

$$\begin{aligned} E[\text{IS Estimator}] &= \frac{1}{P(L>x)} \Bigg((c_1+c_2) p_{new,A} p_{new,B} \frac{p_A}{p_{new,A}} \frac{p_B}{p_{new,B}} \\ &\quad + c_2(1-p_{new,A}) p_{new,B} \frac{1-p_A}{1-p_{new,A}} \frac{p_B}{p_{new,B}} \Bigg) \\ &= \frac{c_2 p_B + c_1 p_A p_B}{P(L>x)} \end{aligned}$$

$$
\begin{aligned}
Var[\text{IS Estimator}] &= \frac{n}{P(L>x)^2 n^2}\Bigg[(c_1+c_2)^2 p_{new,A}p_{new,B}\left(\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 \\
&\quad +c_2^2(1-p_{new,A})p_{new,B}\left(\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2-(c_2p_B+c_1p_Ap_B)^2\Bigg] \\
&= \frac{1}{nP(L>x)^2}\Bigg[(c_1+c_2)^2\frac{p_A^2}{p_{new,A}}\frac{p_B^2}{p_{new,B}} \\
&\quad +c_2^2\frac{(1-p_A)^2}{1-p_{new,A}}\frac{p_B^2}{p_{new,B}}-(c_2p_B+c_1p_Ap_B)^2\Bigg]. \qquad (3.5)
\end{aligned}
$$

If $x < \min\{c_1, c_2\}$;

$$
\begin{aligned}
E[\text{Naive Sim. Estimator}] &= \frac{1}{P(L>x)}\Big[c_1(1-p_B)p_A+c_2(1-p_A)p_B+(c_1+c_2)p_Ap_B\Big] \\
&= \frac{c_1p_A+c_2p_B}{P(L>x)}
\end{aligned}
$$

$$
\begin{aligned}
E[\text{IS Estimator}] &= \frac{1}{P(L>x)}\Bigg[(c_1+c_2)p_{new,A}p_{new,B}\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}} \\
&\quad +c_2(1-p_{new,A})p_{new,B}\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}} \\
&\quad +c_1(1-p_{new,B})p_{new,A}\frac{1-p_B}{1-p_{new,B}}\frac{p_A}{p_{new,A}}\Bigg] \\
&= \frac{c_1p_A+c_2p_B}{P(L>x)}
\end{aligned}
$$

Table 3.6. Optimal and exponential twisting probabilities and corresponding standard deviations (sd) to compute $E[L|L > x]$ (Two independent obligors case).

x	OP		sd (O)	ETP		sd (ET)	sd (naive)
	$p_{new,A}$	$p_{new,B}$		$p_{new,A}$	$p_{new,B}$		
120	1	1	$4.22 * 10^{-12}$	0.377	0.773	0.324	9.30
110	1	1	$4.22 * 10^{-12}$	0.324	0.729	0.374	9.30
104	0.020	1.000	$2.90 * 10^{-4}$	0.295	0.701	0.103	0.464
100	0.248	0.753	0.082	0.276	0.681	0.086	0.422
90	0.248	0.753	0.082	0.233	0.629	0.089	0.421

$$
\begin{aligned}
Var[\text{IS Estimator}] &= \frac{n}{P(L>x)^2 n^2}\Bigg[(c_1+c_2)^2 p_{new,A}p_{new,B}\left(\frac{p_A}{p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 \\
&\quad + c_2^2(1-p_{new,A})p_{new,B}\left(\frac{1-p_A}{1-p_{new,A}}\frac{p_B}{p_{new,B}}\right)^2 \\
&\quad + c_1^2(1-p_{new,B})p_{new,A}\left(\frac{1-p_B}{1-p_{new,B}}\frac{p_A}{p_{new,A}}\right)^2 - (c_1p_A+c_2p_B)^2\Bigg] \\
&= \frac{1}{nP(L>x)^2}\Bigg[(c_1+c_2)^2\frac{p_A^2}{p_{new,A}}\frac{p_B^2}{p_{new,B}} + c_2^2\frac{(1-p_A)^2}{1-p_{new,A}}\frac{p_B^2}{p_{new,B}} \\
&\quad + c_1^2\frac{(1-p_B)^2}{1-p_{new,B}}\frac{p_A^2}{p_{new,A}} - (c_1p_A+c_2p_B)^2\Bigg]. \qquad (3.6)
\end{aligned}
$$

We numerically solve (3.5) and (3.6) for the optimal probabilities. Table 3.6 presents how exponential twisting probabilities perform compared to the optimal probabilities when computing $E[L|L > x]$ in the numerical example of Table 3.3. We observe that exponential twisting probabilities are quite different than the optimal ones especially for large x values in Table 3.6. This makes great difference in standard deviations. Note that, results in this table are based on $1,000,000$ number of replications for all of the methods.

3.2.3. One Homogeneous Group of Obligors

We show that we can use binomial distribution in IS of [13] to compute $P(L > x)$ for an homogeneous portfolio in section 3.1. But, the question is what is the performance of using this approach to compute $P(L > x)$ and $E[L|L > x]$ compared to the optimal IS strategy.

Let's take $p_j = p$ and $c_j = 1$ from $j = 1, ..., m$ so that $L = \sum_{j=1}^{m} c_j Y_j = \sum_{j=1}^{m} Y_j$ is a random variable with binomial distribution with parameters p and m.

3.2.3.1. Computing Tail Loss Probability. Our aim is to compute $P(L > x)$ which has an analytical solution of

$$P(L > x) = \sum_{i=\lfloor x \rfloor + 1}^{m} \binom{m}{i} p^i (1-p)^{m-i}$$

under the assumptions we make.

Naive and IS simulation algorithms are given in Figures 3.9 and 3.10 to compute $P(L > x)$.

---

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L^{(k)}$ from binomial distribution with parameters p and m;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}$.

Figure 3.9. Naive simulation algorithm for $P(L > x)$ (One homogeneous group of obligors case)

It is easy to show the unbiasedness of IS estimator. Our aim is to choose a p_{new} that will minimize the variance of the IS estimator.

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L^{(k)}$ from binomial distribution with parameters p_{new} and m;
   2. calculate likelihood ratio for replication k, $w^{(k)} = \left(\frac{p}{p_{new}}\right)^{L^{(k)}} \left(\frac{1-p}{1-p_{new}}\right)^{m-L^{(k)}}$;
2. Return $\frac{1}{n}\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)}>x\}} w^{(k)}$.

Figure 3.10. IS algorithm for $P(L > x)$ (One homogeneous group of obligors case)

The variance of the IS estimator is

$$\begin{aligned}
&\frac{1}{n}\left[\sum_{i=\lfloor x\rfloor+1}^{m}\left(\frac{p}{p_{new}}\right)^{2i}\left(\frac{1-p}{1-p_{new}}\right)^{2m-2i}\binom{m}{i}p_{new}^{i}(1-p_{new})^{m-i}\right.\\
&\left.-\left(\sum_{i=\lfloor x\rfloor+1}^{m}\binom{m}{i}p^{i}(1-p)^{m-i}\right)^{2}\right]\\
=\;&\frac{1}{n}\left[\sum_{i=\lfloor x\rfloor+1}^{m}\binom{m}{i}\frac{p^{2i}\left(1-p\right)^{2m-2i}}{p_{new}^{i}\left(1-p_{new}\right)^{m-i}}-\left(\sum_{i=\lfloor x\rfloor+1}^{m}\binom{m}{i}p^{i}(1-p)^{m-i}\right)^{2}\right].
\end{aligned}$$

Since, the last term (square of the expectation of the naive and IS estimator) does not depend on p_{new}, it is enough to minimize the first term (second moment of the estimator). Thus, our optimization problem is

$$\min_{p_{new}} \sum_{i=\lfloor x\rfloor+1}^{m}\binom{m}{i}\frac{p^{2i}\left(1-p\right)^{2m-2i}}{p_{new}^{i}\left(1-p_{new}\right)^{m-i}}$$

which we solve numerically.

Let's see how exponential twisting probabilities perform compared to the optimal probabilities in a numerical example given in Table 3.7. We observe that exponential twisting probabilities are quite close to the optimal ones for $x > 10$. This is why we don't see much difference in standard deviations. Note that, results in this table are based on $1,000,000$ number of replications for all of the methods.

Table 3.7. Optimal and exponential twisting IS probabilities and corresponding standard deviations (sd) for the parameter values of $m = 100$ and $p = 0.1$ to compute $P(L > x)$.

x	OP	sd (O)	ETP	sd (ET)	sd (naive)
30	0.311	$1.51 * 10^{-11}$	0.300	$1.53 * 10^{-11}$	$7.78 * 10^{-8}$
25	0.261	$9.17 * 10^{-9}$	0.250	$9.35 * 10^{-9}$	$2.02 * 10^{-6}$
20	0.212	$1.54 * 10^{-6}$	0.200	$1.58 * 10^{-6}$	$2.84 * 10^{-5}$
15	0.164	$5.80 * 10^{-5}$	0.150	$6.15 * 10^{-5}$	$1.96 * 10^{-4}$
10	0.122	$3.57 * 10^{-4}$	0.100	$4.93 * 10^{-4}$	$4.93 * 10^{-4}$

3.2.3.2. Computing Conditional Expectation of Loss. Our aim is to compute $E[L|L > x]$ which has an analytical solution of

$$E[L|L > x] = \frac{1}{P(L > x)} \sum_{i=\lfloor x \rfloor + 1}^{m} i \binom{m}{i} p^i (1-p)^{m-i}.$$

Naive and IS simulation algorithms are given in Figures 3.11 and 3.12.

1. Repeat for replications $k = 1, ..., n$:
    1. generate $L^{(k)}$ from binomial distribution with parameters p and m;
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} \mathbf{1}_{\{L^{(k)} > x\}}}{\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}}$.

Figure 3.11. Naive simulation algorithm for $E[L|L > x]$ (One homogeneous group of obligors case)

Under the assumption of large number of replications, the denominator of the ratio becomes $P(L > x)$. So, it is easy to show the asymptotically unbiasedness of the IS estimator. Our aim is to choose a p_{new} that will minimize the variance of IS estimator.

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L^{(k)}$ from binomial distribution with parameters p_{new} and m;
   2. calculate total loss for replication k, $L^{(k)} = cY$;
   3. calculate likelihood ratio for replication k, $w^{(k)} = \left(\frac{p}{p_{new}}\right)^{L^{(k)}} \left(\frac{1-p}{1-p_{new}}\right)^{m-L^{(k)}}$;
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}{\sum_{k=1}^{n} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}$.

Figure 3.12. IS algorithm for $E[L|L > x]$ (One homogeneous group of obligors case)

The asymptotic variance of the IS estimator is

$$\frac{1}{nP(L>x)^2}\left[\sum_{i=\lfloor x\rfloor+1}^{m} i^2 \left(\frac{p}{p_{new}}\right)^{2i} \left(\frac{1-p}{1-p_{new}}\right)^{2m-2i} \binom{m}{i} p_{new}^i (1-p_{new})^{m-i} - \left(\sum_{i=\lfloor x\rfloor+1}^{m} i\binom{m}{i} p^i (1-p)^{m-i}\right)^2\right]$$

$$= \frac{1}{nP(L>x)^2}\left[\sum_{i=\lfloor x\rfloor+1}^{m} \binom{m}{i} i^2 \frac{p^{2i}\,(1-p)^{2m-2i}}{p_{new}^i\,(1-p_{new})^{m-i}} - \left(\sum_{i=\lfloor x\rfloor+1}^{m} \binom{m}{i} i p^i (1-p)^{m-i}\right)^2\right].$$

Since, the last term (square of the expectation of the naive and IS estimator) does not depend on p_{new}, it is enough to solve for p_{new} that will minimize the first term (second moment of the estimator) to minimize the variance. Thus, our optimization problem is

$$\min_{p_{new}} \sum_{i=\lfloor x\rfloor+1}^{m} \binom{m}{i} i^2 \frac{p^{2i}\,(1-p)^{2m-2i}}{p_{new}^i\,(1-p_{new})^{m-i}}$$

which we solve numerically.

Let's see how exponential twisting probabilities perform compared to the optimal probabilities in a numerical example given in Table 3.8. We observe that exponential twisting probabilities are quite close to the optimal ones for $x > 10$. This is why we don't see much difference in standard deviations. Note that, results in this table are

Table 3.8. Optimal and exponential twisting IS probabilities and corresponding standard deviations (sd) for the parameter values of $m = 100$ and $p = 0.1$ to compute $E[L|L > x]$.

x	OP	sd (O)	ETP	sd (ET)	sd (naive)
30	0.311	0.083	0.300	0.085	402.640
25	0.261	0.064	0.250	0.065	13.051
20	0.212	0.046	0.200	0.047	0.761
15	0.165	0.029	0.150	0.030	0.085
10	0.125	0.016	0.100	0.020	0.020

based on $1,000,000$ number of replications for all of the methods.

3.2.4. Two Homogenous Groups of Obligors

The last case of this section is to have two homogenous group of obligors. Again, all obligors default independently. We assume that for the first group A, $c_j = 1, j = 1, ..., m_A$ and the second B, $c_j = 1, j = 1, ..., m_B$. Moreover, we assume that x takes integer values.

3.2.4.1. Computing Tail Loss Probability. Our aim is to compute $P(L > x)$ which has an analytical solution of

$$P(L > x) = \sum_{i=0}^{m_A} \left[\binom{m_A}{i} p_A^i (1-p_A)^{m_A - i} \sum_{j=[\lfloor x \rfloor - i + 1]^+}^{m_B} \binom{m_B}{j} p_B^j (1-p_B)^{m_B - j} \right]$$

where $[x]^+ = \max(0, x)$.

Naive and IS simulation algorithms are given in Figures 3.13 and 3.14 to compute $P(L > x)$.

Our aim is to minimize the variance of IS estimator to compute the optimal IS

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L_A^{(k)}$ from binomial distribution with parameters p_A and m_A and $L_B^{(k)}$ from binomial distribution with parameters p_B and m_B;
   2. calculate total loss for replication k, $L^{(k)} = L_A^{(k)} + L_B^{(k)}$.
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}$.

Figure 3.13. Naive simulation algorithm for $P(L > x)$ (Two homogeneous groups of obligors case)

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L_A^{(k)}$ from binomial distribution with parameters $p_{new,A}$ and m_A and $L_B^{(k)}$ from binomial distribution with parameters $p_{new,B}$ and m_B;
   2. calculate likelihood ratio for replication k, $w^{(k)} = \left(\frac{p_A}{p_{new,A}}\right)^{L_A^{(k)}} \left(\frac{1-p_A}{1-p_{new,A}}\right)^{m_A - L_A^{(k)}} \left(\frac{p_B}{p_{new,B}}\right)^{L_B^{(k)}} \left(\frac{1-p_B}{1-p_{new,B}}\right)^{m_B - L_B^{(k)}}$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}} w^{(k)}$.

Figure 3.14. IS algorithm for $P(L > x)$ (Two homogeneous groups of obligors case)

probabilities. The resulting minimization problem is

$$\begin{aligned}
&\min_{p_{new,A},p_{new,B}} \sum_{i=0}^{m_A} \binom{m_A}{i} p_{new,A}^{i}(1-p_{new,A})^{m_A-i} \sum_{j=[\lfloor x \rfloor -i+1]^+}^{m_B} \binom{m_B}{j} p_{new,B}^{j}(1-p_{new,B})^{m_B-j} \\
&\quad * \left(\frac{p_A}{p_{new,A}}\right)^{2i} \left(\frac{1-p_A}{1-p_{new,A}}\right)^{2m_A-2i} \left(\frac{p_B}{p_{new,B}}\right)^{2j} \left(\frac{1-p_B}{1-p_{new,B}}\right)^{2m_B-2j} \\
&= \min_{p_{new,A},p_{new,B}} \sum_{i=0}^{m_A} \binom{m_A}{i} \frac{p_A^{2i}(1-p_A)^{2m_A-2i}}{p_{new,A}^{i}(1-p_{new,A})^{m_A-i}} \sum_{j=[\lfloor x \rfloor -i+1]^+}^{m_B} \binom{m_B}{j} \\
&\quad * \frac{p_B^{2j}(1-p_B)^{2m_B-2j}}{p_{new,B}^{j}(1-p_{new,B})^{m_B-j}}
\end{aligned}$$

which we solve numerically.

We give a numerical example to have an idea about how close the optimal and exponential twisting probabilities and corresponding variances are in Table 3.9. Note that, results in this table are based on $1,000,000$ number of replications for all of the methods. Furthermore, in Figures 3.15 to 3.20, we not only give optimal and exponential twisting probabilities proposed by [13] (represented as $+$) but also contour lines of variances and possible exponential twisting pairs of probabilities for $P(L > x)$ for the parameters in Table 3.9. Note that we use R [30] for plotting the graphs. We observe that exponential twisting could be used to compute the optimal probabilities from the figures. However, exponential twisting probabilities proposed by [13] are far away from the optimal probabilities when the number of obligors is small and the tail loss probability we simulate is not very small, as they use the tail approximation of the variance in optimizing the exponential twisting parameter (θ). As it can be seen from our figures and Table 3.9 exponential twisting is asymptotically optimal. That is when the number of obligors increase and tail loss probability we simulate becomes smaller (x increase), exponential twisting probabilities proposed by [13] are quite close to the optimal ones.

Table 3.9. Optimal and exponential twisting probabilities and corresponding variances (var).

m_A	m_B	x	p_A	p_B	OP		ETP		var (O)	var (ET)	var (naive)
5	2	2	0.05	0.05	0.428	0.429	0.286	0.286	$2.93E-11$	$4.61E-11$	$3.74E-9$
5	2	2	0.1	0.05	0.488	0.297	0.326	0.186	$4.96E-10$	$8.09E-10$	$1.62E-8$
10	5	4	0.05	0.05	0.338	0.336	0.267	0.267	$1.11E-12$	$1.38E-12$	$6.14E-10$
10	5	4	0.1	0.05	0.388	0.227	0.312	0.177	$8.03E-11$	$1.02E-10$	$5.66E-9$
100	50	30	0.05	0.05	0.208	0.207	0.200	0.200	$1.92E-27$	$1.97E-27$	$1.61E-17$
100	50	30	0.1	0.05	0.248	0.134	0.236	0.128	$1.86E-17$	$1.91E-17$	$1.87E-12$

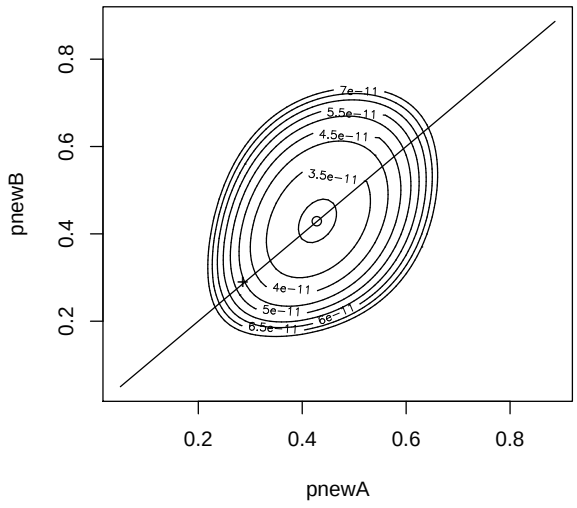

Figure 3.15. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.05, p_B = 0.05, m_A = 5, m_B = 2$ and $x = 2$ in computing $P(L > x)$.

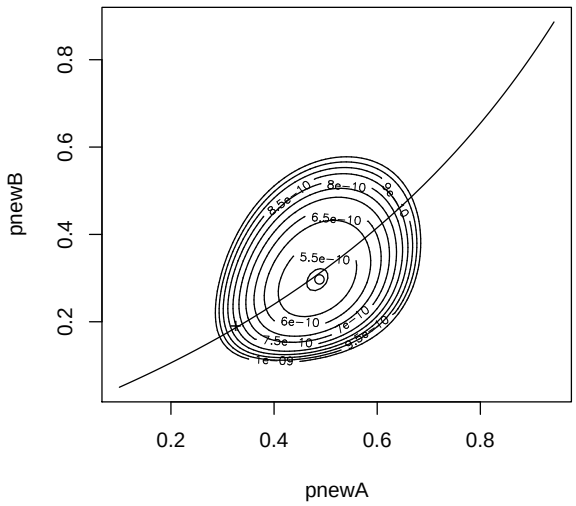

Figure 3.16. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.1, p_B = 0.05, m_A = 5, m_B = 2$ and $x = 2$ in computing $P(L > x)$.

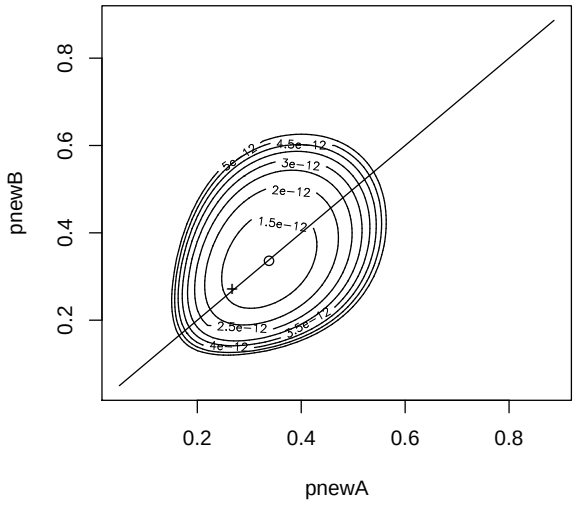

Figure 3.17. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.05, p_B = 0.05, m_A = 10, m_B = 5$ and $x = 4$ in computing $P(L > x)$.

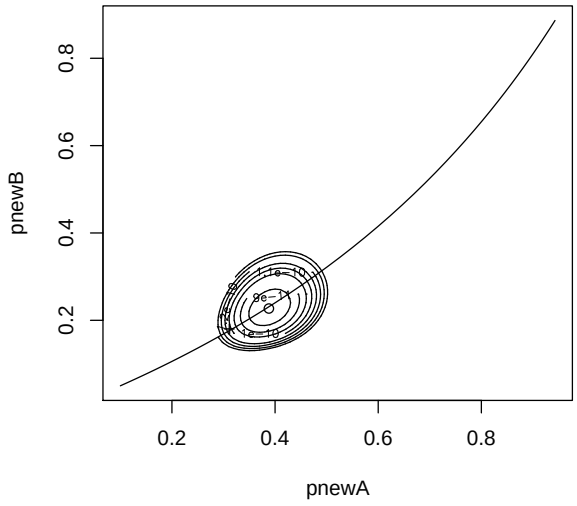

Figure 3.18. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.1, p_B = 0.05, m_A = 10, m_B = 5$ and $x = 4$ in computing $P(L > x)$.

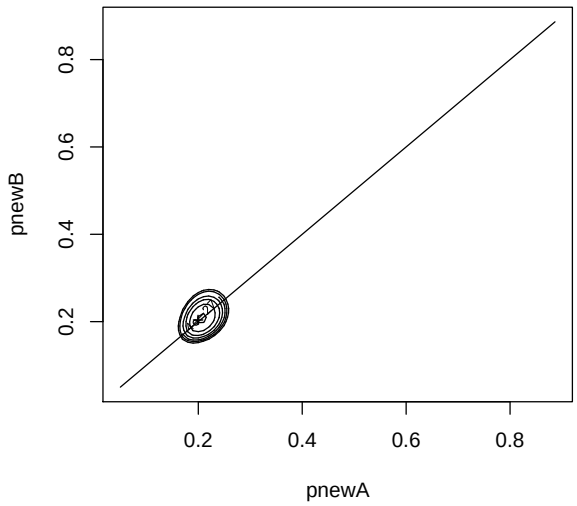

Figure 3.19. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.05, p_B = 0.05, m_A = 100, m_B = 50$ and $x = 30$ in computing $P(L > x)$.

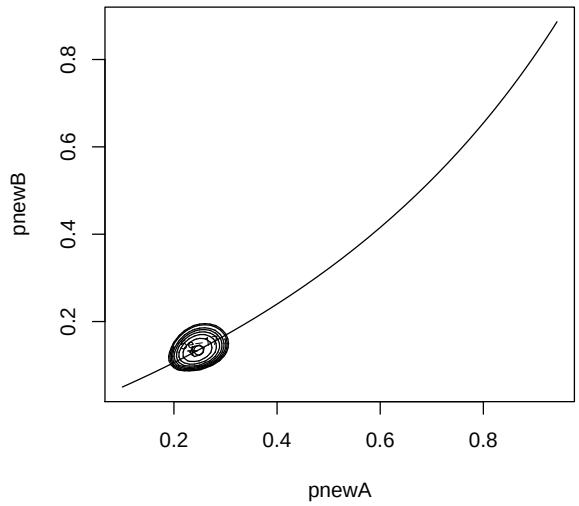

Figure 3.20. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.1, p_B = 0.05, m_A = 100, m_B = 50$ and $x = 30$ in computing $P(L > x)$.

3.2.4.2. Computing Conditional Expectation of Loss. Our aim is to compute $E[L|L > x]$ which has an analytical solution of

$$E(L|L > x) = \frac{1}{P(L > x)} \sum_{i=0}^{m_A} \sum_{j=[\lfloor x \rfloor - i + 1]^+}^{m_B} (i + j) \binom{m_A}{i} p_A^i (1 - p_A)^{m_A - i} \binom{m_B}{j} p_B^j (1 - p_B)^{m_B - j}.$$

Naive and IS simulation algorithms are given in Figures 3.21 and 3.22 to compute $E[L|L > x]$.

Under the assumption of a large number of replications, the denominator of the ratio becomes $P(L > x)$. So, it is easy to show the asymptotical unbiasedness of the IS estimator. Our aim is to choose the $p_{new,A}$ and $p_{new,B}$ pair that will minimize the variance of IS estimator.

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L_A^{(k)}$ from binomial distribution with parameters p_A and m_A and $L_B^{(k)}$ from binomial distribution with parameters p_B and m_B;
   2. calculate total loss for replication k, $L^{(k)} = L_A^{(k)} + L_B^{(k)}$.
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}{\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)}>x\}}}$.

Figure 3.21. Naive simulation algorithm for $E[L|L > x]$ (Two homogeneous group of obligors case)

1. Repeat for replications $k = 1, ..., n$:
   1. generate $L_A^{(k)}$ from binomial distribution with parameters $p_{new,A}$ and m_A and $L_B^{(k)}$ from binomial distribution with parameters $p_{new,B}$ and m_B;
   2. calculate likelihood ratio for replication k, $w^{(k)} = \left(\frac{p_A}{p_{new,A}}\right)^{L_A^{(k)}} \left(\frac{1-p_A}{1-p_{new,A}}\right)^{m_A - L_A^{(k)}} \left(\frac{p_B}{p_{new,B}}\right)^{L_B^{(k)}} \left(\frac{1-p_B}{1-p_{new,B}}\right)^{m_B - L_B^{(k)}}$;
2. Return $\frac{\sum_{k=1}^{n} L^{(k)} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}{\sum_{k=1}^{n} w^{(k)} \mathbf{1}_{\{L^{(k)}>x\}}}$.

Figure 3.22. IS algorithm for $E[L|L > x]$ (Two homogeneous group of obligors case)

The resulting minimization problem is

$$\min_{p_{new,A},p_{new,B}} \sum_{i=0}^{m_A} \sum_{j=[\lfloor x \rfloor - i+1]^+}^{m_B} (i+j)^2 \left(\frac{p_A}{p_{new,A}}\right)^{2i} \left(\frac{1-p_A}{1-p_{new,A}}\right)^{2m_A-2i} * \left(\frac{p_B}{p_{new,B}}\right)^{2j} \left(\frac{1-p_B}{1-p_{new,B}}\right)^{2m_B-2j} \binom{m_A}{i} p_{new,A}^i (1-p_{new,A})^{m_A-i} * \binom{m_B}{j} p_{new,B}^j (1-p_{new,B})^{m_B-j}$$

$$= \min_{p_{new,A},p_{new,B}} \sum_{i=0}^{m_A} \binom{m_A}{i} \frac{p_A^{2i}(1-p_A)^{2m_A-2i}}{p_{new,A}^i (1-p_{new,A})^{m_A-i}} \sum_{j=[\lfloor x \rfloor - i+1]^+}^{m_B} (i+j)^2 \binom{m_B}{j} * \frac{p_B^{2j}(1-p_B)^{2m_B-2j}}{p_{new,B}^j (1-p_{new,B})^{m_B-j}}$$

which we solve numerically.

We give a numerical example to have an idea about how close the optimal and exponential twisting probabilities proposed by [13] and corresponding variances are in Table 3.10. Note that, results in this table are based on $1,000,000$ number of replications for all of the methods. Furthermore, in Figures 3.23 to 3.28, we not only give optimal and exponential twisting probabilities proposed by [13] (represented as $+$) but also contour lines of variances and possible exponential twisting pairs of probabilities for $E[L|L > x]$ for the parameters in Table 3.10. Since, we use the same exponential twisting probabilities of [13] for both $P(L > x)$ and $E[L|L > x]$, and also optimal probabilities we compute are nearly the same for both cases, the comments we made for $P(L > x)$ are completely valid for this section as well.

We give more numerical results in Table A.1 to A.4. Please note that, results in these tables are based on $1,000,000$ replications for all of the methods. It is obvious that when we increase the number of obligors and keep the default probabilities the same, percent differences of variances becomes smaller between exponential twisting and optimal IS (see Tables A.3 and A.4). The reason is that asymptotical optimality is not only related with rarity of the event we simulate but with the number of obligors in the portfolio. So, when we increase the number of obligors we are indeed making the event less rare but performance of exponential twisting is improving. However, there

Table 3.10. Optimal and exponential twisting IS probabilities and corresponding variances (var).

m_A	m_B	x	p_A	p_B	OP		ETP		var (O)	var (ET)	var (naive)
5	2	2	0.05	0.05	0.430	0.430	0.286	0.286	$1.84E-5$	$2.93E-5$	$2.00E-3$
5	2	2	0.1	0.05	0.485	0.297	0.326	0.186	$1.61E-5$	$2.68E-5$	$5.76E-4$
10	5	4	0.05	0.05	0.334	0.334	0.267	0.267	$7.29E-5$	$9.09E-5$	$4.20E-2$
10	5	4	0.1	0.05	0.390	0.228	0.312	0.177	$6.14E-5$	$7.89E-5$	$5.00E-3$
100	50	30	0.05	0.05	0.207	0.207	0.200	0.200	0.007	0.007	$6.07E7$
100	50	30	0.1	0.05	0.245	0.133	0.236	0.128	0.005	0.005	530.49

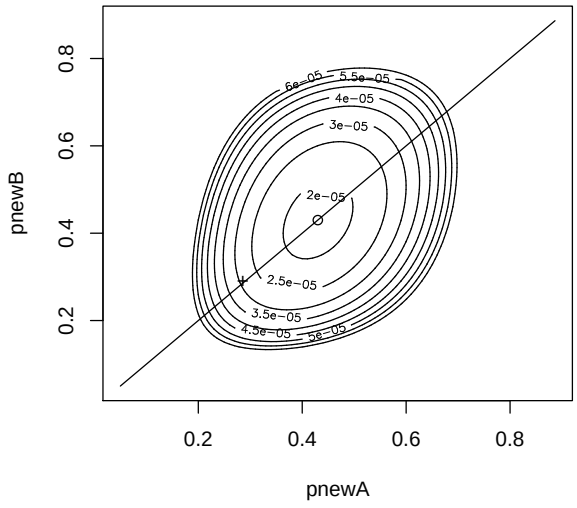

Figure 3.23. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.05, p_B = 0.05, m_A = 5, m_B = 2$ and $x = 2$ in computing $E[L|L > x]$.

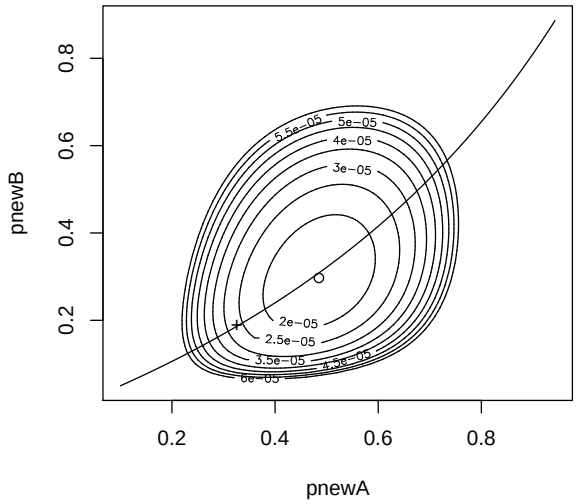

Figure 3.24. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.1, p_B = 0.05, m_A = 5, m_B = 2$ and $x = 2$ in computing $E[L|L > x]$.

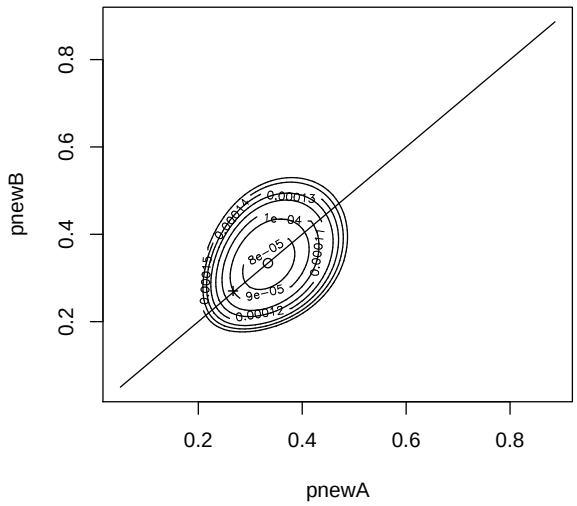

Figure 3.25. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.05, p_B = 0.05, m_A = 10, m_B = 5$ and $x = 4$ in computing $E[L|L > x]$.

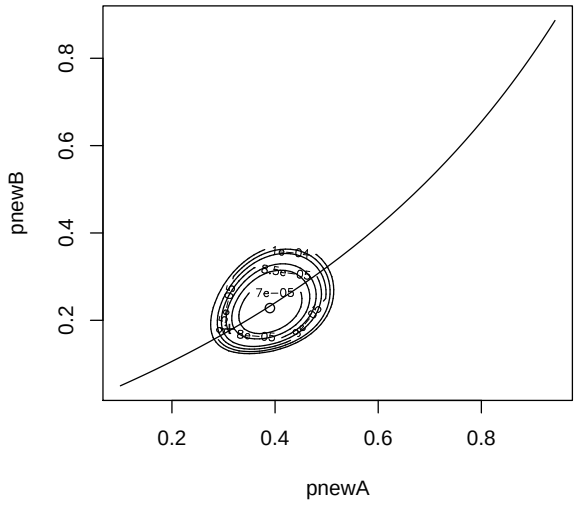

Figure 3.26. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.1, p_B = 0.05, m_A = 10, m_B = 5$ and $x = 4$ in computing $E[L|L > x]$.

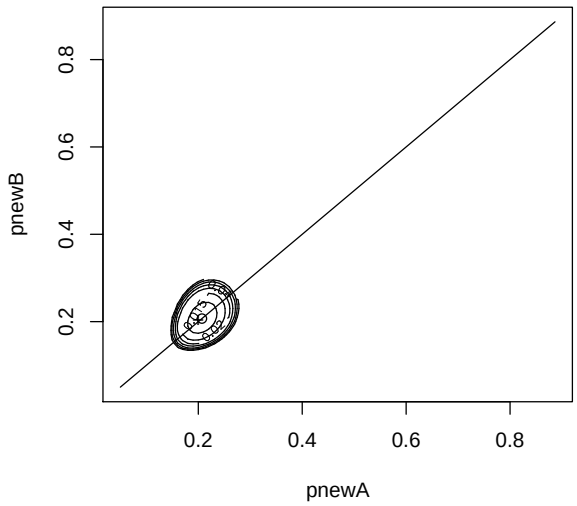

Figure 3.27. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.05, p_B = 0.05, m_A = 100, m_B = 50$ and $x = 30$ in computing $E[L|L > x]$.

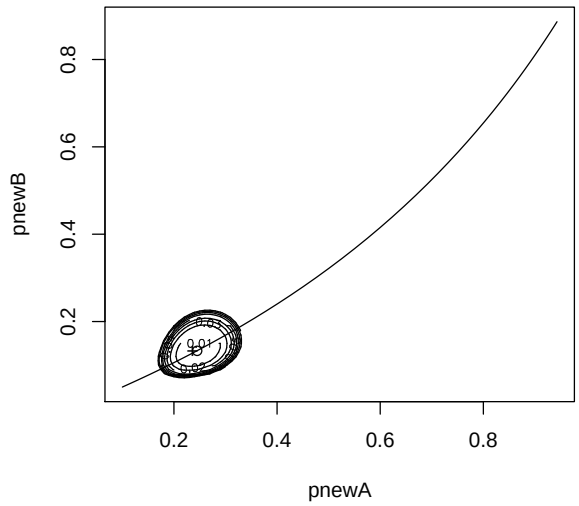

Figure 3.28. Optimal and exponential twisting [13] probabilities and corresponding variances for $p_A = 0.1, p_B = 0.05, m_A = 100, m_B = 50$ and $x = 30$ in computing $E[L|L > x]$.

are exceptions to this situation in Tables A.1 and A.2. The reason is that for these tables $x = 1$. So, the event we simulate is not rare as in Tables A.3 and A.4. And, sometimes increasing the number of obligors can worsen the asymptotical optimality since the event was not rare enough and we made it less rare.

The second observation is that when we keep the number of obligors the same but increase the default probabilities, the performance of exponential twisting gets worse. However, there is an exception to this. If the default probabilities are the same then this has an affect on the performance of exponential twisting. Such that percent difference of variances are not increasing further but decreasing. Under symmetric default probabilities, exponential twisting works better compared to the non-symmetric cases.

If we compare the percent differences for $x = 1$ and $x = 5$ for both tail loss probabilities and for expected shortfall, percent differences are always smaller for $x = 5$. As we explained above, this is directly related with the rarity of the event we simulate.

Finally, some of the exponential twisting probabilities are the same with the naive ones in Tables A.1 and A.2. As we have explained throughout the chapter, if the expected value of the loss is greater than or equal to x then the exponential twisting probabilities are set to the original probabilities.

3.3. Application Example

In this section our aim is to show how our methodology which is used throughout the chapter could be used to compute optimal probabilities for two small financial examples.

Our first example is quite similar to the numerical example of [33] where it is used to demonstrate that expected shortfall is a better risk measure than VaR when one considers the tail risk of the portfolio. See Table 3.11 for the specific profile of bonds in the mutual funds. Suppose that we want to invest 100 million dollar to these mutual funds. We assume that defaults occur totally independently and maturity is 1 year. Moreover, we accept that all the information given in Table 3.11 is constant in this 1 year duration.

If we use the notation below;
$W0$: Initial wealth
W : Wealth at the end of the year 1
$X1$: Amount invested in portfolio A
$X2$: Amount invested in portfolio B,
then the expected utility (assuming logarithmic utility) of the final wealth will be

$$E[u(W)] = \sum_{i=0}^{100} \binom{100}{i} 0.01^i 0.99^{100-i} \sum_{j=0}^{50} \binom{50}{j} 0.05^j 0.95^{50-j} \ln(\tilde{w}(i,j))$$

where

$$\tilde{w}(i,j) = 1.03X_1\frac{100-i}{100} + 1.05X_2\frac{50-j}{50} + 1.02(W_0 - X_1 - X_2). \qquad (3.7)$$

Table 3.11. Profiles of bonds included in the mutual funds

	Number of bonds	Coupon (%)	Default rate (%)
Diversied portfolio A	100	3.00	1.00
Diversied portfolio B	50	5.00	5.00
Risk-free asset	1	2.00	0

To compute VaR at $100(1-\alpha)$ percent confidence level, we need to evaluate $P(u(W) < x)$ for a bunch of x values until this probability approximately equals α. Thus, we need to use IS to simulate efficiently $P(u(W) < x)$.

Since, $P(u(W) < x) = P(W < exp(x))$, we can define $\tilde{x} = exp(x)$ to get rid of the utility function. $P(W < \tilde{x})$ has the analytical solution of

$$\sum_{i=0}^{100} \binom{100}{i} 0.01^i 0.99^{100-i} \sum_{j=\lceil 50-50C \rceil^+}^{50} \binom{50}{j} 0.05^j 0.95^{50-j}$$

where

$$C = \frac{\tilde{x} - 1.02(W_0 - X_1 - X_2) - 1.03X_1\frac{100-i}{100}}{1.05X_2}. \tag{3.8}$$

We apply the IS strategy of the previous sections that is we increase the default probabilities of bonds to $p_{new,A}$ and $p_{new,B}$. Our aim is to minimize the variance of the IS estimator to compute the optimal IS probabilities. The resulting minimization problem is

$$\min_{p_{new,A}, p_{new,B}} \sum_{i=0}^{100} \binom{100}{i} \frac{0.01^{2i} 0.99^{200-2i}}{p_{new,A}^i (1-p_{new,A})^{100-i}} \sum_{j=\lceil 50-50C \rceil^+}^{50} \binom{50}{j} * \frac{0.05^{2j} 0.95^{100-2j}}{p_{new,B}^j (1-p_{new,B})^{50-j}}$$

which we solve using the Nelder-Mead Simplex method as implemented in GSL [9].

Table 3.12. Optimal and exponential twisting IS probabilities and corresponding variances to compute $P(W < \tilde{x})$.

$\tilde{x}$	$P(W < \tilde{x})$	OP		ETP		var (O)	var (ET)	var (naive)
100	$5.33E-2$	0.026	0.092	0.023	0.085	$6.22E-7$	$6.56E-7$	$5.04E-6$
99	$6.47E-3$	0.035	0.111	0.032	0.106	$1.39E-8$	$1.43E-8$	$6.43E-7$
98.5	$2.46E-3$	0.039	0.119	0.037	0.115	$2.27E-9$	$2.30E-9$	$2.45E-7$
98.2	$9.87E-4$	0.040	0.131	0.040	0.121	$4.10E-10$	$4.31E-10$	$9.86E-8$
98	$6.68E-4$	0.041	0.136	0.042	0.124	$1.94E-10$	$2.07E-10$	$6.68E-8$

To compute the exponential twisting probabilities $(p_{new,A}, p_{new,B})$, first of all we should declare two variables (c_A, c_B) for the losses. Here, $c_A = X_1(1+0.03)/m_A$ and $c_B = X_2(1+0.05)/m_B$. Then, remember that exponential twisting uses the strategy; if $E[W] > x$ then choose θ such that $E_\theta[W] = x$ otherwise $\theta = 0$ given that $p_{new,A}(\theta) = \frac{p_A e^{\theta c_A}}{p_A e^{\theta c_A} + (1-p_A)}$ and $p_{new,B}(\theta) = \frac{p_B e^{\theta c_B}}{p_B e^{\theta c_B} + (1-p_B)}$.

For $X_1 = 60$, $X_2 = 20$, we compute optimal and exponential twisting probabilities for a series of $\tilde{x}$ values in Table 3.12. Note that variances given in this table are for $10,000$ number of replications for all of the methods.

The second problem is to compute $E[W|W < \tilde{x}]$ which has the analytical solution of

$$\frac{1}{P(W < \tilde{x})} \sum_{i=0}^{100} \binom{100}{i} 0.01^i 0.99^{100-i} \sum_{j=\lceil 50-50C \rceil^+}^{50} \binom{50}{j} 0.05^j 0.95^{50-j} \tilde{w}(i,j)$$

where $\tilde{w}$ is given in (3.7) and C is given in (3.8).

As we have done in previous sections, the asymptotic variance of our IS estimator

Table 3.13. Optimal and exponential twisting IS probabilities and corresponding variances to compute $E[W|W < \tilde{x}]$.

$\tilde{x}$	$E[W\|W < \tilde{x}]$	OP		ETP		var (O)	var (ET)	var (naive)
100	99.47	0.026	0.092	0.023	0.085	2.18	2.30	17.58
99	98.51	0.035	0.111	0.032	0.106	3.23	3.33	149.05
98.5	98.12	0.039	0.119	0.037	0.115	3.64	3.69	390.85
98.2	97.77	0.040	0.131	0.040	0.121	4.04	4.25	967.25
98	97.63	0.041	0.136	0.042	0.124	4.15	4.43	1425.60

is

$$\frac{1}{nP(W<\tilde{x})^2}\left[\sum_{i=0}^{100}\binom{100}{i}\frac{0.01^{2i}0.99^{200-2i}}{p_{new,A}^{i}(1-p_{new,A})^{100-i}}\sum_{j=\lceil 50-50C\rceil^+}^{50}\binom{50}{j}\right.$$
$$*\frac{0.05^{2j}0.95^{100-2j}}{p_{new,B}^{j}(1-p_{new,B})^{50-j}}(\tilde{w}(i,j))^2$$
$$\left.-\left(\sum_{i=0}^{100}\binom{100}{i}0.01^{i}0.99^{100-i}\sum_{j=\lceil 50-50C\rceil^+}^{50}\binom{50}{j}0.05^{j}0.95^{50-j}\tilde{w}(i,j)\right)^2\right]$$

Thus, the minimization problem for the optimal IS probabilities is

$$\min_{p_{new,A},p_{new,B}}\sum_{i=0}^{100}\binom{100}{i}\frac{0.01^{2i}0.99^{200-2i}}{p_{new,A}^{i}(1-p_{new,A})^{100-i}}\sum_{j=\lceil 50-50C\rceil^+}^{50}\binom{50}{j}$$
$$*\frac{0.05^{2j}0.95^{100-2j}}{p_{new,B}^{j}(1-p_{new,B})^{50-j}}(\tilde{w}(i,j))^2$$

which we solve again using the Nelder-Mead Simplex method as implemented in GSL.

For $X_1 = 60$, $X_2 = 20$, we compute optimal and exponential twisting probabilities for a series of $\tilde{x}$ values in Table 3.13. Note that, variances given in this table are for $10,000$ number of replications for all of the methods. Furthermore, we use the same exponential twisting probabilities computed for simulating $P(W < \tilde{x})$. To our knowledge, this is the proposed strategy in related papers for applying exponential twisting for ES. If we compare the optimal probabilities in Tables 3.12 and 3.13, we

see that they are the same indeed. Thus, we see the correctness of this approach by this small example.

3.3.1. Example 2

Our second numerical example is a portfolio consisting of obligors from two rating groups, A and B. We have 1000 obligors ($m_A = m_B = 1000$) in each group and default probabilities are 0.005 and 0.02. However, contrary to our first example exposure levels are not the same in the groups. They have the form given below

$$c_{A,j} = (\lceil 5j/m_A \rceil)^2, j = 1, ..., m_A$$
$$c_{B,j} = (\lceil 5j/m_B \rceil)^2, j = 1, ..., m_B.$$

Possible exposures are $1, 4, 9, 16$, and 25, with 200 obligors at each level. There is no easy analytical solution for tail loss probability and expected shortfall in this case. Our aim is again to compare optimal and exponential twisting probabilities and corresponding variances but this time we should use simulation to compute them. We use GSL to solve the 10-dimensional (10 groups of obligors) optimization problem for optimal IS and the 1-dimensional optimization problem for exponential twisting using $1,000,000$ replications of the simulations. In exponential twisting, we want to compute a θ that will change the default probabilities as in (2.4) so that expectation of the losses under this new measure will be equal to x.

The resultant variances are given in Table 3.14 for $P(L > x)$ and in Table 3.15 for $E[L|L > x]$. Note that the given variances are for $10,000$ replications. The numerical results in this section once again show that exponential twisting probabilities are nearly the same with optimal ones when the number of obligors is high and the event we simulate is the rare.

[33] compares VaR and ES in a simple real-world problem. They conclude that ES should also be used for financial risk management. However, they complain about

Table 3.14. Variances for simulating $P(L > x)$ under IS using optimal and exponential twisting probabilities and naive simulation.

x	$P(L > x)$	var (O)	var (ET)	var (naive)
400	$4.33E-2$	$4.05E-7$	$4.22E-7$	$4.13E-6$
450	$9.88E-3$	$2.93E-8$	$3.07E-8$	$1.04E-6$
500	$1.81E-3$	$1.21E-9$	$1.27E-9$	$1.90E-7$
550	$2.67E-4$	$3.01E-11$	$3.01E-11$	$2.00E-8$

Table 3.15. Variances for simulating $E[L|L > x]$ under IS using optimal and exponential twisting probabilities and naive simulation.

x	$E[L\|L > x]$	var (O)	var (ET)	var (naive)
400	433.74	36.83	39.99	410.95
450	479.68	63.61	64.85	2626.26
500	526.49	98.57	99.11	13253.54
550	573.72	138.29	139.96	434281.00

the requirement of increased sample sizes (compared to VaR) when computing ES. Different to that comment of [33], we have shown that IS can be easily applied to compute the expected shortfall.

4. A NEW ALGORITHM FOR THE NORMAL COPULA MODEL

In this chapter we propose a new efficient simulation method for computing tail loss probabilities and conditional expectations in the normal copula framework. We replace inner IS (exponential twisting) by inner replications implemented by a geometric shortcut.

It is important to estimate tail loss probabilities and conditional expectations for VaR and ES. We first discuss the problem of simulating tail loss probabilities and conditional expectations for single loss and value at risk values. In Section 4.5 we consider the problem of simulating tail loss probabilities and conditional expectations for several loss and value at risk values simultaneously.

4.1. Geometric Shortcut: Independent Obligors

We first consider the case of independent obligors (i.e. set all $a_{jl} = 0$). This is useful to clarify how we can improve naive simulation for computing tail probabilities. The naive algorithm for tail loss probability computation is given in Figure 4.1.

---

1. Repeat for replications $k = 1, ..., n$:
   1. generate $Y_j, j = 1, ..., m$, from $p_j(z)$;
   2. calculate total loss for replication k, $L^{(k)} = \sum_{j=1}^{m} c_j Y_j$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}$.

Figure 4.1. Tail loss probability computation using naive simulation for independent obligors.

We use the default probabilities to generate default indicators in step 1 of the algorithm given in Figure 4.1. We can always decrease the variance of $P(L > x)$ by increasing the number of replications without any variance reduction technique.

However, before increasing the number of replications, we should make the simulation fast so that increasing the number of replications does not change the cost of naive simulation too much. To accomplish this, observe that we only use the loss values of replications in the output. Thus, we don't need to generate each default indicator. Instead, it is enough to know in which replications defaults occurred for each obligor. So, for a large number of replications, e.g. $n = 1000$, and small values of p_j it is sensible to use the geometric distribution to generate the defaults for each obligor. Whenever we get a default for obligor j, we increase the loss value ($L^{(k)}$) for that replication by the loss level of obligor j. That is, we change the direction of the simulation. Instead of simulating the loss repetition by repetition we simulate the defaults of n repetitions obligor by obligor. A visualization of this idea is given in Figure 4.2. Algorithm given in Figure 4.3 contains the details of the "geometric shortcut" idea.

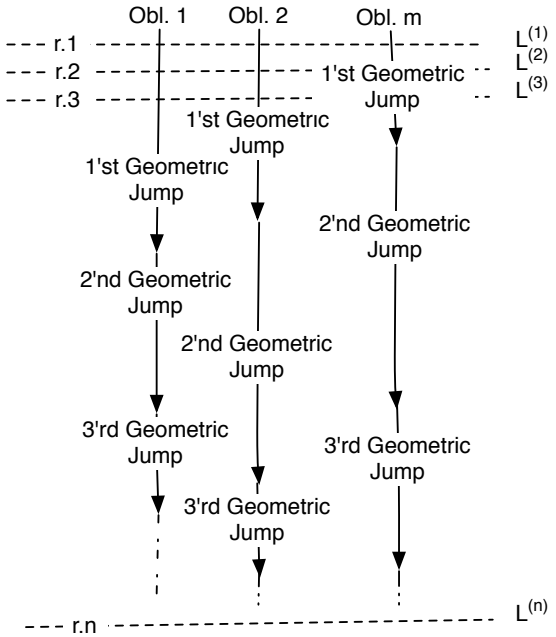

Figure 4.2. Geometric shortcut in generating default indicators

Let's compute the speed up by comparing the required number of uniform random numbers in the algorithms given in Figures 4.1 and 4.3 under the assumption that generating geometric random variate has the same cost as generating uniform random variates. Obviously $1 + \lfloor np_j \rfloor$ is the expected number of uniform random numbers required for obligor j in the algorithm given in Figure 4.3. Moreover, if we use the

1. Repeat for obligors $j = 1, ..., m$:
   1. initialize s to zero;
   2. repeat until $s > n$:
      (a) make a geometric jump for obligor j under p_j to update position of last default, s using $s = s +$ geometric random variate (R.V.);
      (b) $L^{(s)} = L^{(s)} + c_j$;
2. Return $\frac{1}{n}\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)}>x\}}$.

Figure 4.3. Tail loss probability simulation using the geometric shortcut for independent obligors.

generated last uniform random number (not used for locating the default) for obligor j in finding the first default for obligor $j+1$, $1+\sum_{j=1}^{m}\lfloor np_j \rfloor \approx \sum_{j=1}^{m}\lfloor np_j \rfloor$ is the required expected total number of uniform random numbers in the algorithm given in Figure 4.3. Since, the required number of uniform random numbers is equivalent to nm for the algorithm given in Figure 4.1, the speed up ratio is

$$r_s = \frac{nm}{\sum_{j=1}^{m}\lfloor np_j \rfloor} \geq \frac{1}{\bar{p}} \tag{4.1}$$

where $\bar{p}$ is the average of the default probabilities. This means that we can increase the number of replications of the algorithm given in Figure 4.3 to a multiple of r_s without increasing the simulation time compared to the algorithm given in Figure 4.1. Thus the algorithm given in Figure 4.3 with nr_s replications would reduce the variance to the $1/r_s$ times the variance of the algorithm given in Figure 4.1 without increasing the execution time. This is why we could interpret "geometric shortcut" as a variance reduction technique.

4.2. Inner Replications using Geometric Shortcut: Dependent Obligors

We consider the more realistic problem of dependent obligors in this section. Dependence across obligors does not totally change the structure of the problem in section 4.1. Since, conditional on $Z = z$, obligors default independently as defined in

(2.2).

1. Repeat for replications $k = 1, ..., n$: /* Outer replications */
   1. generate $z_l \sim N(0,1), l = 1, ..., d$, independently;
   2. calculate $p_j(z), j = 1, ..., m$, as in (2.2) where $z = (z_1, ..., z_d)$;
   3. generate $Y_j, j = 1, ..., m$, from $p_j(z)$; /* Inner replications of size 1 */
   4. calculate total loss for replication k, $L^{(k)} = \sum_{j=1}^{m} c_j Y_j$;
2. Return $\frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} > x\}}$.

Figure 4.4. Tail loss probability computation using naive simulation for dependent obligors.

Let us first give the naive simulation algorithm in Figure 4.4 to better explain the possible improvement over it. We have to add two steps to the the independent obligors case of the algorithm given in Figure 4.1: generating common risk factors and calculating the conditional default probabilities. We would like to make use of the geometric distribution to generate the default indicators for each obligor. However, we can not proceed with the same p_j values as, for each replication we have different conditional default probabilities for each obligor. Thus, the only way we can use the geometric shortcut is by increasing the number of inner replications. As a matter of fact, it seems sensible to use the conditional default probabilities more than once as their computation is quite expensive. We give a visualization of our algorithm in Figure 4.5 and the algorithm in Figure 4.6.

In the previous section for independent obligors we have calculated the speed up ratio in (4.1). If we compute the same ratio, this time comparing the algorithms given in Figures 4.4 and 4.6, for one outer replication, we get

$$\frac{m}{\sum_{j=1}^{m} \lfloor n_{in} p_j(z) \rfloor} \geq \frac{m}{\sum_{j=1}^{m} n_{in} p_j(z)} = \frac{1}{n_{in} \bar{p}_Z} \geq \frac{1}{n_{in}} \left\lfloor \frac{1}{\bar{p}_Z} \right\rfloor$$

where $\bar{p}_Z$ is the average of the conditional default probabilities for obligors. We don't want to be much slower than the naive simulation so this ratio should be greater than

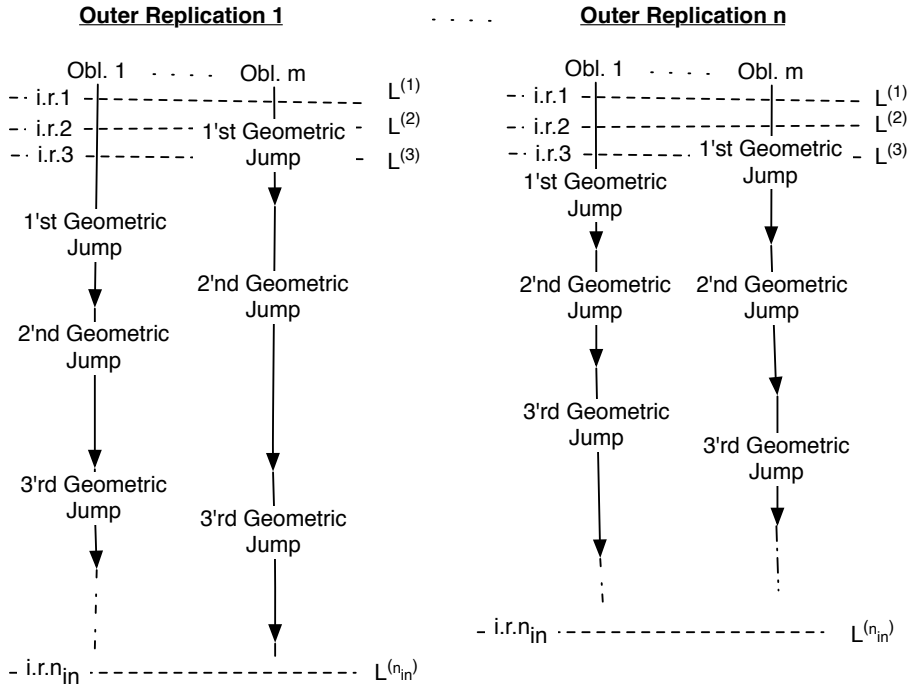

Figure 4.5. Inner replications using geometric shortcut in generating default indicators

1. Thus,

$$n_{in} \leq \left\lfloor \frac{1}{\bar{p}_Z} \right\rfloor .$$

Up to now we compare inner replications of the algorithms given in Figures 4.4 and 4.6 (step (3) of the algorithm given in Figure 4.4 and step (4) of the algorithm given in Figure 4.6). But, n_{in} is also used in step 5 of the algorithm given in Figure 4.6. And, this step should not take much longer than calculating conditional default probabilities (common step for both algorithms). Therefore, we should select n_{in} such that $n_{in} \leq m$. When we combine both constraints, we get $n_{in} = min(\lfloor 1/\bar{p}_Z \rfloor, m)$.

1. Repeat for replications $k = 1, ..., n$: /* Outer replications */
   1. generate $z_l \sim N(0,1), l = 1, ..., d$, independently;
   2. calculate $p_j(z), j = 1, ..., m$, as in (2.2) where $z = (z_1, ..., z_d)$;
   3. construct a loss vector $L_{(in)}$ of size $n_{in} = min(\lfloor 1/\bar{p}_Z \rfloor, m)$;
   4. repeat for obligors $j = 1, ..., m$; /* Inner replications */
      (a) initialize s to zero;
      (b) repeat until $s > n$;
         (I) make a geometric jump for obligor j under $p_j(z)$ using $s = s +$ geometric R.V.; ;
         (II) $L^{(s)}_{(in)} = L^{(s)}_{(in)} + c_j$;
   5. compute $p^{(k)}_{in} = \frac{1}{n_{in}} \sum_{t=1}^{n_{in}} \mathbf{1}_{\{L^{(t)}_{(in)} > x\}}$ where $p^{(k)}_{in}$ denotes the loss probability of replication k;
2. Return $\frac{1}{n} \sum_{k=1}^{n} p^{(k)}_{in}$.

Figure 4.6. Tail loss probability simulation using inner replications using the geometric shortcut for dependent obligors.

4.3. Integrating IS with Inner Replications using the Geometric Shortcut: Dependent Obligors

The implementation of inner replications using the geometric shortcut alone is not sufficient to decrease the variance for highly dependent obligors. Because the main source of variance is the outer simulation not the inner one for highly dependent obligors. Thus we need an IS strategy on Z to increase the conditional default probabilities. It is easy to use a multi-normal IS density with different means (μ_l) and standard deviations (σ_l) for $l = 1, ..., d$. However, just shifting the mean vector could be sufficient for the normal copula framework (see [14]). As mean shift we use the mode of the zero-variance IS distribution (see Chapter 5).

Finding the mode of the zero-variance IS distribution requires the solution of the multidimensional optimization problem;

$$\max_z P(L > x|Z = z)e^{-z^T z/2}.$$

We use the normal approximation for $P(L > x|Z)$ because it is fast and reliable in computing mean shifts compared to other methods (see Chapter 5). Since, $E[L|Z = z] = \sum_{j=1}^{m} c_j p_j(z)$ and $Var[L|Z = z] = \sum_{j=1}^{m} c_j^2 [p_j(z) - p_j(z)^2]$, we can use the normal approximation:

$$P\left(L > x|Z = z\right) \approx 1 - \Phi\left(\frac{x - E[L|Z = z]}{\sqrt{Var[L|Z = z]}}\right).$$

Thus, we have to solve the optimization problem

$$\max_z \left[1 - \Phi\left(\frac{x - E[L|Z = z]}{\sqrt{Var[L|Z = z]}}\right)\right] e^{-z^T z/2}. \tag{4.2}$$

After selecting a new mean μ for Z, we combine IS with inner replications using the

geometric shortcut. Only two steps of the algorithm given in Figure 4.6 are affected by this integration. In step (1) of the algorithm given in Figure 4.6, instead of generating z_l from $N(0,1)$ we generate it from $N(\mu_l,1)$ where μ_l is the component of vector μ in dimension l. And, in step (5), we have to multiply the loss probability of the inner replication k with the likelihood ratio

$$w_\mu = e^{-\mu^T Z + \mu^T \mu/2}, \tag{4.3}$$

which relates the density of the $N(0,I)$ distribution to that of the $N(\mu,I)$ distribution where I is the identity matrix. The full algorithm is given below for the sake of completeness.

---

1. Compute μ using (4.2).
2. Repeat for replications $k = 1, ..., n$: /* Outer replications */
   1. generate $z_l \sim N(\mu_l, 1), l = 1, ..., d$, independently;
   2. calculate w_μ as in (4.3);
   3. calculate $p_j(z), j = 1, ..., m$, as in (2.2) where $z = (z_1, ..., z_d)$;
   4. construct a loss vector $L_{(in)}$ of size $n_{in} = min(\lfloor 1/\bar{p}_Z \rfloor, m)$;
   5. repeat for obligors $j = 1, ..., m$; /* Inner replications */
      (a) initialize s to zero;
      (b) repeat until $s > n$;
         (I) make a geometric jump for obligor j under $p_j(z)$ to update $s = s +$ R.V.;
         (II) $L_{(in)}^{(s)} = L_{(in)}^{(s)} + c_j$;
   6. compute $p_{in}^{(k)} = \frac{w_\mu}{n_{in}} \sum_{t=1}^{n_{in}} \mathbf{1}_{\{L_{(in)}^{(t)} > x\}}$. where $p_{in}^{(k)}$ stands for the loss probability of replication k;
3. Return $\frac{1}{n} \sum_{k=1}^{n} p_{in}^{(k)}$.

Figure 4.7. Tail loss probability simulation using integration of IS with inner replications using the geometric shortcut for dependent obligors.

4.4. Computing Conditional Expectation of Loss: Dependent Obligors

We described our new methodology for tail loss probability computation. It is time to explain how the same methodology can be used for the computation of conditional expectations. We consider only the integration of IS with inner replications using the geometric shortcut for dependent obligors.

If we assume that $P(L \geq \text{VaR}_\alpha) > 0$, ES can be represented as

$$r = E[L|L \geq \text{VaR}_\alpha] = \frac{E\left[L\mathbf{1}_{\{L \geq \text{VaR}_\alpha\}}\right]}{P(L \geq \text{VaR}_\alpha)}.$$

The naive simulation estimate for this ratio is

$$\hat{r}^{naive} = \frac{\sum_{k=1}^{n} L^{(k)}\mathbf{1}_{\{L^{(k)} \geq \text{VaR}_\alpha\}}}{\sum_{k=1}^{n} \mathbf{1}_{\{L^{(k)} \geq \text{VaR}_\alpha\}}}. \tag{4.4}$$

See the algorithm using this estimate for ES given below.

1. Repeat for replications $k = 1, ..., n$: /* Outer replications */
   1. generate $z_l \sim N(0,1), l = 1, ..., d$, independently;
   2. calculate $p_j(z), j = 1, ..., m$, as in (2.2) where $z = (z_1, ..., z_d)$;
   3. generate $Y_j, j = 1, ..., m$, from $p_j(z)$; /* Inner replications of size 1 */
   4. calculate total loss for replication k, $L^{(k)} = \sum_{j=1}^{m} c_j Y_j$;
2. Return $\hat{r}^{naive}$.

Figure 4.8. Naive simulation for computing ES for dependent obligors.

[12] develops IS estimates for computing conditional expectation contributions of obligors given that large losses occurs for the portfolio. We outline [12] before discussing our new method for ES but describe the IS estimate of the conditional expectation of the full portfolio instead of contributions of obligors. If $f(z)$ is our distribution to generate the z vector and $g(z)$ is the new IS distribution for this purpose, $w = \frac{f(z)}{g(z)}$

(equal to (4.3) for our problem) is the likelihood ratio for the generated loss, L. Then

$$r = \frac{\tilde{E}\left[Lw\mathbf{1}_{\{L\geq \mathrm{VaR}_\alpha\}}\right]}{\tilde{E}[w\mathbf{1}_{\{L\geq \mathrm{VaR}_\alpha\}}]} \tag{4.5}$$

where $\tilde{E}$ denotes the expectation under the new measure, which means generating z vector from $g(z)$ instead of $f(z)$ for the simulation estimate.

If $L^{(k)}$ and $w^{(k)}$ denotes the loss and likelihood ratio of the kth replication of a simulation study in the given order then IS estimator of ES is

$$\hat{r}^{IS} = \frac{\sum_{k=1}^{n} L^{(k)}w^{(k)}\mathbf{1}_{\{L^{(k)}\geq \mathrm{VaR}_\alpha\}}}{\sum_{k=1}^{n} w^{(k)}\mathbf{1}_{\{L^{(k)}\geq \mathrm{VaR}_\alpha\}}}. \tag{4.6}$$

To compute the precision of (4.6), [12] proposes to use a confidence interval

$$\hat{r}^{IS} \pm z_{\delta/2}\frac{\hat{\sigma}^{IS}}{\sqrt{n}} \tag{4.7}$$

where

$$\hat{\sigma}^{IS} = \left(\frac{n\sum_{k=1}^{n}\left(L^{(k)}w^{(k)} - \hat{r}^{IS}w^{(k)}\right)^2\mathbf{1}_{\{L^{(k)}\geq \mathrm{VaR}_\alpha\}}}{\left(\sum_{k=1}^{n} w^{(k)}\mathbf{1}_{\{L^{(k)}\geq \mathrm{VaR}_\alpha\}}\right)^2}\right)^{1/2} \tag{4.8}$$

and $z_{\delta/2}$ is the $(1-\delta/2)$ percentile of the standard normal distribution. This confidence interval is given under the conditions; $P(L \geq \mathrm{VaR}_\alpha) > 0$, second moments of the expectations given in (4.5) are bounded and taking the ratio zero whenever the denominator is zero in (4.8).

Since, we have IS in our method, we should use a variant of the approach given above. Inner replications using the geometric shortcut give new loss values in the number of inner replications per replication. Thus, taking the average of loss values in an inner replication and then using this value $\bar{L}^{(k)}$ in (4.6) will be our new estimate for

ES. Thus,

$$\hat{r}^{new} = \frac{\sum_{k=1}^{n} \bar{L}^{(k)} w^{(k)} \mathbf{1}_{\{\bar{L}^{(k)} \geq \mathrm{VaR}_\alpha\}}}{\sum_{k=1}^{n} w^{(k)} \mathbf{1}_{\{\bar{L}^{(k)} \geq \mathrm{VaR}_\alpha\}}}. \tag{4.9}$$

In the same manner, we can use the confidence interval given in (4.7) by just replacing $L^{(k)}$ with $\bar{L}^{(k)}$. See the algorithm given in Figure 4.9.

---

1. Compute μ using (4.2).
2. Repeat for replications $k = 1, ..., n$: /* Outer replications */
   1. generate $z_l \sim N(\mu_l, 1), l = 1, ..., d$, independently;
   2. calculate $w^{(k)}$ as in (4.3);
   3. calculate $p_j(z), j = 1, ..., m$, as in (2.2) where $z = (z_1, ..., z_d)$;
   4. construct a loss vector L_{in} of size $n_{in} = min(\lfloor 1/\bar{p}_Z \rfloor, m)$;
   5. repeat for obligors $j = 1, ..., m$; /* Inner replications */
      (a) initialize s to zero;
      (b) repeat until $s > n$;
         (I) make a geometric jump for obligor j under $p_j(z)$ to update $s = s +$ R.V.;
         (II) $L_{(in)}^{(s)} = L_{(in)}^{(s)} + c_j$;
   6. calculate $\bar{L}^{(k)} = \frac{1}{n_{in}} \sum_{t=1}^{n_{in}} L_{in}^{(t)}$;
3. Return $\hat{r}^{new}$.

Figure 4.9. ES simulation using integration of IS with inner replications using the geometric shortcut for dependent obligors.

4.5. Numerical Results

In this section, we evaluate the performance of our new method for the numerical examples of [13]. We compare our new approach with naive Monte Carlo simulation and the two-step IS method of [13] and [12] (both abbreviated as IS in the tables). We give two different tables for the numerical examples. While, the first table compares methods according to the size of the half lengths of the confidence intervals for point estimates of tail loss probabilities, the second table does the same for point estimates

of the ES. We also give execution times of the methods for a better comparison. Probability and half length of the 95 percent confidence interval are abbreviated as prob. and hl in the tables. In the final column of the tables, we supply the variance ratio (VR); variance of IS divided by variance of the new method for easy comparison. See Appendix B.2 for the C codes.

The normal approximation method for computing mean shifts produces very similar results with the tail bound approximation given in [13]. Thus, to compare solely inner IS (exponential twisting) with inner replications using the geometric shortcut, we use normal approximation to compute the optimal shifts for both methods in the numerical results. Note that, the computation of the θ parameter, for twisting the default probabilities in [13] is a tedious job. It is an optimization problem and computation of accurate θ values decreases the variance but increases the execution time. We spent some time to reach a fast and sufficiently precise version of the Newton method. Our first implementation was 30 percent slower.

The first example of [13] is a portfolio of $m = 1000$ obligors in a 10-factor model. The marginal default probabilities and exposures have the following form:

$$p_j = 0.01(1 + \sin(16\Pi j/m)), j = 1, ..., m;$$
$$c_j = (\lceil 5j/m \rceil)^2, j = 1, ..., m.$$

Thus, the marginal default probabilities vary between 0 percent and 2 percent with a mean of 1 percent, and the possible exposures are $1, 4, 9, 16$, and 25, with 200 obligors at each level. These parameters represent a significant departure from an homogeneous model with constant p_j.

For the factor loadings, the a_{jl} are generated independently and uniformly from the interval $(0, 1/\sqrt{d}), d = 10$; the upper limit of this interval ensures that the sum of squared entries $a_{j1}^2, ..., a_{jd}^2$ for each obligor does not exceed 1. We simulate the tail loss probability and expected shortfall for a series of x and VaR_α values in Tables 4.1 and

4.2. Half lengths of the confidence intervals for the simulations and execution times of the methods are given in the tables.

Table 4.1. Tail loss probabilities and half lengths (hl) of the confidence intervals for naive, IS and the new method in the 10-factor model. $n = 10,000$. Execution times (in seconds) are in parentheses.

x	prob. (naive)	hl (naive)	prob. (IS)	hl (IS)	prob. (new)	hl (new)	VR
500	$3.78 * 10^{-2}(6)$	$3.73 * 10^{-3}$	$3.80 * 10^{-2}(15)$	$1.15 * 10^{-3}$	$3.81 * 10^{-3}(11)$	$8.66 * 10^{-4}$	1.75
1,000	$9.40 * 10^{-3}(7)$	$1.89 * 10^{-3}$	$8.45 * 10^{-3}(16)$	$2.90 * 10^{-4}$	$8.55 * 10^{-3}(11)$	$2.29 * 10^{-4}$	1.6
2,000	$9.00 * 10^{-4}(7)$	$5.88 * 10^{-4}$	$8.21 * 10^{-4}(16)$	$2.93 * 10^{-5}$	$8.54 * 10^{-4}(11)$	$2.69 * 10^{-5}$	1.19
3,000	$1.00 * 10^{-4}(7)$	$1.96 * 10^{-4}$	$1.02 * 10^{-4}(16)$	$3.92 * 10^{-6}$	$1.07 * 10^{-4}(11)$	$3.79 * 10^{-6}$	1.07
4,000	-	-	$1.33 * 10^{-5}(16)$	$5.53 * 10^{-7}$	$1.29 * 10^{-5}(11)$	$5.25 * 10^{-7}$	1.10
5,000	-	-	$1.36 * 10^{-6}(16)$	$6.51 * 10^{-8}$	$1.42 * 10^{-6}(10)$	$6.07 * 10^{-8}$	1.15

Table 4.2. ES values and half lengths (hl) of the confidence intervals using naive, IS and the new method in the 10-factor model. $n = 100,000$. Execution times (in seconds) are in parentheses.

VaR_α	ES (naive)	hl	ES (IS)	hl	ES (new)	hl	VR
500	842.7(67)	12.5	841.1(147)	3.0	845.5(109)	2.9	1.07
1,000	1,413.8(66)	29.9	1,419.2(150)	3.9	1,420.2(106)	3.8	1.05
2,000	2,468.7(66)	98.5	2,478.5(154)	4.3	2,475.4(103)	4.6	0.87
3,000	3,471.9(66)	172.2	3,476.3(145)	4.4	3,476.1(101)	4.8	0.84
4,000	4,744.5(65)	746.3	4,452.9(153)	4.5	4,444.4(98)	4.8	0.88
5,000	-	-	5,403.1(155)	4.5	5,397.7(97)	4.6	0.96

The next example is a 21-factor model, again with 1,000 obligors. The marginal default probabilities fluctuate as in the first example, and the exposures c_i increases from 1 to 100 linearly as i increases from 1 to 1,000. The matrix of the factor loadings, $A = (a_{jl}, j = 1, ..., 1000, l = 1, ..., 21)$, has the following block structure:

$$A = \left(\begin{array}{c|ccc|c} & F & & & G \\ R & & \ddots & & \vdots \\ & & & F & G \end{array}\right), \quad G = \begin{pmatrix} g & & \\ & \ddots & \\ & & g \end{pmatrix},$$

Table 4.3. Tail loss probabilities and half lengths (hl) of the confidence intervals for naive, IS and the new method in the 21-factor model. $n = 10,000$. Execution times (in seconds) are in parentheses.

x	prob. (naive)	hl	prob. (IS)	hl	prob. (new)	hl	VR
$2,500$	$5.20 * 10^{-2}(8)$	$4.36 * 10^{-3}$	$5.09 * 10^{-2}(17)$	$1.73 * 10^{-3}$	$5.03 * 10^{-2}(15)$	$1.57 * 10^{-3}$	1.22
$10,000$	$1.09 * 10^{-2}(8)$	$2.04 * 10^{-3}$	$1.11 * 10^{-2}(21)$	$4.17 * 10^{-4}$	$1.12 * 10^{-2}(16)$	$4.02 * 10^{-4}$	1.08
$20,000$	$2.30 * 10^{-3}(8)$	$9.39 * 10^{-4}$	$2.71 * 10^{-3}(20)$	$9.68 * 10^{-5}$	$2.77 * 10^{-3}(15)$	$9.74 * 10^{-5}$	0.99
$30,000$	$8.00 * 10^{-4}(8)$	$5.53 * 10^{-4}$	$6.26 * 10^{-4}(20)$	$2.43 * 10^{-5}$	$6.37 * 10^{-4}(13)$	$2.47 * 10^{-5}$	0.97
$40,000$	-	-	$7.47 * 10^{-5}(21)$	$3.45 * 10^{-6}$	$7.36 * 10^{-5}(13)$	$3.35 * 10^{-6}$	1.04

with R a column vector of $1,000$ entries all equal to 0.8; F, a column vector of 100 entries all equal to 0.4; G a 100×10 matrix, and g, a column vector of 10 entries, all equal to 0.4. We simulate the tail loss probability and expected shortfall for a series of x and VaR_α values in Tables 4.3 and 4.4.

Table 4.4. ES values and half lengths (hl) of the confidence intervals using naive, IS and the new method in the 21-factor model. $n = 250,000$. Execution times (in seconds) are in parentheses.

VaR_α	ES (naive)	hl	ES (IS)	hl	ES (new)	hl	VR
$2,500$	$7,614.7(208)$	107.2	$7,584.6(349)$	31.2	$7,580.3(307)$	31.2	1.0
$10,000$	$16,782.9(209)$	237.9	$16,798.3(400)$	41.5	$16,827.5(311)$	41.8	0.99
$20,000$	$26,195.7(209)$	430.8	$26,395.6(442)$	36.4	$26,422.9(302)$	36.5	0.99
$30,000$	$34,637.9(209)$	569.1	$34,831.1(465)$	28.4	$34,832.9(296)$	28.6	0.99
$40,000$	42,379.5(208)	959.0	$42,590.1(477)$	16.9	$42,597.5(287)$	17.0	0.99

We defined the third numerical example ourselves. It is a 5-factor model with 4800 obligors. Obligors are separated into 6 segments of size 800 each as seen in Table 4.5. Default probabilities, exposure levels and factor loadings are the same in each segment for the obligors. We again simulate the tail loss probability and expected shortfall for a series of x and VaR_α values in Tables 4.6 and 4.7.

Table 4.5. Portfolio composition for the 5-factor model; default probabilities, exposure levels and factor loadings for six segments.

Segment	Obligor j	p_j	c_j	$a_{j,1}$	$a_{j,2}$	$a_{j,3}$	$a_{j,4}$	$a_{j,5}$
$1A$	$1-800$	0.01	20	0.7	0.5	0.1		
$1B$	$801-1600$	0.02	10	0.7	0.5	0.1		
$2A$	$1601-2400$	0.02	10	0.7		0.2	0.4	
$2B$	$2401-3200$	0.04	5	0.7		0.2	0.4	
$3A$	$3201-4000$	0.03	5	0.7			0.4	0.5
$3B$	$4001-4800$	0.05	1	0.7			0.4	0.5

Table 4.6. Tail loss probabilities and half lengths (hl) of the confidence intervals for naive, IS and the new method in the 5-factor model. $n = 10,000$. Execution times (in seconds) are in parentheses.

x	prob. (naive)	hl	prob. (IS)	hl	prob. (new)	hl	VR
$5,000$	$4.66*10^{-2}(25)$	$4.13*10^{-3}$	$4.55*10^{-2}(52)$	$1.52*10^{-3}$	$4.55*10^{-3}(37)$	$1.46*10^{-3}$	1.09
$10,000$	$2.02*10^{-2}(24)$	$2.76*10^{-3}$	$1.94*10^{-2}(53)$	$6.96*10^{-4}$	$1.81*10^{-2}(37)$	$5.97*10^{-4}$	1.36
$15,000$	$8.00*10^{-3}(24)$	$1.75*10^{-3}$	$8.50*10^{-3}(54)$	$2.95*10^{-4}$	$8.36*10^{-3}(37)$	$2.84*10^{-4}$	1.08
$20,000$	$5.30*10^{-3}(24)$	$1.42*10^{-3}$	$3.87*10^{-3}(57)$	$1.36*10^{-4}$	$4.03*10^{-3}(36)$	$1.39*10^{-4}$	0.95
$25,000$	$2.40*10^{-3}(24)$	$9.59*10^{-4}$	$1.83*10^{-3}(57)$	$6.69*10^{-5}$	$1.81*10^{-3}(37)$	$6.60*10^{-5}$	1.03
$30,000$	$1.10*10^{-3}(24)$	$6.50*10^{-4}$	$7.80*10^{-4}(59)$	$3.05*10^{-5}$	$8.13*10^{-4}(36)$	$3.11*10^{-5}$	0.96

Table 4.7. ES values and half lengths (hl) of the confidence intervals using naive, IS and the new method in the 5-factor model. $n = 100,000$. Execution times (in seconds) are in parentheses.

VaR_α	ES (naive)	hl	ES (IS)	hl	ES (new)	hl	VR
$5,000$	$10,619.2(239)$	169.8	$10,730.3(500)$	52.4	$10,742.4(368)$	55.0	0.91
$10,000$	$16,167.4(240)$	265.8	$16,295.9(517)$	57.3	$16,225.6(367)$	57.7	0.98
$15,000$	$21,164.6(241)$	358.1	$21,307.0(534)$	56.4	$21,279.0(367)$	56.2	1.00
$20,000$	$26,100.9(239)$	512.4	$25,836.0(543)$	50.8	$25,784.9(365)$	51.1	0.99
$25,000$	29,989.0(242)	493.5	$29,989.7(559)$	43.4	$29,959.6(367)$	43.1	1.01
$30,000$	$33,661,6(241)$	601.7	$33,748.1(573)$	33.4	$33,764.4(368)$	34.0	0.96

We did some numerical experiments to see how Tables 4.1−4.4 and 4.6−4.7 are affected if we change the number of inner replications in the new method. They clearly showed that the main source of variation in the simulation is the outer simulation. We have arrived at this conclusion since increasing the number of inner replications does not reduce the variance any more. Our choice for the number of inner replications used in the algorithms is obviously a good approximation for the optimal number of inner replications. Thus, we can report that both of the methods as the inner simulations (inner replications using the geometric shortcut or exponential twisting) most of the time reduce the variance as much as they could. But, we can add some comments; the new method performs better when calculating larger loss probabilities for out of tail cases and the new method is approximately one and a half times faster than IS in all cases.

4.5.1. Simultaneous Simulations

We separately simulate tail loss probabilities (ES) for all x (VaR_α) in producing Tables 4.1 and 4.3 (4.2 and 4.4). However, in practice people are interested in the tail loss probability distribution to assess VaR. In the same manner, we want to see how expected shortfall changes with respect to VaR_α. Thus, it is of greatest practical importance to compute tail loss probabilities and ES for a wide range of x and VaR_α values in a single simulation. So, as the next step we compare these methods in achieving loss probabilities (ES) simultaneously over a wide range of x (VaR_α) values.

IS strategy of [13] ([12]) fixes optimal mean shift and twisted default probabilities on x (VaR_α). This makes a single simulation of IS at more than one point difficult. But, in section 6 of [13] it is suggested to use the IS distribution computed at the smallest loss level for all values we need. Thus, in Table 4.8 (4.9), we apply the IS distribution computed at $x = 500$ ($\text{VaR}_\alpha = 500$) for the rest of the x (VaR_α) for IS. Contrary to IS of [13] ([12]), inner replications using the geometric shortcut do not have any parameter that is related to x (VaR_α). This means that there is no problem to simultaneously apply the geometric shortcut for a wide range of x values (VaR_α). However, the new method is an integration of outer IS and the geometric

shortcut. Thus, we should change the implementation of outer IS to make the new method simultaneously applicable. One simple way could be first of all computing optimal mean shifts for the minimum and maximum x values (VaR_α) then using a density mixture of the normal distributions having means of these shifts and variances 1 to get optimal mean shifts for simulations. Similarly, instead of sampling from this density mixture, we can use a normal distribution having the mean and variance of this density mixture to get samples for the simulation. In Tables 4.8 and 4.9, we use that normal distribution having the mean and variance of the density mixture of equal weights as (outer) IS-density.

Combining outer IS described above and the geometric shortcut for inner replications performs much better than IS applied by [13] as can be seen from Tables 4.8 and 4.9. Only for the smallest x it is a little bit worse than IS as by the design of outer IS the suggestion of [13] is best for the minimal x value and our normal mixture approach is worse for small x values. Note that the improvement of our method comes both from the usage of outer IS with a suitable larger variance and from the usage of the geometric shortcut which works very well for all x-values considered. Numerical results for the second example are similar to Tables 4.8 and 4.9, so we do not include them here. Also note that our new method is about 30 percent and 50 percent faster. Thus the actual gain is 30 percent and 50 percent higher than the Variance Reduction factors given in Tables 4.8 and 4.9.

To visiualize what we have observed in Tables 4.8 and 4.9, we have drawn Figures 4.10 and 4.11. Simulations are based on $1,000$ and $10,000$ replications in the given order for the figures. In each case, the three curves show the sample mean (the center line) and a 95 percent confidence interval (the two outer lines) computed separately at each point. Note that, while solid lines refer to sample mean and confidence interval of the new method, dashed lines refer to sample mean and confidence interval of IS.

Table 4.8. Tail loss probabilities and half lengths (hl) of the confidence intervals in a single simulation using naive, IS and the new method in the 10-factor model. $n = 10,000$. Execution times for the methods are 7, 16 and 12 seconds in the given order.

x	prob. (naive)	hl	prob. (IS)	hl	prob. (new)	hl	VR
500	$3.90 * 10^{-2}$	$3.79 * 10^{-3}$	$3.90 * 10^{-2}$	$1.18 * 10^{-3}$	$3.97 * 10^{-2}$	$1.73 * 10^{-3}$	0.47
1,000	$8.00 * 10^{-3}$	$1.75 * 10^{-3}$	$8.35 * 10^{-3}$	$3.16 * 10^{-4}$	$8.69 * 10^{-3}$	$3.09 * 10^{-4}$	1.05
1,500	$2.80 * 10^{-3}$	$1.03 * 10^{-3}$	$2.42 * 10^{-3}$	$1.17 * 10^{-4}$	$2.60 * 10^{-3}$	$8.73 * 10^{-5}$	1.79
2,000	$1.20 * 10^{-3}$	$6.80 * 10^{-4}$	$8.29 * 10^{-4}$	$3.76 * 10^{-5}$	$8.57 * 10^{-4}$	$2.99 * 10^{-5}$	1.58
2,500	$2.00 * 10^{-4}$	$2.77 * 10^{-4}$	$2.84 * 10^{-4}$	$2.24 * 10^{-5}$	$3.00 * 10^{-4}$	$1.12 * 10^{-5}$	3.99
3,000	$2.00 * 10^{-4}$	$2.77 * 10^{-4}$	$9.86 * 10^{-5}$	$1.01 * 10^{-5}$	$1.06 * 10^{-4}$	$4.46 * 10^{-6}$	5.12
3,500	–	–	$3.92 * 10^{-5}$	$5.00 * 10^{-6}$	$3.65 * 10^{-5}$	$1.73 * 10^{-6}$	8.26
4,000	–	–	$1.43 * 10^{-5}$	$2.40 * 10^{-6}$	$1.22 * 10^{-5}$	$6.80 * 10^{-7}$	12.5
4,500	–	–	$4.51 * 10^{-6}$	$1.11 * 10^{-6}$	$4.32 * 10^{-6}$	$2.73 * 10^{-7}$	16.39
5,000	–	–	$1.08 * 10^{-6}$	$4.17 * 10^{-7}$	$1.39 * 10^{-6}$	$1.02 * 10^{-7}$	16.74

Table 4.9. ES values and half lengths (hl) of the confidence intervals in a single simulation using naive, IS and the new method in the 10-factor model. $n = 100,000$. Execution times for the methods are 66, 151 and 104 seconds in the given order.

VaR_α	ES (naive)	hl	ES (IS)	hl	ES (new)	hl	VR
500	839.8	12.3	840.8	3.0	840.6	4.5	0.44
1,000	1,440.4	32.9	1,420.9	4.2	1,419.2	4.5	0.87
1,500	1,914.0	54.4	1,960.4	5.5	1,961.2	4.7	1.37
2,000	2,429.4	86.7	2,473.8	7.1	2,477.7	4.9	2.10
2,500	3,135.1	196.8	2,977.9	9.0	2,978.7	5.2	3.00
3,000	3,602.4	267.7	3,470.6	11.5	3,477.3	5.5	4.37
3,500	4,081.5	244.6	3,967.7	14.7	3,963.8	5.9	6.21
4,000	4,473.5	268.2	4,449.5	19.1	4,447.6	6.4	8.91
4,500	–	–	4,918.4	24.3	4,4925.3	6.9	12.40
5,000	–	–	5,407.4	32.0	5,396.5	7.6	17.73

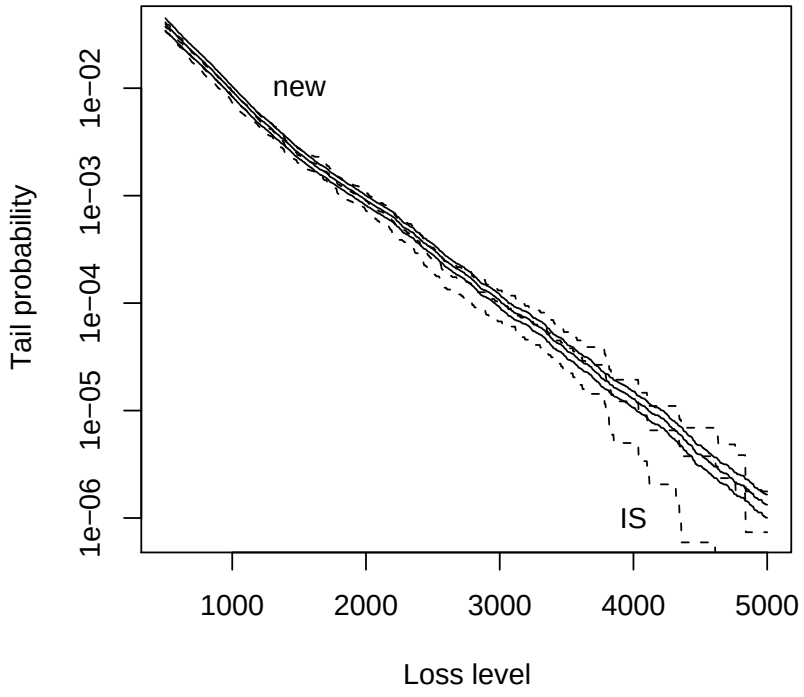

Figure 4.10. Comparison of new method with IS in estimating tail loss probabilities in the 10-factor model using $1,000$ replications.

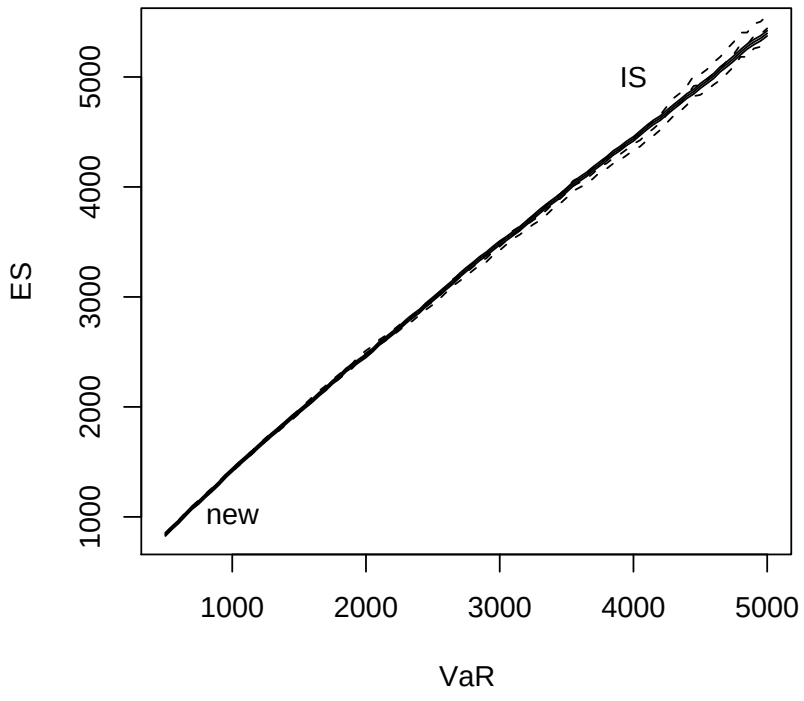

Figure 4.11. Comparison of new method with IS in estimating expected shortfall in the 10-factor model using $10,000$ replications.

4.6. Conclusion

We developed a new method for simulating tail loss probabilities and ES for a standard credit risk portfolio in this Chapter. The new method is an integration of IS with inner replications using geometric shortcut for dependent obligors in a normal copula framework. Numerical results show that our new method is much better than naive simulation for computing tail loss probabilities and ES at a single x and VaR_α value. Furthermore, it is clearly better than two-step IS ([12,13]) in a single simulation to compute tail loss probabilities and ES at multiple x and VaR_α values.

5. COMPARISON OF MEAN SHIFTS FOR IS

For the "normal copula" credit risk model inner importance sampling strategies such as exponential twisting or inner replications of geometric shortcut proposed in Chapter 4 are not enough for reducing the variance for highly dependent obligors. Since, large losses occur primarily because of large outcomes of systematic risk factors in the normal copula framework. One of the classical methods to use as additional (outer) importance sampling strategy is to shift the mean of the systematic risk factors. There are many different approaches that use mean shift optimization heuristics (or approximations) in the literature. See [7] for a summary and comparison of the three. Although, these methods often work, [7] reports that some of the mean shifting methods are performing not better than naive Monte Carlo simulation for middium-sized portfolios. We consider approximate homogenous portfolios only, since [16] reports that for a multifactor heterogeneous model the use of a mixture of IS distributions, each associated with a different shift of mean is required.

In this chapter we first of all explain three different approaches, tail bound approximation used in [13], normal approximation used in Chapter 4 and homogenous portfolio approximation of [25]. The first two methods rely on the approximation (proposed by [14]) that finds the optimal mean shift as the mode of the zero-variance IS distribution whereas the last one uses homogenous portfolio approximation. Then we assess the performance of these three approximations in the numerical examples of Chapter 4.

5.1. Mode of Zero-Variance IS Distribution

$P(L > x)$ is the integration of $P(L > x|Z)$ under the distribution of Z. We are willing to choose a new IS distribution for Z that reduces the variance of Monte Carlo simulation for this integration. It is well-known that the zero-variance IS distribution is a normalization of $P(L > x|Z = z)e^{-z^T z/2}$ where $e^{-z^T z/2}$ is the probability density of $Z = z$. However, the normalization constant is $P(L > x)$, we are looking for.

Nevertheless we can derive some useful information.

[14] propose to use a normal density with the mode of the zero-variance IS distribution as the optimal density in an option-pricing context. Furthermore, [13] uses this approach to compute the optimal mean shifts as the second step of the IS for computing tail loss probabilities in the normal copula framework.

The optimal mean shift is simply the solution to the minimization problem

$$\max_z P(L > x|Z = z)e^{-\frac{1}{2}z^T z} \tag{5.1}$$

where $e^{-z^T z/2}$ is the probability density of $Z = z$. However, this minimization problem is not simple because there exists no simple closed form for $P(L > x|Z = z)$ for large portfolios. Thus, we have to use some approximations for $P(L > x|Z = z)$ to compute the optimal mean shift.

5.1.1. Tail Bound Approximation

Since, $P(L > x|Z = z) \leq e^{-\theta_x(z)x+\psi_{L|Z}(\theta_x(z))}$, [13] uses the tail bound approximation

$$P(L > x|Z = z) \approx e^{-\theta_x(z)x+\psi_{L|Z}(\theta_x(z))}$$

where

$$\psi_{L|Z}(\theta) = \sum_{j=1}^{m} \log(1 + p_j(Z)(e^{c_j\theta} - 1)).$$

5.1.2. Normal Approximation

We compute the conditional expectation and variance of the loss, $E[L|Z=z] = \sum_{j=1}^{m} c_j p_j(z)$ and $Var[L|Z=z] = \sum_{j=1}^{m} c_j^2 \left[p_j(z) - p_j(z)^2\right]$ that we can use the normal approximation:

$$P\left(L > x|Z=z\right) \approx 1 - \Phi\left(\frac{x - E[L|Z=z]}{\sqrt{Var[L|Z=z]}}\right).$$

After approximating $P(L > x|Z=z)$ using tail bound or normal approximation, we solve the d-dimensional optimization problem in (5.1). We use the Nelder-Mead Simplex method as implemented in GSL to solve the optimization problem.

5.2. Homogenous Portfolio Approximation

[25] proposes an IS strategy that computes an optimal mean shift using homogenous portfolio approximation to calculate expected shortfall for credit portfolios. [25] approximates the whole portfolio by a homogenous one in which all obligors default with the same probability p and have the same exposure level c;

$$p = \frac{\sum_{j=1}^{m} p_j c_j}{\sum_{j=1}^{m} c_j}, \quad c = \frac{\sum_{j=1}^{m} c_j}{m}.$$

Moreover, different to the latent variable structure in (2.1) all obligors have the same factor loadings in the latent variables;

$$\bar{X}_j = a_0 \epsilon_j + a_1 Z_1 + .. + a_d Z_d, \quad , j = 1, ..., m$$

where

$$(a_1, ..., a_d) = \frac{\Psi}{s} \text{ with } \Psi = (\Psi_1, ..., \Psi_m) = \sum_{j=1}^{m} g_j(a_{j1}, ..., a_{jd})$$

and $g_j = p_j c_j$. The scaling factor s is found by the equation

$$s^2 = \frac{1}{1-a_0^2} \sum_{k=1}^{d} \sum_{l=1}^{d} \Psi_k \Psi_l Cov(Z_k, Z_l)$$

where

$$1 - a_0^2 = \frac{\sum_{k=1}^{d} \sum_{l=1}^{d} \Psi_k \Psi_l Cov(Z_k, Z_l) - \sum_{j=1}^{m} g_j^2 (1 - a_{j0}^2)}{\left(\sum_{j=1}^{m} g_j\right)^2 - \sum_{j=1}^{m} g_j^2}.$$

Please note that $Cov(Z_k, Z_l) = 1$ if $k = l$ otherwise 0 for the credit portfolio model used in [13] and Chapter 4.

The computation of scaling factor s is based on the idea that the weighted sum of the latent variables' covariances in the original and the homogeneous portfolio should be equal. Thus, following equation holds

$$\sum_{i}^{m} \sum_{j}^{m} g_i g_j Cov(X_i, X_j) = \sum_{i}^{m} \sum_{j}^{m} g_i g_j Cov(\bar{X}_i, \bar{X}_j).$$

After evaluating the equivalent homogeneous portfolio, [25] considers the loss function for the infinite homogeneous portfolio;

$$L_d^{\infty}(Z_1, ..., Z_d) = c\Phi\left(\frac{\Phi^{-1}(p) + \sum_{l=1}^{d} a_l Z_l}{a_0}\right).$$

The IS estimator that shifts the mean of vector Z as $\mu = (\mu_1, ..., \mu_d)$ is

$$L_d^{\infty} \mathbf{1}_{\{L_d^{\infty} > c^{\infty}\}} \frac{N_{0,1}}{N_{\mu,1}}$$

for the expected shortfall of the infinite homogeneous portfolio for a given probability level $100(1-\alpha)$ percent. Please note that c^{∞} is the $100(1-\alpha)$ percent quantile of L_d^{∞}.

[25] computes μ in two steps. In the first step they decrease the dimension of the problem to one (only one systematic risk factor) and compute the optimal mean shift for this case. Then, in the second step, they lift the optimal mean shift computed in the first step to dimension d.

For the first step, the optimization problem is to find $\mu^{(1)}$ that minimizes

$$\int_{\Phi^{-1}(1-\alpha)}^{\infty} \frac{\left(L_1^{\infty} N_{0,1}\right)^2}{N_{\mu^{(1)},1}} dx \tag{5.2}$$

where

$$L_1^{\infty} = c\Phi\left(\frac{\Phi^{-1}(p) + \sqrt{1-a_0^2}x}{a_0}\right).$$

After computing $\mu^{(1)}$, [25] uses the following equation

$$\mu_k = \frac{\mu^{(1)} \sum_{l=1}^{d} Cov(Z_k, Z_l) a_l}{\sqrt{1-a_0^2}}, \quad , k = 1, ..., d$$

to compute the optimal mean shift vector. For more details and proofs see [24].

To summarize, homogenous portfolio approximation of [25] has the advantage of requiring a one-dimensional optimization problem instead of d-dimensional one. On the other hand, it includes the evaluation of the numerical integration given in (5.2).

5.3. Numerical Results

In this section, we evaluate the performance of the optimal mean shift algorithms given in the previous sections for the numerical examples of Chapter 4. Since, our new methodology proposed in Chapter 4 has an outer IS strategy like the two-step IS methodology of [13] and variances computed are approximately the same for single loss values, we can use both of the methods for assessing the performance of outer IS methodologies. We choose the two-step IS methodology of [13] because it is a well

Table 5.1. Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 10-factor model to compute tail loss probabilities. $n = 10,000$. Execution times (in seconds) are in parentheses.

x	prob.	hl (TBA)	hl (NA)	hl (HA)
500	$3.87 * 10^{-2}$	$1.14 * 10^{-3}(17)$	$1.14 * 10^{-3}(12)$	$1.19 * 10^{-3}(11)$
$1,000$	$8.67 * 10^{-3}$	$2.99 * 10^{-4}(16)$	$2.95 * 10^{-4}(13)$	$2.99 * 10^{-4}(11)$
$2,000$	$8.43 * 10^{-4}$	$2.98 * 10^{-5}(15)$	$2.93 * 10^{-5}(12)$	$2.99 * 10^{-5}(12)$
$3,000$	$1.08 * 10^{-4}$	$4.14 * 10^{-6}(17)$	$3.95 * 10^{-6}(11)$	$4.10 * 10^{-6}(12)$
$4,000$	$1.32 * 10^{-5}$	$5.51 * 10^{-7}(17)$	$5.38 * 10^{-7}(12)$	$5.52 * 10^{-7}(12)$

known method and many could be interested in the results of this comparison. [7] omits the exponential twisting part of [13] in their comparison of the methods. We include it to compare the mean shifts for a good simulation method. See Appendix B.3 for the C codes.

Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 10-factor model are given in Table 5.1 (to compute tail loss probabilities) and 5.2 (to compute expected shortfalls). If we compare the half lengths of the confidence intervals, normal approximation is most of the time better than the other two methods. On the other hand, the fastest one is the homogenous approximation. As we have only slight differences between the half lengths of the confidence intervals and execution times of the methods, we can definitely use all three methods for finding the mean shift of outer IS in the 10-factor model.

Half lengths of the confidence intervals for using tail bound approximation, normal approximation and homogenous approximation for the optimal mean shift as outer IS and exponential twisting as inner IS in the 21-factor model are given in Table 5.3

Table 5.2. Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 10-factor model to compute expected shortfalls. $n = 100,000$. Execution times (in seconds) are in parentheses.

VaR_α	ES	hl (TBA)	hl (NA)	hl (HA)
500	842.1	2.99(126)	2.99(113)	2.99(108)
1,000	1,419.8	3.90(131)	3.91(121)	3.89(114)
2,000	2,478.0	4.30(127)	4.25(120)	4.28(120)
3,000	3,480.2	4.48(121)	4.48(114)	4.49(110)
4,000	4,447.2	4.47(126)	4.50(117)	4.47(117)

(to compute tail loss probabilities) and 5.4 (to compute expected shortfalls). If we compare the methods according to their half length of the confidence intervals and execution times, homogeneous approximation is always better than the other two in the 21-factor model but the differences are small.

Half lengths of the confidence intervals for using tail bound approximation, normal approximation and homogenous approximation for the optimal mean shift as outer IS and exponential twisting as inner IS in the 5-factor model are given in Table 5.3 (to compute tail loss probabilities) and 5.4 (to compute expected shortfalls). If we solely look at the half length of the confidence intervals, homogeneous approximation is definitely the worst in the 21-factor model. And, normal approximation is generally slightly better than tail bound approximation. If we compare the execution times, homogenous approximation is fastest as in the previous models and tail bound approximation is slowest.

Summarizing we evaluate the performance of three mean shift optimization approximations: tail bound approximation, normal approximation and homogenous approximation. Note that, the calculated mean shifts are different to each other for the methods discussed in this section as reported in [13]. For different models, we arrived

Table 5.3. Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 21-factor model to compute tail loss probabilities. $n = 10,000$. Execution times (in seconds) are in parentheses.

x	prob.	hl (TBA)	hl (NA)	hl (HA)
$2,500$	$4.96 * 10^{-2}$	$1.62 * 10^{-3}(25)$	$1.69 * 10^{-3}(14)$	$1.57 * 10^{-3}(11)$
$10,000$	$1.16 * 10^{-2}$	$4.10 * 10^{-4}(27)$	$4.23 * 10^{-4}(16)$	$3.62 * 10^{-4}(13)$
$20,000$	$2.69 * 10^{-3}$	$9.69 * 10^{-5}(26)$	$9.66 * 10^{-5}(17)$	$9.65 * 10^{-5}(13)$
$30,000$	$6.29 * 10^{-4}$	$2.51 * 10^{-5}(37)$	$2.43 * 10^{-5}(17)$	$2.36 * 10^{-5}(13)$
$40,000$	$7.59 * 10^{-5}$	$3.48 * 10^{-6}(27)$	$3.37 * 10^{-6}(16)$	$3.20 * 10^{-6}(14)$

Table 5.4. Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 21-factor model to compute expected shortfalls. $n = 100,000$. Execution times (in seconds) are in parentheses.

VaR_α	ES	hl (TBA)	hl (NA)	hl (HA)
$2,500$	$7,597.3$	49.31(125)	49.30(113)	48.73(111)
$10,000$	$16,816.3$	65.74(140)	65.51(129)	59.95(123)
$20,000$	$26,360.4$	56.63(150)	57.43(139)	56.45(127)
$30,000$	$34,840.7$	45.67(168)	44.74(145)	43.75(131)
$40,000$	$42,587.9$	27.05(160)	26.57(148)	26.05(131)

Table 5.5. Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 5-factor model to compute tail loss probabilities. $n = 10,000$. Execution times (in seconds) are in parentheses.

x	prob.	hl (TBA)	hl (NA)	hl (HA)
$5,000$	$4.64 * 10^{-2}$	$1.53 * 10^{-3}(55)$	$1.53 * 10^{-3}(52)$	$1.55 * 10^{-3}(46)$
$10,000$	$1.82 * 10^{-2}$	$6.24 * 10^{-4}(56)$	$6.53 * 10^{-4}(54)$	$6.34 * 10^{-4}(51)$
$15,000$	$8.29 * 10^{-3}$	$2.82 * 10^{-4}(58)$	$2.79 * 10^{-4}(55)$	$3.20 * 10^{-4}(51)$
$20,000$	$3.91 * 10^{-3}$	$1.38 * 10^{-4}(61)$	$1.34 * 10^{-4}(55)$	$1.67 * 10^{-4}(53)$
$25,000$	$1.88 * 10^{-3}$	$6.83 * 10^{-5}(60)$	$6.67 * 10^{-5}(57)$	$8.27 * 10^{-5}(53)$
$30,000$	$7.71 * 10^{-4}$	$3.03 * 10^{-5}(67)$	$3.00 * 10^{-5}(58)$	$3.47 * 10^{-5}(55)$

Table 5.6. Half lengths (hl) of the confidence intervals for using tail bound approximation (TBA), normal approximation (NA) and homogenous approximation (HA) for the optimal mean shift as outer IS and exponential twisting as inner IS in the 5-factor model to compute expected shortfalls. $n = 100,000$. Execution times (in seconds) are in parentheses.

VaR_α	ES	hl (TBA)	hl (NA)	hl (HA)
$5,000$	$10,729.5$	52.76(513)	53.18(503)	53.38(457)
$10,000$	$16,366.8$	57.41(539)	56.28(534)	56.74(486)
$15,000$	$21,296.4$	56.39(540)	56.73(532)	61.05(504)
$20,000$	$25,856.6$	51.27(553)	50.50(544)	60.78(515)
$25,000$	$29,948.1$	43.36(565)	43.27(554)	49.38(526)
$30,000$	$33,747.6$	33.46(582)	33.48(569)	38.00(535)

at different conclusions. Half lengths of the confidence intervals are very similar for all of the approximations. Also, the speed of the approximations are quite similar when number of factors are large. However, tail bound approximation is slowest for a small number of factors. Moreover, normal approximation (being simplest) or homogenous approximation would be our choice when using one of these approximations. Since, both are fast, reliable and outperform tail bound approximation.

6. BETTER CONFIDENCE INTERVALS FOR IS

It is well known that for highly skewed distributions the standard method of using the t statistic for the confidence interval for the mean does not give robust results. This is an important problem for IS as its final distribution is often skewed due to a heavy tailed weight ditribution (see [11]). In this chapter, we first explain the transformation proposed by [21] (Hall's transformation) to correct the confidence interval of the mean and then evaluate the performance of this method for two numerical examples from finance, which have closed form solutions. Finally, we assess the performance of this method for the credit risk models given in Chapter 4.

6.1. Hall's Transformation of the t Statistic

We describe Hall's transformation of the t statistic to correct the confidence interval for the mean. Let's give the required notation first. Note that we use the same notation as [34], compares the performance of Johnson's and Hall's transformations and their bootstrap versions by looking at the coverage accuracy of one-sided confidence intervals.

We assume that we have a random sample, $X_1, ..., X_n$, from a population with mean θ and variance τ^2. Then we may use the t statistic

$$T = \frac{\sqrt{n}\left(\hat{\theta} - \theta\right)}{\hat{\tau}} \tag{6.1}$$

where $\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\hat{\tau}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{\theta}\right)^2$ to construct a confidence interval for θ. The resulting one sided, $1-\alpha$ confidence interval has a coverage error of order $n^{-1/2}$ as it can be seen from (6.1). However, [21] proposes the following monotone and invertible transformation which assumes that T admits a first order Edgeworth expansion;

$$g(T) = T + n^{-1/2}\hat{\gamma}(aT^2 + b) + n^{-1}(1/3)\left(a\hat{\gamma}\right)^2 T^3$$

where $\hat{\gamma} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{\theta}\right)^3/\hat{\tau}^3$, $a = 1/3$, $b = 1/6$. This transformation has a unique inverse

$$T = g^{-1}(x) = n^{1/2}(a\hat{\gamma})^{-1}\left[(1 + 3a\hat{\gamma}(n^{-1/2}x - n^{-1}b\hat{\gamma}))^{1/3} - 1\right].$$

Using this transformation, [21] gives the upper and lower endpoint with confidence $1-\alpha$: $(-\infty, \hat{\theta} - n^{-1/2}\hat{\tau}g^{-1}(z_\alpha))$ and $(\hat{\theta} - n^{-1/2}\hat{\tau}g^{-1}(z_{1-\alpha}), \infty)$. It can be shown that for the upper and lower endpoints $1-\alpha$ the coverage error converges with speed n^{-1} instead of $n^{-1/2}$.

6.2. TWO EXAMPLES

In this section we want to assess the performance of Hall's transformation compared to the standard t statistics. We consider one-sided coverage accuracies as the measure of evaluation as [21] and [34]. To be able to simulate exact coverage levels we choose examples for which analytical solutions are available.

To give the coverage accuracies, we use $10,000$ samples in these examples. We take the average of the $10,000$ bernoulli trials indicating whether the exact value of the mean is in the calculated confidence intervals to estimate the coverage accuracies. Thus, we can expect to have a natural variance on the coverage levels. It is easy to see that the 99 percent acceptance region for the 95 percent one-sided coverage levels are 0.944 and 0.956. See Appendix B.4 for the C codes

6.2.1. Example 1

We want to compute the following probability

$$P\left(L = \frac{\sum_{i=1}^{d} Z_i}{\sqrt{d}} > x\right) \tag{6.2}$$

where Z_i's are iid standard normal variates. We are applying an IS strategy such that new Z_i's are distributed normally with mean μ_i and standard deviation σ_i. For the one dimensional case, we show that if we choose a mean shift (μ) such that the expected value of L is x then this is an asymptotically optimal mean shift in Appendix A.2. Thus, we choose $\mu_i = x/\sqrt{d}$ so that the expected value of L is x.

We give the achieved coverage levels for 95 percent upper and lower endpoint confidence intervals for a series of σ_i, d and n values in Table 6.1. We observe that Hall's confidence interval works better than standard t statistic when n is small. Although, increasing the dimension of the problem decreases the quality of the confidence level of both methods except for the case with $\sigma_i = 1.0$, Hall's transformation produces better confidence intervals than ordinary t statistic.

6.2.2. Example 2

The maturity of the Asian option is 0.2 (years) and control points are $0.12, 0.14$, $0.16, 0.18$ and 0.2. We choose a geometric average Asian option to price since we have a closed form solution for it and the price of it is quite close to an arithmetic average Asian option.

We use IS to decrease the variance of the simulation. To compute the optimal IS parameters, we use the Nelder-Mead Simplex method as implemented in GSL. We use $n = 10,000,000$ for the optimization. Achieved coverage levels for 95 percent upper and lower endpoint confidence intervals are given in Table 6.2 for $S_0 = 100$ and $r = 0.09$. Note that, we use $10,000$ simulated samples to compute the coverage levels.

It is easy to see from Table 6.2, there is no statistical evidence showing Hall's method is better than the standard t statistic for small values of the strike price (K). However, as the strike price increases, Hall's confidence interval clearly works better than the standard t statistic.

Table 6.1. Achieved coverage levels for 95 percent upper and lower endpoint confidence intervals ($x = 2$)

			Upper Endpoint		Lower Endpoint	
σ	d	n	Ordinary t	Hall	Ordinary t	Hall
0.9	10	1000	0.935	0.946	0.959	0.948
		10000	0.950	0.954	0.953	0.951
	20	1000	0.925	0.942	0.968	0.950
		10000	0.935	0.943	0.956	0.948
	50	1000	0.897	0.929	0.980	0.954
		10000	0.9257	0.942	0.969	0.950
1.0	10	1000	0.945	0.952	0.957	0.952
		10000	0.947	0.948	0.953	0.951
	20	1000	0.945	0.950	0.956	0.952
		10000	0.950	0.951	0.954	0.953
	50	1000	0.946	0.953	0.956	0.951
		10000	0.946	0.948	0.954	0.952
1.1	10	1000	0.943	0.952	0.955	0.948
		10000	0.948	0.951	0.955	0.953
	20	1000	0.939	0.950	0.957	0.948
		10000	0.943	0.946	0.954	0.952
	50	1000	0.929	0.949	0.966	0.950
		10000	0.942	0.949	0.960	0.954

Table 6.2. Achieved coverage levels for 95 percent upper and lower endpoint confidence intervals ($S_0 = 100$, $r = 0.09$).

		optimal IS			Upper Endpoint		Lower Endpoint	
volatility	K	μ	σ	n	Ordinary t	Hall	Ordinary t	Hall
0.1	90	0.108	1.001	1000	0.955	0.954	0.953	0.954
				10000	0.951	0.951	0.949	0.949
	100	0.288	1.035	1000	0.947	0.950	0.955	0.953
				10000	0.947	0.948	0.952	0.952
	105	0.477	1.108	1000	0.946	0.953	0.958	0.951
				10000	0.945	0.948	0.954	0.951
	110	0.753	1.283	1000	0.927	0.949	0.964	0.949
				10000	0.944	0.951	0.956	0.950
	115	1.106	1.527	1000	0.905	0.951	0.979	0.954
				10000	0.938	0.954	0.959	0.950
	120	1.508	1.798	1000	0.848	0.940	0.989	0.959
				10000	0.920	0.945	0.969	0.951

6.3. Credit Risk Application

In this section, we evaluate the performance of Hall's confidence interval compared to standard t statistics on the numerical examples of Chapter 4. We use the two-step IS methodology of [13] to compute tail loss probabilities and confidence intervals using ordinary t statistics and Hall's method in Tables 6.3 to 6.5. The tail loss probabilities in the tables were computed using $1,000,000$ simulations. There is only a small difference in the upper and lower borders of the confidence intervals for the two methods for all of the models.

Moreover, we computed achieved coverage probabilities for 95 percent upper and lower endpoint confidence intervals for computing tail loss probabilities for all of the models in Table 6.6. We use $10,000$ simulated samples to compute the coverage levels in this table. Hall's confidence interval apparently works better than the standard t statistic for our credit risk examples.

Using Hall's method for ES computation is more difficult as it requires the skewness of the ratio estimate which is not considered in the literature.

Table 6.3. Nearly exact tail loss probabilities and upper and lower bound of the confidence intervals by using ordinary t statistics and Hall's method in the 10-factor model. $n = 1,000$.

		Ordinary t		Hall	
x	prob.	upper bound	lower bound	upper bound	lower bound
500	$3.86 * 10^{-2}$	$4.09 * 10^{-2}$	$3.47 * 10^{-2}$	$4.10 * 10^{-2}$	$3.49 * 10^{-2}$
$1,000$	$8.59 * 10^{-3}$	$9.46 * 10^{-3}$	$7.86 * 10^{-3}$	$9.51 * 10^{-3}$	$7.90 * 10^{-3}$
$2,000$	$8.45 * 10^{-4}$	$9.84 * 10^{-4}$	$8.16 * 10^{-4}$	$9.88 * 10^{-4}$	$8.19 * 10^{-4}$
$3,000$	$1.06 * 10^{-4}$	$1.15 * 10^{-4}$	$9.41 * 10^{-5}$	$1.15 * 10^{-4}$	$9.47 * 10^{-5}$
$4,000$	$1.31 * 10^{-5}$	$1.41 * 10^{-5}$	$1.11 * 10^{-5}$	$1.42 * 10^{-5}$	$1.13 * 10^{-5}$

To summarize, Hall's method is a simple method and more accurate than the

Table 6.4. Nearly exact tail loss probabilities and upper and lower bound of the confidence intervals by using ordinary t statistics and Hall's method in the 21-factor model. $n = 1,000$.

		Ordinary t		Hall	
x	prob.	upper bound	lower bound	upper bound	lower bound
$2,500$	$5.00 * 10^{-2}$	$5.29 * 10^{-2}$	$4.37 * 10^{-2}$	$5.35 * 10^{-2}$	$4.41 * 10^{-2}$
$10,000$	$1.12 * 10^{-2}$	$1.14 * 10^{-2}$	$9.44 * 10^{-3}$	$1.15 * 10^{-2}$	$9.49 * 10^{-3}$
$20,000$	$2.72 * 10^{-3}$	$3.16 * 10^{-3}$	$2.62 * 10^{-3}$	$3.17 * 10^{-3}$	$2.63 * 10^{-3}$
$30,000$	$6.16 * 10^{-4}$	$6.93 * 10^{-4}$	$5.63 * 10^{-4}$	$6.97 * 10^{-4}$	$5.66 * 10^{-4}$
$40,000$	$7.35 * 10^{-5}$	$7.96 * 10^{-5}$	$6.25 * 10^{-5}$	$8.02 * 10^{-5}$	$6.30 * 10^{-5}$

Table 6.5. Nearly exact tail loss probabilities and upper and lower bound of the confidence intervals by using ordinary t statistics and Hall's method in the 5-factor model. $n = 1,000$.

		Ordinary t		Hall	
x	prob.	upper bound	lower bound	upper bound	lower bound
$5,000$	$4.65 * 10^{-2}$	$5.05 * 10^{-2}$	$4.28 * 10^{-2}$	$5.08 * 10^{-2}$	$4.30 * 10^{-2}$
$10,000$	$1.84 * 10^{-2}$	$2.18 * 10^{-2}$	$1.78 * 10^{-2}$	$2.20 * 10^{-2}$	$1.80 * 10^{-2}$
$15,000$	$8.35 * 10^{-3}$	$9.30 * 10^{-3}$	$7.64 * 10^{-3}$	$9.41 * 10^{-3}$	$7.71 * 10^{-3}$
$20,000$	$3.97 * 10^{-3}$	$4.27 * 10^{-3}$	$3.56 * 10^{-3}$	$4.29 * 10^{-3}$	$3.57 * 10^{-3}$
$25,000$	$1.85 * 10^{-3}$	$2.02 * 10^{-3}$	$1.67 * 10^{-3}$	$2.03 * 10^{-3}$	$1.67 * 10^{-3}$
$30,000$	$7.78 * 10^{-4}$	$8.99 * 10^{-4}$	$7.28 * 10^{-4}$	$9.03 * 10^{-4}$	$7.33 * 10^{-4}$

Table 6.6. Achieved coverage levels for 95 percent upper and lower endpoint confidence intervals for computing tail loss probabilities. $n = 1,000$.

		Upper Endpoint		Lower Endpoint	
model	x	Ordinary t	Hall	Ordinary t	Hall
$10-$factor	500	0.935	0.943	0.962	0.955
	1,000	0.937	0.946	0.958	0.950
	2,000	0.943	0.951	0.960	0.954
	3,000	0.934	0.944	0.960	0.952
	4,000	0.935	0.945	0.958	0.951
$21-$factor	2,500	0.928	0.939	0.971	0.961
	10,000	0.931	0.945	0.962	0.950
	20,000	0.938	0.949	0.963	0.957
	30,000	0.941	0.952	0.951	0.944
	40,000	0.939	0.951	0.957	0.947
$5-$factor	5,000	0.935	0.945	0.961	0.951
	15,000	0.941	0.951	0.956	0.948
	20,000	0.947	0.955	0.951	0.944
	25,000	0.934	0.943	0.963	0.956
	30,000	0.943	0.952	0.958	0.950

standard t statistic in constructing the confidence interval for IS simulations. The difference between the methods becomes large when we have small number of simulations (or large skewness). And, we may have to use small number of simulations for real credit risk portfolios because of the number of obligors (up to $100,000$ is important in practice) and/or the complexity of the dependence structure across obligors.

7. CONCLUSIONS

Understanding and improving of Monte Carlo simulations for portfolio credit risk was the aim of this thesis. We focused on the normal copula model, which is the core of most standard industry methodologies, for the dependence structure across obligors. Throughout the thesis we considered not only VaR but also ES as risk measure.

We showed how we can compute optimal IS probabilities and compared them with the "asymptotically optimal" probabilities for credit portfolios consisting of group of independent obligors. The results showed that "asymptotically optimal" probabilities are nearly the same with the optimal ones when the number of obligors is high and the event we simulate is a rare event.

Then, we developed a new method for simulating tail loss probabilities and conditional expectations for a standard credit risk portfolio. The new method is an integration of IS with inner replications using geometric shortcut for dependent obligors in a normal copula framework. Numerical results show that our new method is much better than naive simulation and as good as two-step IS (see [12,13]) for computing tail loss probabilities and conditional expectations at a single x and VaR_α value. Furthermore, it is clearly better than two-step IS in a single simulation to compute tail loss probabilities and conditional expectations at multiple x and VaR_α values. Therefore, the new method is a substantial improvement for an important real world problem.

Then, we evaluated the performance of outer IS strategies, which consider only shifting the mean of the systematic risk factors on realistic credit risk portfolios. The results showed that normal approximation and homogenous approximation are better than tail bound approximation as they are fast, reliable and outperform tail bound approximation.

Finally, we demonstrated with examples that Hall's transformation [21] could be safely used in correcting the confidence intervals of financial simulations. Thus, we

suggest to use Hall's confidence interval instead of standard t statistic for simulations that include IS, especially also for credit risk simulations.

APPENDIX A: ADDITIONAL NUMERICAL RESULTS AND NOTES

A.1. Additional Numerical Results For Chapter 3

Table A.1. Optimal and exponential twisting [13] probabilities and percent differences of variances between exponential twisting and optimal (% diff. of var. ET-O) and naive simulation and optimal (percent diff. of var. naive-O) for $x = 1$ in computing $P(L > x)$.

m_A	m_B	p_A	p_B	OP		ETP		% diff. of var. ET-O	% diff. of var. naive-O
5	5	0.001	0.001	0.198	0.199	0.100	0.100	80.25	973724.78
5	5	0.001	0.01	0.05	0.356	0.021	0.179	94.94	40771.36
5	5	0.001	0.05	0.01	0.39	0.005	0.195	110.43	2421.61
5	5	0.01	0.01	0.198	0.199	0.100	0.100	82.30	10802.93
5	5	0.01	0.05	0.079	0.324	0.036	0.164	98.44	1574.12
5	5	0.05	0.005	0.358	0.046	0.180	0.020	103.11	1953.67
5	10	0.001	0.001	0.138	0.135	0.067	0.067	74.29	397629.38
5	10	0.001	0.01	0.02	0.189	0.010	0.095	80.51	9612.09
5	10	0.001	0.05	0.01	0.202	0.002	0.099	96.58	551.33
5	10	0.01	0.01	0.138	0.135	0.067	0.067	77.57	4655.53
5	10	0.01	0.05	0.04	0.184	0.019	0.091	102.29	476.23
5	10	0.05	0.005	0.318	0.039	0.163	0.018	97.44	1559.54
5	20	0.001	0.001	0.079	0.08	0.040	0.040	70.49	135733.58
5	20	0.001	0.01	0.01	0.098	0.005	0.049	76.45	2523.54
5	20	0.001	0.05	0.01	0.107	0.001	0.05	143.27	143.27
5	20	0.01	0.01	0.079	0.08	0.040	0.040	76.95	1739.69
5	20	0.01	0.05	0.02	0.103	0.01	0.05	145.22	145.22
5	20	0.05	0.005	0.278	0.033	0.139	0.015	92.16	1090.41
10	20	0.001	0.001	0.069	0.067	0.033	0.033	69.48	93302.38
10	20	0.001	0.01	0.01	0.095	0.005	0.047	76.30	2405.26
10	20	0.001	0.05	0.01	0.107	0.001	0.05	128.48	128.48
10	20	0.01	0.01	0.069	0.068	0.033	0.033	77.97	1242.87
10	20	0.01	0.05	0.02	0.099	0.01	0.05	133.24	133.24
10	20	0.05	0.005	0.168	0.018	0.083	0.009	104.07	399.51
20	20	0.001	0.001	0.05	0.05	0.025	0.025	68.65	52209.61
20	20	0.001	0.01	0.01	0.091	0.005	0.045	75.98	2193.01
20	20	0.001	0.05	0.01	0.107	0.001	0.05	102.20	102.20
20	20	0.01	0.01	0.05	0.05	0.025	0.025	81.92	742.15
20	20	0.01	0.05	0.02	0.093	0.01	0.05	112.71	112.71
20	20	0.05	0.005	0.098	0.01	0.05	0.005	133.12	133.12

Table A.2. Optimal and exponential twisting [13] probabilities and percent differences of variances between exponential twisting and optimal (% diff. of var. ET-O) and naive simulation and optimal (% diff. of var. naive-O) for $x = 1$ in computing $E[L|L > x]$.

m_A	m_B	p_A	p_B	OP		ETP		% diff. of var. ET-O	% diff. of var. naive-O
5	5	0.001	0.001	0.2	0.2	0.100	0.100	80.37	978223.16
5	5	0.001	0.01	0.047	0.354	0.021	0.179	95.96	41722.41
5	5	0.001	0.05	0.011	0.392	0.005	0.195	116.51	2667.78
5	5	0.01	0.01	0.2	0.2	0.100	0.100	83.73	11300.54
5	5	0.01	0.05	0.079	0.325	0.036	0.164	106.26	1789.91
5	5	0.05	0.005	0.358	0.046	0.180	0.020	109.88	2184.05
5	10	0.001	0.001	0.133	0.133	0.067	0.067	74.60	400894.95
5	10	0.001	0.01	0.022	0.19	0.010	0.095	82.15	10092.88
5	10	0.001	0.05	0.005	0.204	0.002	0.099	123.46	729.54
5	10	0.01	0.01	0.134	0.134	0.067	0.067	80.21	5008.18
5	10	0.01	0.05	0.041	0.188	0.0188	0.091	128.96	632.83
5	10	0.05	0.005	0.324	0.04	0.163	0.018	105.31	1775.48
5	20	0.001	0.001	0.08	0.08	0.040	0.040	70.75	137460.83
5	20	0.001	0.01	0.01	0.098	0.005	0.049	80.45	2796.17
5	20	0.001	0.05	0.002	0.114	0.001	0.05	297.53	297.53
5	20	0.01	0.01	0.081	0.081	0.040	0.040	82.57	1977.12
5	20	0.01	0.05	0.023	0.111	0.01	0.05	285.77	285.77
5	20	0.05	0.005	0.276	0.033	0.139	0.015	102.55	1285.31
10	20	0.001	0.001	0.067	0.067	0.033	0.033	69.92	94819.05
10	20	0.001	0.01	0.01	0.096	0.005	0.048	80.46	2672.93
10	20	0.001	0.05	0.002	0.114	0.001	0.05	295.77	295.77
10	20	0.01	0.01	0.068	0.068	0.033	0.033	85.84	1454.69
10	20	0.01	0.05	0.022	0.107	0.01	0.05	274.24	274.24
10	20	0.05	0.005	0.173	0.019	0.083	0.009	136.82	551.47
20	20	0.001	0.001	0.05	0.05	0.025	0.025	69.06	53296.22
20	20	0.001	0.01	0.01	0.091	0.005	0.045	80.49	2451.80
20	20	0.001	0.05	0.002	0.113	0.001	0.05	292.42	292.42
20	20	0.01	0.01	0.051	0.051	0.025	0.025	95.47	922.19
20	20	0.01	0.05	0.021	0.101	0.01	0.05	256.00	256.00
20	20	0.05	0.005	0.107	0.011	0.05	0.005	274.18	274.18

Table A.3. Optimal and exponential twisting [13] probabilities and percent differences of variances between exponential twisting and optimal (% diff. of var. ET-O) and naive simulation and optimal (% diff. of var. naive-O) for $x = 5$ in computing $P(L > x)$.

m_A	m_B	p_A	p_B	OP		ETP		% diff. of var. ET-O	% diff. of var. naive-O
5	5	0.001	0.001	0.598	0.599	0.500	0.500	29.77	$1.60E17$
5	5	0.001	0.01	0.338	0.857	0.239	0.761	46.83	$3.88E13$
5	5	0.001	0.05	0.248	0.957	0.121	0.879	94.72	$3.07E10$
5	5	0.01	0.01	0.598	0.599	0.500	0.500	29.88	$1.68E11$
5	5	0.01	0.05	0.408	0.797	0.305	0.695	38.10	$6.92E8$
5	5	0.05	0.005	0.858	0.337	0.764	0.236	47.68	$2.63E9$
5	10	0.001	0.001	0.398	0.399	0.333	0.333	19.77	$5.26E15$
5	10	0.001	0.01	0.098	0.548	0.078	0.461	24.85	$8.81E10$
5	10	0.001	0.05	0.02	0.586	0.0180	0.491	28.42	$1.14E7$
5	10	0.01	0.01	0.398	0.399	0.333	0.333	19.90	$5.81E9$
5	10	0.01	0.05	0.168	0.52	0.129	0.436	23.37	4621209.84
5	10	0.05	0.005	0.768	0.219	0.672	0.164	30.14	$5.00E8$
5	20	0.001	0.001	0.238	0.24	0.200	0.200	15.54	$1.30E14$
5	20	0.001	0.01	0.04	0.291	0.031	0.24	16.61	$5.94E8$
5	20	0.001	0.05	0.01	0.299	0.006	0.248	17.96	89757.58
5	20	0.01	0.01	0.238	0.24	0.200	0.200	15.75	$1.60E8$
5	20	0.01	0.05	0.069	0.284	0.056	0.236	17.85	67547.46
5	20	0.05	0.005	0.648	0.139	0.563	0.109	21.52	$7.38E7$
10	20	0.001	0.001	0.198	0.199	0.167	0.167	14.74	$3.79E13$
10	20	0.001	0.01	0.04	0.283	0.030	0.235	15.91	$4.89E8$
10	20	0.001	0.05	0.01	0.298	0.006	0.247	17.46	86410.81
10	20	0.01	0.01	0.198	0.2	0.167	0.167	14.99	$4.92E7$
10	20	0.01	0.05	0.069	0.27	0.052	0.224	17.02	50179.78
10	20	0.05	0.005	0.458	0.071	0.387	0.057	19.84	1995254.55
20	20	0.001	0.001	0.148	0.149	0.125	0.125	13.83	$5.72E12$
20	20	0.001	0.01	0.03	0.263	0.028	0.222	14.52	$3.40E8$
20	20	0.001	0.05	0.01	0.295	0.006	0.244	16.36	80095.11
20	20	0.01	0.01	0.148	0.149	0.125	0.125	14.19	8278042.15
20	20	0.01	0.05	0.06	0.246	0.047	0.203	16.60	29893.18
20	20	0.05	0.005	0.268	0.033	0.223	0.027	17.36	49944.99

Table A.4. Optimal and Exponential Twisting ([13]) probabilities and percent differences of variances between exponential twisting and optimal (% diff. of var. ET-O) and naive simulation and optimal (% diff. of var. naive-O) for $x = 5$ in computing $E[L|L > x]$.

m_A	m_B	p_A	p_B	OP		ETP		% diff. of var. ET-O	% diff. of var. naive-O
5	5	0.001	0.001	0.6	0.6	0.500	0.500	29.79	$1.60E17$
5	5	0.001	0.01	0.342	0.858	0.239	0.761	46.88	$3.89E13$
5	5	0.001	0.05	0.244	0.957	0.121	0.879	94.85	$3.08E10$
5	5	0.01	0.01	0.6	0.6	0.500	0.500	29.91	$1.68E11$
5	5	0.01	0.05	0.404	0.796	0.305	0.695	38.20	$6.96E8$
5	5	0.05	0.005	0.861	0.338	0.764	0.236	47.81	$2.64E9$
5	10	0.001	0.001	0.4	0.4	0.333	0.333	19.78	$5.26E15$
5	10	0.001	0.01	0.101	0.55	0.078	0.461	24.94	$8.85E10$
5	10	0.001	0.05	0.024	0.588	0.018	0.491	29.22	$1.17E7$
5	10	0.01	0.01	0.4	0.4	0.333	0.333	19.94	$5.84E9$
5	10	0.01	0.05	0.164	0.519	0.129	0.436	23.60	4713387.76
5	10	0.05	0.005	0.764	0.218	0.672	0.164	30.25	$5.04E8$
5	20	0.001	0.001	0.24	0.24	0.200	0.200	15.56	$1.30E14$
5	20	0.001	0.01	0.038	0.291	0.031	0.242	16.72	$6.00E8$
5	20	0.001	0.05	0.008	0.299	0.006	0.248	18.73	94311.58
5	20	0.01	0.01	0.24	0.24	0.200	0.200	15.81	$1.62E8$
5	20	0.01	0.05	0.069	0.284	0.056	0.236	18.29	70992.21
5	20	0.05	0.005	0.645	0.139	0.563	0.109	21.61	$7.46E7$
10	20	0.001	0.001	0.2	0.2	0.1676	0.167	14.77	$3.80E13$
10	20	0.001	0.01	0.036	0.282	0.030	0.235	16.38	$4.96E8$
10	20	0.001	0.05	0.008	0.297	0.006	0.247	18.65	91151.17
10	20	0.01	0.01	0.2	0.2	0.167	0.167	15.08	$5.00E7$
10	20	0.01	0.05	0.065	0.269	0.052	0.224	17.92	53131.52
10	20	0.05	0.005	0.458	0.071	0.387	0.057	20.00	2043266.41
20	20	0.001	0.001	0.15	0.15	0.125	0.125	13.89	$5.74E12$
20	20	0.001	0.01	0.034	0.266	0.028	0.222	15.83	$3.47E8$
20	20	0.001	0.05	0.0080	0.294	0.006	0.244	18.47	85220.65
20	20	0.01	0.01	0.15	0.15	0.125	0.125	14.34	8459243.87
20	20	0.01	0.05	0.057	0.244	0.047	0.203	17.50	31860.70
20	20	0.05	0.005	0.268	0.033	0.223	0.027	17.84	52699.03

A.2. Asymptotically Optimal Mean Shift For One Dimensional Portfolio Problem

Our main problem is to compute the following probability

$$P\left(L = Z > x\right)$$

where Z has standard normal distribution. We want to show that if we choose a mean shift (μ) for Z such that $E[L] = x$ (i.e. $\mu = x$ here) then this mean shift is an asymptotically optimal one.

Second moment of the IS estimator which has a mean shift μ is

$$M_2(\mu, x) = \int_x^\infty \frac{N_{0,1}^2}{N_{\mu,1}} dz. \tag{A.1}$$

Then, if we choose $\mu = x$ and take the derivative of (A.1) with respect to μ using Mathematica ([32]),

$$\frac{\partial M_2(\mu = x, x)}{\partial \mu} = -e^{-x^2}\sqrt{\frac{2}{\Pi}} + xe^{x^2} erfc\left(\sqrt{2}x\right).$$

where $erfc(.)$ is complementary error function. Finally,

$$\lim_{x \to +\infty} \frac{\partial M_2(\mu = x, x)}{\partial \mu} = 0.$$

This shows that our choice of $\mu = x$ is an asymptotically optimal mean shift. Furthermore, we numerically minimize (A.1) to compute optimal mean shifts for various x values using Mathematica in Table A.5. The results validate the asymptotically optimality of the choice $\mu = x$.

Table A.5. Optimal mean shifts for various x values.

x	2	3	4
optimal μ	2.216	3.155	4.000

APPENDIX B: C CODES

B.1. Common Codes

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_multimin.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_cdf.h>
#include <gsl/gsl_randist.h>
#include <gsl/gsl_statistics.h>
#include <gsl/gsl_sf_erf.h>
#include <gsl/gsl_sf_gamma.h>
#include <gsl/gsl_sort_double.h>
#include <gsl/gsl_sort_vector.h>
#include <gsl/gsl_permutation.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_roots.h>
#include <gsl/gsl_errno.h>

// Global variables
int model = 3;
const int d=5;
const int numbofOblgrs = 4800;
double VarProb;
double pZ[numbofOblgrs];
double pthetaZ[numbofOblgrs];
double x_p;
double p[numbofOblgrs];
double c[numbofOblgrs];
double a[numbofOblgrs][d];
double mu[d];
double mu_min[d];
double mu_max[d];
double theta;
double a_h[d];
double mu1;
double c_h, p_h;
double RSquare;
double RiSquare[numbofOblgrs];
double s;
double psi[numbofOblgrs];
double meanofDM[d];
double stdDevofDM[d];
double sumCV = 0;
gsl_rng * r;
```

```
void GlassLiNumericExampleInit(){
  const gsl_rng_type * Trand;
  unsigned long randSeed;
  Trand = gsl_rng_default;
  r = gsl_rng_alloc (Trand);

  srand(time(NULL));
  randSeed = rand();
  gsl_rng_set (r, randSeed);

//  model 1 and 2
  for(int i=0;i< numbofOblgrs;i++){
    if(model == 1){
      for(int j = 0;j < d; j++){
        a[i][j] = gsl_rng_uniform (r) * sqrt(1.0/d);

      }
      p[i] = 0.01*(1+sin(16*M_PI*(i+1)/numbofOblgrs));
      c[i] = pow(ceil(5*(double)(i+1)/numbofOblgrs),2);
    }
    else if(model == 2){
      a[i][0] = 0.8;
      a[i][i/100+1] = 0.4;
      int k = i%100;
      a[i][k/10+11] = 0.4;
      p[i] = 0.01*(1+sin(16*M_PI*(i+1)/numbofOblgrs));
      double c1 = 99.0/999;
      double c0 = 1-c1;
      c[i] = c0 + c1*(i+1);
    }
  }

// model 3
  if(model == 3){
    // Segment 1A
    for(int i=0;i< 800;i++){
      a[i][0] = 0.7;
      a[i][1] = 0.5;
      a[i][2] = 0.1;
      p[i] = 0.01;
      c[i] = 20;
    }
    // Segment 1B
    for(int i=800;i< 1600;i++){
      a[i][0] = 0.7;
      a[i][1] = 0.5;
      a[i][2] = 0.1;
      p[i] = 0.02;
      c[i] = 10;
    }
    // Segment 2A
```

```
    for(int i=1600;i< 2400;i++){
      a[i][0] = 0.7;
      a[i][2] = 0.2;
      a[i][3] = 0.4;
      p[i] = 0.02;
      c[i] = 10;
    }
    // Segment 2B
    for(int i=2400;i< 3200;i++){
      a[i][0] = 0.7;
      a[i][2] = 0.2;
      a[i][3] = 0.4;
      p[i] = 0.04;
      c[i] = 5;
    }

    // Segment 3A
    for(int i=3200;i< 4000;i++){
      a[i][0] = 0.7;
      a[i][3] = 0.4;
      a[i][4] = 0.5;
      p[i] = 0.03;
      c[i] = 5;
    }
    // Segment 3B
    for(int i=4000;i< 4800;i++){
      a[i][0] = 0.7;
      a[i][3] = 0.4;
      a[i][4] = 0.5;
      p[i] = 0.05;
      c[i] = 1;
    }
  }


  for(int i=0;i< numbofOblgrs;i++){
    double sqr_a=0;
    for(int j = 0;j<d;j++){
      sqr_a = sqr_a + pow(a[i][j], 2);
    }
    RiSquare[i] = sqr_a;
  }
}

void computeDependentProb(double z_g[]){
  double sum_az;
  double sqr_a;

  for(int i=0;i< numbofOblgrs;i++){
    sum_az = 0;
    sqr_a = 0;
    for(int j = 0;j<d;j++){
```

```
      sum_az = sum_az + a[i][j]*z_g[j];
      sqr_a = sqr_a + pow(a[i][j], 2);
    }
    pZ[i] = gsl_cdf_ugaussian_P((sum_az+
    gsl_cdf_ugaussian_Pinv(p[i]))/sqrt(1-sqr_a));
  }
}

void computeTwistedDependentProb(double z_g[]){
  for(int i=0;i< numbofOblgrs;i++){
    pthetaZ[i] = pZ[i]*exp(theta*c[i]) / (1 +
    pZ[i]*(exp(theta*c[i])-1));
  }
}

double computeLoss(){
  double loss = 0;
  for(int i=0;i<numbofOblgrs;i++){
    if(gsl_rng_uniform(r) < pthetaZ[i]){
      loss = loss + c[i];
    }
  }
  return loss;
}

double computeAverageProbDependent(){
  double avrg = 0;
  for(int j=0;j<numbofOblgrs;j++){
    avrg = avrg + pZ[j]/numbofOblgrs;
  }
  return avrg;
}

// Bisection method to solve theta.
double solveThetaXZ(double z_g[]){
  int counter = 0;
  double a=0,b;
  double va, vb,vc;
  double delta = 0.00001;

  double expected = 0;
  for(int i=0;i<numbofOblgrs;i++){
    expected = expected + pZ[i]*c[i];
  }
  if(expected >= x_p)
    return 0;

  va = theta_y_Equation(a);
  b = 0.01;
  while(true){
    counter++;
    vb = theta_y_Equation(b);
```

```
    if(va*vb<0){
      break;
    }
    b = 10*b;
    if(counter > 10){
      return -1;
    }
  }

  double c;

  counter = 0;
  c = (a+b)/2;
  vc = theta_y_Equation(c);
  while(fabs(vc)>delta){
    counter++;

    if(counter > 1000){
      return -1;
    }

    if(va*vc<0){
      b = c;
      vb = theta_y_Equation(b);
    }
    else{
      a = c;
      va = theta_y_Equation(a);
    }
    c = (a+b)/2;
    vc = theta_y_Equation(c);
  }
  return c;
}

double solveThetaXZ_Newton(double z_g[]){
  double p0 = 0.1;
  double p;
  double delta = 0.001;
  int count = 1;

  while(1){
    p = p0 - theta_y_Equation(p0) / derv_theta_y_Equation(p0);
    if(count == 3 || fabs(p-p0) < delta)
      break;
    count ++;
    p0 = p;
  }
  if(fabs(p-p0) < delta)
    return p;
  else
    return 0;
```

```
}

double theta_y_Equation(double theta){
  double sum = 0;
  for(int i=0;i<numbofOblgrs;i++){
    sum = sum + pZ[i]*c[i]*exp(c[i]*theta)/(1+pZ[i]*
    (exp(c[i]*theta)-1));
  }
  sum = sum - x_p;
  return sum;
}

double derv_theta_y_Equation(double theta){
  double sum = 0;
  for(int i=0;i<numbofOblgrs;i++){
    sum = sum - pow(pZ[i]*c[i],2)*exp(2*c[i]*theta)/pow((1+
    pZ[i]*(exp(c[i]*theta)-1)),2) + pZ[i]*pow(c[i],2)*exp(c[i]*theta)/
    (1+pZ[i]*(exp(c[i]*theta)-1));
  }
  return sum;
}
```

B.2. Codes For The New Algorithm For The Normal Copula Model

```
double simulateProbofExceedance(int numbSim){
  double loss;
  double prob = 0;
  double varArray [numbSim];
  double z[d];

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1);
    }
    computeDependentProb(z);

    loss = 0;
    for(int j=0;j< numbofOblgrs;j++){
      double u = gsl_rng_uniform(r);
      if(u < pZ[j]){
        loss = loss + c[j];
      }
    }

    varArray[i] = 0;
    if(loss > x_p){
      varArray[i] = 1.0;
      prob = prob + 1.0/numbSim;
    }
  }
```

```
  double variance = gsl_stats_variance(varArray, 1, numbSim);
  double confinter = 1.96 * pow(variance/numbSim, 0.5);
  double lowerLim = prob - confinter;
  double upperLim = prob + confinter;

  return variance;
}

double simulateISProbofExceedance(int numbSim){
  double loss;
  double varArray[numbSim];
  double prob = 0;
  double expectedLoss = 0;
  double z[d];
  double likelihoodRatio;

  double mu_muT = 0;
  for(int k=0;k< d;k++){
    mu_muT = mu_muT + mu[k]*mu[k];
  }

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }

    computeDependentProb(z);

    expectedLoss = 0;
    for(int k=0;k< numbofOblgrs;k++){
      expectedLoss = expectedLoss + pZ[k]*c[k];
    }

    if(expectedLoss >= x_p)
      theta = 0;
    else{
//      theta = solveThetaXZ(z);
      theta = solveThetaXZ_Newton(z);
    }

    computeTwistedDependentProb(z);

    loss = computeLoss();

    double sum = 0;
    for(int k=0;k< numbofOblgrs;k++){
      sum = sum + log(1 + pZ[k]*(exp(theta*c[k])-1));
    }
    double mu_zT = 0;
    for(int k=0;k< d;k++){
      mu_zT = mu_zT + mu[k]*z[k];
```

```
    }

    varArray[i] = 0;

    if(loss > x_p){
      likelihoodRatio = exp(-theta*loss + sum)*exp(-mu_zT+mu_muT/2);
      varArray[i] = likelihoodRatio;
      prob = prob + varArray[i]/numbSim;
    }
  }

  double variance = gsl_stats_variance(varArray, 1, numbSim);
  double confinter = 1.96 * pow(variance/numbSim, 0.5);
//  double lowerLim = prob - confinter;
//  double upperLim = prob + confinter;

  return prob;
}

double simulateProbofExceedanceOuterISInnerReplGeometric(int numbSim){
  double varArray [numbSim];
  double prob = 0;
  double avrgDepProb;
  int numInnerRepl;
  double probInside;
  double expectedAverage = 0;
  int lastPositonofDeflt;
  double z[d];

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }
    computeDependentProb(z);

    avrgDepProb = computeAverageProbDependent();

    numInnerRepl = GSL_MIN((int)(1/avrgDepProb),
    GSL_MAX(100, numbofOblgrs));

    double loss[numInnerRepl];

    probInside = 0;
    for(int n=0;n<numInnerRepl;n++){
      loss[n] = 0;
    }

    for(int j=0;j<numbofOblgrs;j++){
      lastPositonofDeflt = gsl_ran_geometric(r, pZ[j]) - 1;

      while(lastPositonofDeflt < numInnerRepl &&
      lastPositonofDeflt >= 0) {
```

```
        loss[lastPositonofDeflt] = loss[lastPositonofDeflt] + c[j];
        lastPositonofDeflt = lastPositonofDeflt +
        gsl_ran_geometric(r, pZ[j]);

      }
    }

    for(int n=0;n<numInnerRepl;n++) {
      if(loss[n] > x_p) {
        probInside = probInside + 1.0/numInnerRepl;
      }
    }

    double mu_zT = 0;
    double mu_muT = 0;
    for(int k=0;k< d;k++){
      mu_muT = mu_muT + mu[k]*mu[k];
      mu_zT = mu_zT + mu[k]*z[k];
    }

    varArray[i] = exp(-mu_zT+mu_muT/2) * probInside;
    expectedAverage = expectedAverage + exp(-mu_zT+mu_muT/2)/numbSim;
    prob = prob + varArray[i]/numbSim;

  }

  double variance = gsl_stats_variance(varArray, 1, numbSim);
  double confinter = 1.96 * pow(variance/numbSim, 0.5);
  double lowerLim = prob - confinter;
  double upperLim = prob + confinter;
  return variance;
}

double simulateNaiveExpectedShortfall(int numbSim){
  double loss;
  double varArray[numbSim];
  double expectedVal = 0;
  double z[d];
  double sumDen = 0;
  double likelihoodRatio;

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1);
    }

    computeDependentProb(z);
    loss = 0;

    for(int i=0;i<numbofOblgrs;i++){
      if(gsl_rng_uniform(r) < pZ[i]){
        loss = loss + c[i];
```

```
      }
    }


    varArray[i] = 0;

    if(loss >= x_p){
      varArray[i] = loss;
      sumDen = sumDen + 1;
      expectedVal = expectedVal + varArray[i];
    }
  }

  expectedVal = expectedVal / sumDen;

  double variance = 0;
  for(int i=0;i<numbSim;i++){
    if(varArray[i] > 0)
      variance = variance + pow(varArray[i] - expectedVal, 2);
  }

  variance = numbSim * variance / pow(sumDen, 2);

  double confinter = 1.96 * pow(variance/numbSim, 0.5);
  double lowerLim = expectedVal - confinter;
  double upperLim = expectedVal + confinter;
  return expectedVal;
}

double simulateISExpectedShortfall (int numbSim){
  double loss;
  double varArray[numbSim];
  double likelihoodRatio[numbSim];
  double expectedVal = 0;
  double z[d];
  double sumlikelihoodRatio = 0;

  double mu_muT = 0;
  for(int k=0;k< d;k++){
    mu_muT = mu_muT + mu[k]*mu[k];
  }


  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }

    computeDependentProb(z);

    double expectedLoss = 0;
    for(int k=0;k< numbofOblgrs;k++){
```

```
    expectedLoss = expectedLoss + pZ[k]*c[k];
  }

  if(expectedLoss >= x_p)
    theta = 0;
  else{
    theta = solveThetaXZ_Newton(z);
  }

  computeTwistedDependentProb(z);

  loss = computeLoss();

  varArray[i] = 0;
  likelihoodRatio[i] = 0;

  if(loss >= x_p){
    double sum = 0;
    for(int k=0;k< numbofOblgrs;k++){
      sum = sum + log(1 + pZ[k]*(exp(theta*c[k])-1));
    }

    double mu_zT = 0;
    for(int k=0;k< d;k++){
      mu_zT = mu_zT + mu[k]*z[k];
    }

    likelihoodRatio[i]=exp(-theta*loss + sum)*exp(-mu_zT+mu_muT/2);

    varArray[i] = loss;

    sumlikelihoodRatio = sumlikelihoodRatio + likelihoodRatio[i];

    expectedVal = expectedVal + varArray[i]*likelihoodRatio[i];
  }

}

expectedVal = expectedVal / sumlikelihoodRatio;

double variance = 0;
for(int i=0;i<numbSim;i++){
  if(likelihoodRatio[i] > 0)
    variance = variance+pow(likelihoodRatio[i],2)*
    pow(varArray[i]-expectedVal, 2);
}

variance = numbSim * variance / pow(sumlikelihoodRatio, 2);

double confinter = 1.96 * pow(variance/numbSim, 0.5);
double lowerLim = expectedVal - confinter;
double upperLim = expectedVal + confinter;
```

```
  return expectedVal;
}

double simulateExpectedShortfallOuterISInnerReplGeometric(int numbSim){
  double varArray [numbSim];
  double likelihoodRatio[numbSim];
  double expectedVal = 0;
  double avrgDepProb;
  int numInnerRepl;
  double expectedAverageInside;
  int lastPositonofDeflt;
  double z[d];
  double sumlikelihoodRatio = 0;

  double mu_muT = 0;
  for(int k=0;k< d;k++){
    mu_muT = mu_muT + mu[k]*mu[k];
  }

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }
    computeDependentProb(z);

    avrgDepProb = computeAverageProbDependent();

    numInnerRepl = GSL_MIN((int)(1/avrgDepProb),
    GSL_MAX(100, numbofOblgrs));

    double loss[numInnerRepl];

    for(int n=0;n<numInnerRepl;n++){
      loss[n] = 0;
    }

    for(int j=0;j<numbofOblgrs;j++){
      lastPositonofDeflt = gsl_ran_geometric(r, pZ[j]) - 1;

      while(lastPositonofDeflt < numInnerRepl &&
      lastPositonofDeflt >= 0) {
        loss[lastPositonofDeflt] = loss[lastPositonofDeflt] + c[j];
        lastPositonofDeflt = lastPositonofDeflt +
        gsl_ran_geometric(r, pZ[j]);
      }
    }

    expectedAverageInside = 0;
    for(int n=0;n<numInnerRepl;n++) {
      expectedAverageInside = expectedAverageInside +
      loss[n]/numInnerRepl;
    }
```

```
    varArray[i] = 0;
    likelihoodRatio[i] = 0;

    if(expectedAverageInside >= x_p){
      double mu_zT = 0;
      for(int k=0;k< d;k++){
        mu_zT = mu_zT + mu[k]*z[k];
      }

      likelihoodRatio[i] = exp(-mu_zT + mu_muT/2);

      varArray[i] = expectedAverageInside;

      sumlikelihoodRatio = sumlikelihoodRatio + likelihoodRatio[i];

      expectedVal = expectedVal + varArray[i]*likelihoodRatio[i];
    }
  }

  expectedVal = expectedVal / sumlikelihoodRatio;

  double variance = 0;
  for(int i=0;i<numbSim;i++){
    if(likelihoodRatio[i] > 0)
      variance = variance + pow(likelihoodRatio[i],2) *
      pow(varArray[i] - expectedVal, 2);
  }

  variance = numbSim * variance / pow(sumlikelihoodRatio, 2);

  double confinter = 1.96 * pow(variance/numbSim, 0.5);
  double lowerLim = expectedVal - confinter;
  double upperLim = expectedVal + confinter;

  return variance;
}

double expectedLossGrtrThnVaRNormalDist(double mean,
double sigma, double x){
  double xs = (x-mean)/sigma;
  double expectedVal;
  if(gsl_sf_erfc(xs/sqrt(2)) > pow(10,-40))
    expectedVal = exp(-pow(xs,2)/2) * sqrt(2/M_PI) /
    gsl_sf_erfc(xs/sqrt(2));
  else
    expectedVal = 0;

  expectedVal = sigma*expectedVal + mean;
  return expectedVal;
}
```

```
void simulateISProbofExceedanceVaR (int numbSim, double x_IS, int N){
  double varArray [N][numbSim];
  double prob[N];
  double z[d];
  double u;
  int key;
  double x[N];
  double probInside[N];
  double expectedLoss, loss, likelihoodRatio;

  // All IS distribution will be for x..
  x_p = x_IS;

  for(int l = 0; l < N; l++){
    x[l] = l*500 + 500;
    prob[l] = 0;
  }

  double mu_muT = 0;
  for(int k=0;k< d;k++){
    mu_muT = mu_muT + mu[k]*mu[k];
  }


  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }

    computeDependentProb(z);

    expectedLoss = 0;
    for(int k=0;k< numbofOblgrs;k++){
      expectedLoss = expectedLoss + pZ[k]*c[k];
    }

    if(expectedLoss >= x_p)
      theta = 0;
    else{
//      theta = solveThetaXZ(z);
      theta = solveThetaXZ_Newton(z);
    }

    computeTwistedDependentProb(z);

    loss = computeLoss();

    double sum = 0;
    for(int k=0;k< numbofOblgrs;k++){
      sum = sum + log(1 + pZ[k]*(exp(theta*c[k])-1));
    }
    double mu_zT = 0;
```

```
    for(int k=0;k< d;k++){
      mu_zT = mu_zT + mu[k]*z[k];
    }

    likelihoodRatio = exp(-theta*loss + sum)*exp(-mu_zT + mu_muT/2);

    for(int l=0;l<N;l++){
      varArray[l][i] = 0;
      if(loss > x[l]){
        varArray[l][i] = likelihoodRatio;
        prob[l] = prob[l] + varArray[l][i]/numbSim;
      }
    }
  }

  for(int l = 0; l < N; l++){

    double variance = gsl_stats_variance(varArray[l], 1, numbSim);
    double confinter = 1.96 * pow(variance/numbSim, 0.5);

    printf("x[l] %lf \n", x[l]);
    printf("prob IS %.8lf \n", prob[l]);
    printf("confinter IS %.12lf \n", confinter);
    printf("\n");

    double lowerLim = prob[l] - confinter;
    double upperLim = prob[l] + confinter;
  }

}

void simulateNaiveProbofExceedanceVaR (int numbSim, int N){
  double varArray [N][numbSim];
  double prob[N];
  double z[d];
  double x[N];
  double probInside[N];
  double loss;

  for(int l = 0; l < N; l++){
    x[l] = l*500 + 500;
    prob[l] = 0;
  }

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1);
    }
    computeDependentProb(z);

    loss = 0;
    for(int j=0;j< numbofOblgrs;j++){
```

```
      double u = gsl_rng_uniform(r);
      if(u < pZ[j]){
        loss = loss + c[j];
      }
    }

    for(int l = 0; l < N; l++){
      varArray[l][i] = 0;
      if(loss > x[l]){
        varArray[l][i] = 1.0;
        prob[l] = prob[l] + varArray[l][i]/numbSim;
      }
    }
  }

  for(int l = 0; l < N; l++){

    double variance = gsl_stats_variance(varArray[l], 1, numbSim);

    printf("x[l] %lf \n", x[l]);
    printf("prob naive %.8lf \n", prob[l]);
    printf("variance naive %.12lf \n", variance);
    printf("\n");

//    double confinter = 1.96 * pow(variance/numbSim, 0.5);
//    double lowerLim = prob[l] - confinter;
//    double upperLim = prob[l] + confinter;
  }

}

void simulateProbofExceedanceOuterIS
InnerReplGeometricDensityMixtureVaRNormalEq
(int numbSim, double xmin, double xmax, int N){
  double varArray [N][numbSim];
  double prob[N];
  double avrgDepProb;
  int numInnerRepl;
  double expectedAverage = 0;
  int lastPositonofDeflt;
  double z[d];
  double u;
  int key;
  double x[N];
  double probInside[N];

  for(int l = 0; l < N; l++){
    x[l] = l*5 + 500;
    prob[l] = 0;
  }

  for(int i=0;i<numbSim;i++){
```

```
for(int j=0;j<d;j++){
  z[j] = gsl_ran_gaussian (r, stdDevofDM[j]) + meanofDM[j];
}

computeDependentProb(z);

avrgDepProb = computeAverageProbDependent();

numInnerRepl = GSL_MIN((int)(1/avrgDepProb),
GSL_MAX(100, numbofOblgrs));


double loss[numInnerRepl];

for(int l=0;l<N;l++){
  probInside[l] = 0;
}


for(int n=0;n<numInnerRepl;n++){
  loss[n] = 0;
}

for(int j=0;j<numbofOblgrs;j++){
  lastPositonofDeflt = gsl_ran_geometric(r, pZ[j]) - 1;

  while(lastPositonofDeflt < numInnerRepl &&
  lastPositonofDeflt >= 0) {
    loss[lastPositonofDeflt] = loss[lastPositonofDeflt] + c[j];
    lastPositonofDeflt = lastPositonofDeflt +
    gsl_ran_geometric(r, pZ[j]);

  }
}

for(int n=0;n<numInnerRepl;n++) {

  for(int l=0;l<N;l++){
    if(loss[n] > x[l]) {
      probInside[l] = probInside[l] + 1.0/numInnerRepl;
    }
  }
}

double outerWeight = 1.0;
for(int k=0;k< d;k++){
  outerWeight = outerWeight * 2 / (exp(mu_min[k]*z[k] -
  0.5*mu_min[k]*mu_min[k]) +
  exp(mu_max[k]*z[k] - 0.5*mu_max[k]*mu_max[k]));
}
```

```
    for(int l=0;l<N;l++){
      varArray[l][i] = outerWeight * probInside[l];
      prob[l] = prob[l] + varArray[l][i]/numbSim;
    }
  }

  for(int k = 0; k < N; k++){

    double variance = gsl_stats_variance(varArray[k], 1, numbSim);
    printf("x[l] %lf \n", x[k]);
    printf("prob new %.8lf \n", prob[k]);
    printf("variance new %.12lf \n", variance);
    printf("\n");

    double confinter = 1.96 * pow(variance/numbSim, 0.5);
    double lowerLim = prob[k] - confinter;
    double upperLim = prob[k] + confinter;
  }

}


void densityMixtureNormalApprox(double xmin, double xmax){
  x_p = xmin;
  solveOptMeanShiftNormalApprox();
  for(int k=0;k< d;k++){
    mu_min[k] = mu[k];
  }

  x_p = xmax;
  solveOptMeanShiftNormalApprox();
  for(int k=0;k< d;k++){
    mu_max[k] = mu[k];
  }
}

void normalEquivalentofDMNormalApprox(double xmin, double xmax){
  densityMixtureNormalApprox(xmin, xmax);

  for(int k=0;k< d;k++){
    meanofDM[k] = 0.5*mu_min[k] + 0.5*mu_max[k];
    stdDevofDM[k] = pow(1 + 0.25*(pow(mu_min[k],2) +
    pow(mu_max[k],2)) - 0.5*mu_min[k]*mu_max[k], 0.5);
  }

}

void simulateISESWide_x (int numbSim, double x_IS, int N){
  double varArray[numbSim];
  double likelihoodRatio[numbSim];
  int isOver_x[N][numbSim];
  double sumlikelihoodRatio[N], expectedVal[N];
```

```
  double z[d];
  double x[N];
  double expectedLoss, loss;

  // All IS distribution will be for x..
  x_p = x_IS;

  for(int l = 0; l < N; l++){
    x[l] = l*50 + 500;
    sumlikelihoodRatio[l] = 0;
    expectedVal[l] = 0;
  }

  double mu_muT = 0;
  for(int k=0;k< d;k++){
    mu_muT = mu_muT + mu[k]*mu[k];
  }


  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }

    computeDependentProb(z);

    expectedLoss = 0;
    for(int k=0;k< numbofOblgrs;k++){
      expectedLoss = expectedLoss + pZ[k]*c[k];
    }

    if(expectedLoss >= x_p)
      theta = 0;
    else{
//      theta = solveThetaXZ(z);
      theta = solveThetaXZ_Newton(z);
    }

    computeTwistedDependentProb(z);

    loss = computeLoss();

    double sum = 0;
    for(int k=0;k< numbofOblgrs;k++){
      sum = sum + log(1 + pZ[k]*(exp(theta*c[k])-1));
    }
    double mu_zT = 0;
    for(int k=0;k< d;k++){
      mu_zT = mu_zT + mu[k]*z[k];
    }
    varArray[i] = loss;
    likelihoodRatio[i] = exp(-theta*loss + sum)*exp(-mu_zT + mu_muT/2);
```

```
    for(int l=0;l<N;l++){
      isOver_x[l][i] = 0;
      if(loss >= x[l]){
        isOver_x[l][i] = 1;

        sumlikelihoodRatio[l] = sumlikelihoodRatio[l] +
        likelihoodRatio[i];

        expectedVal[l] = expectedVal[l] + loss*likelihoodRatio[i];
      }

    }
  }

  for(int l = 0; l < N; l++){
    expectedVal[l] = expectedVal[l] / sumlikelihoodRatio[l];

    double variance = 0;
    for(int i=0;i<numbSim;i++){
      if(isOver_x[l][i] > 0)
        variance = variance + pow(likelihoodRatio[i],2) *
        pow(varArray[i] - expectedVal[l], 2);
    }

    variance = numbSim * variance / pow(sumlikelihoodRatio[l], 2);
    double confinter = 1.96 * pow(variance/numbSim, 0.5);

    printf("x[l] %lf \n", x[l]);
    printf("ES IS %.8lf \n", expectedVal[l]);
    printf("confinter IS %.12lf \n", confinter);
    printf("\n");

    double lowerLim = expectedVal[l] - confinter;
    double upperLim = expectedVal[l] + confinter;
  }
}

void simulateESOuterISInnerReplGeometricWide_x (int numbSim, int N){
  double varArray[numbSim];
  double likelihoodRatio[numbSim];
  int isOver_x[N][numbSim];
  double sumlikelihoodRatio[N], expectedVal[N];
  double z[d];
  double x[N];
  double expectedLoss, loss;
  int numInnerRepl;
  double expectedAverageInside, avrgDepProb;
  int lastPositonofDeflt;

  for(int l = 0; l < N; l++){
    x[l] = l*50 + 500;
```

```
    sumlikelihoodRatio[l] = 0;
    expectedVal[l] = 0;
  }


  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, stdDevofDM[j]) + meanofDM[j];
    }

    computeDependentProb(z);

    avrgDepProb = computeAverageProbDependent();

    numInnerRepl = GSL_MIN((int)(1/avrgDepProb),
    GSL_MAX(100, numbofOblgrs));

    double loss[numInnerRepl];

    for(int n=0;n<numInnerRepl;n++){
      loss[n] = 0;
    }

    for(int j=0;j<numbofOblgrs;j++){
      lastPositonofDeflt = gsl_ran_geometric(r, pZ[j]) - 1;

      while(lastPositonofDeflt < numInnerRepl &&
      lastPositonofDeflt >= 0) {
        loss[lastPositonofDeflt] = loss[lastPositonofDeflt] + c[j];
        lastPositonofDeflt = lastPositonofDeflt +
        gsl_ran_geometric(r, pZ[j]);
      }
    }

    expectedAverageInside = 0;
    for(int n=0;n<numInnerRepl;n++) {
      expectedAverageInside = expectedAverageInside +
      loss[n]/numInnerRepl;
    }

    double outerWeight = 1.0;
    for(int k=0;k< d;k++){
      outerWeight = outerWeight * 2 / (exp(mu_min[k]*z[k] -
      0.5*mu_min[k]*mu_min[k]) + exp(mu_max[k]*z[k] -
      0.5*mu_max[k]*mu_max[k]));
    }

    likelihoodRatio[i] = outerWeight;
    varArray[i] = expectedAverageInside;

    for(int l=0;l<N;l++){
      isOver_x[l][i] = 0;
```

```
      if(expectedAverageInside >= x[l]){
        isOver_x[l][i] = 1;

        sumlikelihoodRatio[l] = sumlikelihoodRatio[l] +
        likelihoodRatio[i];

        expectedVal[l] = expectedVal[l] +
        expectedAverageInside*likelihoodRatio[i];
      }

    }
  }

  for(int l = 0; l < N; l++){
    expectedVal[l] = expectedVal[l] / sumlikelihoodRatio[l];

    double variance = 0;
    for(int i=0;i<numbSim;i++){
      if(isOver_x[l][i] > 0)
        variance = variance + pow(likelihoodRatio[i],2) *
        pow(varArray[i] - expectedVal[l], 2);
    }

    variance = numbSim * variance / pow(sumlikelihoodRatio[l], 2);
    double confinter = 1.96 * pow(variance/numbSim, 0.5);

    printf("x[l] %lf \n", x[l]);
    printf("ES new %.8lf \n", expectedVal[l]);
    printf("confinter new %.12lf \n", confinter);
    printf("\n");

    double lowerLim = expectedVal[l] - confinter;
    double upperLim = expectedVal[l] + confinter;
  }
}
```

B.3. Codes For The Comparison of Mean Shifts For IS

```
int solveOptMeanShiftNormalApprox(){

    size_t np = d;
    double par[d] = {1.0};

    const gsl_multimin_fminimizer_type *T =
    gsl_multimin_fminimizer_nmsimplex;

    gsl_multimin_fminimizer *s = NULL;
    gsl_vector *ss, *x;
    gsl_multimin_function minex_func;
```

```
    size_t iter = 0, i;
    int status;
    double size;

    /* Initial vertex size vector */
    ss = gsl_vector_alloc (np);

    /* Set all step sizes to 1 */
    gsl_vector_set_all (ss, 1.0);

    /* Starting point */
    x = gsl_vector_alloc (np);

    gsl_vector_set_all (x, 2);
//    gsl_vector_set (x, 0, 2.2);

    /* Initialize method and iterate */
    minex_func.f = &my_f;
    minex_func.n = np;
    minex_func.params = (void *)&par;

    s = gsl_multimin_fminimizer_alloc (T, np);
    gsl_multimin_fminimizer_set (s, &minex_func, x, ss);

    do
    {
      iter++;
      status = gsl_multimin_fminimizer_iterate(s);

      if (status)
          break;

    size = gsl_multimin_fminimizer_size (s);
      status = gsl_multimin_test_size (size, 1e-2);

      if (status == GSL_SUCCESS)
      {
//    printf ("converged to minimum at\n");
      }

  }
    while (status == GSL_CONTINUE && iter < 10000);

    for (i = 0; i < np; i++)
    {
    mu[i] = gsl_vector_get (s->x, i);
    }

    gsl_vector_free(x);
    gsl_vector_free(ss);
    gsl_multimin_fminimizer_free (s);
```

```
    return status;
}

double objectiveFunctionZ (const gsl_vector *v, void *params) {
  double z_g[d];
  double sum = 0;

  for(int j=0;j< d;j++){
    z_g[j] = gsl_vector_get(v, j);
  }

  computeDependentProb(z_g);
//  double theta = solveThetaXZ_Newton(z_g);
  double theta = solveThetaXZ(z_g);

  double sumsqrz = 0;

  for(int i=0;i< numbofOblgrs;i++){
    sum = sum + log(1 + pZ[i]*(exp(theta*c[i])-1));
  }

  for(int i=0;i< d;i++){
    sumsqrz = sumsqrz + pow(z_g[i], 2);
  }

  return theta*x_p - sum + 0.5*sumsqrz;
}

int solveOptMeanShiftGlassLi(){

  size_t np = d;
  double par[d] = {1.0};

   const gsl_multimin_fminimizer_type *T =
     gsl_multimin_fminimizer_nmsimplex;

      gsl_multimin_fminimizer *s = NULL;
  gsl_vector *ss, *x;
  gsl_multimin_function minex_func;

  size_t iter = 0, i;
  int status;
  double size;

      /* Initial vertex size vector */
      ss = gsl_vector_alloc (np);


    /* Set all step sizes to 1 */
  gsl_vector_set_all (ss, 1.0);

      /* Starting point */
```

```
      x = gsl_vector_alloc (np);

    gsl_vector_set_all (x, 2);

          /* Initialize method and iterate */
          minex_func.f = &objectiveFunctionZ;
          minex_func.n = np;
          minex_func.params = (void *)&par;

  s = gsl_multimin_fminimizer_alloc (T, np);
  gsl_multimin_fminimizer_set (s, &minex_func, x, ss);

  do
      {
    iter++;
        status = gsl_multimin_fminimizer_iterate(s);

      if (status)
            break;

        size = gsl_multimin_fminimizer_size (s);
              status = gsl_multimin_test_size (size, 1e-2);

      }
  while (status == GSL_CONTINUE && iter < 10000);

  for (i = 0; i < np; i++)
  {
    mu[i] = gsl_vector_get (s->x, i);
      }

  gsl_vector_free(x);
  gsl_vector_free(ss);
  gsl_multimin_fminimizer_free (s);

  return status;
}

double my_f (const gsl_vector *v, void *params) {
  double *dp = (double *)params;
  double z_g[d];
  double A = 0 , B = 0;
  double f_z = 1;


  for(int j=0;j< d;j++){
    z_g[j] = gsl_vector_get(v, j);
//     printf("%lf,\n", z_g[j]);

  }

  computeDependentProb(z_g);
```

```
  for(int i=0;i< numbofOblgrs;i++){
    A = A + c[i]*pZ[i];
    B = B + pow(c[i], 2) * (pZ[i] - pow(pZ[i], 2));
  }

//  printf("A %lf B %lf \n", A, B);

  for(int j=0;j< d;j++){
    f_z = f_z * gsl_ran_gaussian_pdf(z_g[j],1);
  }

  return -(1-gsl_cdf_ugaussian_P((x_p - A) / pow(B,0.5))) * f_z;
}

double returnCovariance(int i, int j){
  if(i == j){
    return 1;
  }
  else{
    return 0;
  }
}

double returnCovarianceOrg(int i, int j){
  double sum = 0;
  if(i == j){
    return 1;
  }
  else{
    sum = 0;
    for(int k=0;k< d;k++){
      sum = sum + a[i][k]*a[j][k];
    }
    return sum;
  }
}


double returnCovarianceHom(int i, int j){
  double sum = 0;
  if(i == j){
    return 1;
  }
  else{
    sum = 0;
    for(int k=0;k< d;k++){
      sum = sum + pow(a_h[k], 2);
    }
    return sum;
  }
}
```

```
void computePsi(){
  for(int k=0;k< d;k++){
    psi[k] = 0;
    for(int i=0;i< numbofOblgrs;i++){
      psi[k] = psi[k] + p[i]*c[i]*a[i][k];
    }
  }
}

void computeRSquare(){
  computePsi();
  double sum1 = 0;
  for(int i=0;i< d;i++){
    for(int j=0;j< d;j++){
      sum1 = sum1 + psi[i]*psi[j]*returnCovariance(i, j);
    }
  }

  double sum2 = 0;
  for(int i=0;i< numbofOblgrs;i++){
    sum2 = sum2 + pow(p[i]*c[i], 2) * RiSquare[i];
  }

  double sum3 = 0;
  for(int i=0;i< numbofOblgrs;i++){
    sum3 = sum3 + pow(p[i]*c[i], 2);
  }

  double sum4 = 0;
  for(int i=0;i< numbofOblgrs;i++){
    sum4 = sum4 + p[i]*c[i];
  }
  sum4 = pow(sum4, 2);

  RSquare = (sum1 - sum2) / (sum4 - sum3);
  printf("RSquare %lf \n", RSquare);
}

void computeS(){
  computeRSquare();
  double sum = 0;
  for(int i=0;i< d;i++){
    for(int j=0;j< d;j++){
      sum = sum + psi[i]*psi[j]*returnCovariance(i, j);
    }
  }
  sum = sum / RSquare;
  s = pow(sum, 0.5);
  printf("s %lf \n", s);
}
```

```
void computeHomFactorLoadings(){
  computeS();
  for(int k=0;k< d;k++){
    a_h[k] = 0;
    for(int i=0;i< numbofOblgrs;i++){
      a_h[k] = a_h[k] + p[i]*c[i]*a[i][k];
    }
    a_h[k] = a_h[k] / s;
  }
}

void homogenousPortEquivalent(){
  double sum_c;
  c_h = 0;
  for(int i=0;i< numbofOblgrs;i++){
    c_h = c_h + c[i] / numbofOblgrs;
    sum_c = sum_c + c[i];
  }

  p_h = 0;
  for(int i=0;i< numbofOblgrs;i++){
    p_h = p_h + p[i]*c[i] / sum_c;
  }

  computeHomFactorLoadings();

}

double functionToIntegrate(double x, void * params){
  return pow(c_h * gsl_cdf_ugaussian_P((gsl_cdf_ugaussian_Pinv(p_h) +
  pow(RSquare, 0.5)*x)/sqrt(1-RSquare))*gsl_ran_gaussian_pdf(x,1),2)/
  gsl_ran_gaussian_pdf(x-mu1,1);
}

double integrateExpectedShortfallforHomPortf (const gsl_vector *v,
void *params) {
  double *dp = (double *)params;
  mu1 = gsl_vector_get(v, 0);

  gsl_integration_workspace * w =
  gsl_integration_workspace_alloc (1000);
  double result, error;
  double alpha = 1.0;

  gsl_function F;
  F.function = &functionToIntegrate;
  F.params = &alpha;
  gsl_integration_qags(&F, gsl_cdf_ugaussian_Pinv(VarProb),
  5, 0, 1e-4, 1000, w, &result, &error);
  gsl_integration_workspace_free (w);
  return result;
}
```

```
double simVarianceHomPort (const gsl_vector *v, void *params) {
  double *dp = (double *)params;
  double loss;
  double z_h, likelihoodRatio, mu_zT;
//one dimensional mean shift
  mu1 = gsl_vector_get(v, 0);
  int numbSim = 1000000;
  double varArray[numbSim];
  double prob = 0;

  double mu_muT = mu1*mu1;

  for(int i=0;i<numbSim;i++){
    z_h = gsl_ran_gaussian (r, 1) + mu1;
    loss = c_h * gsl_cdf_ugaussian_P((gsl_cdf_ugaussian_Pinv(p_h) +
    pow(RSquare, 0.5)*z_h)/sqrt(1-RSquare));

    varArray[i] = 0;
    if(z_h > gsl_cdf_ugaussian_Pinv(VarProb)){
      mu_zT = mu1*z_h;
      likelihoodRatio = exp(-mu_zT+mu_muT/2)*loss;
      varArray[i] = likelihoodRatio;
      prob = prob + varArray[i]/numbSim;
    }
  }
  double variance = gsl_stats_variance(varArray, 1, numbSim);
  return variance;
}

int solveforOneDimShiftHomogenousPortfolio(){

    size_t np = 1;
    double par[1] = {1.0};

    const gsl_multimin_fminimizer_type *T =
    gsl_multimin_fminimizer_nmsimplex;
    gsl_multimin_fminimizer *s = NULL;
    gsl_vector *ss, *x;
    gsl_multimin_function minex_func;

    size_t iter = 0, i;
    int status;
    double size;

    /* Initial vertex size vector */
    ss = gsl_vector_alloc (np);

    /* Set all step sizes to 1 */
    gsl_vector_set_all (ss, 0.01);

    /* Starting point */
```

```
    x = gsl_vector_alloc (np);

    gsl_vector_set_all (x, 1);
//    gsl_vector_set (x, 0, 2.2);

    /* Initialize method and iterate */
//    minex_func.f = &simVarianceHomPort;
    minex_func.f = integrateExpectedShortfallforHomPortf;
    minex_func.n = np;
    minex_func.params = (void *)&par;

    s = gsl_multimin_fminimizer_alloc (T, np);
    gsl_multimin_fminimizer_set (s, &minex_func, x, ss);

    do
    {
      iter++;
      status = gsl_multimin_fminimizer_iterate(s);

      if (status)
          break;

    size = gsl_multimin_fminimizer_size (s);
    status = gsl_multimin_test_size (size, 1e-5);
    }
    while (status == GSL_CONTINUE && iter < 10000);

    gsl_vector_free(x);
    gsl_vector_free(ss);
    gsl_multimin_fminimizer_free (s);

    return status;
}

void computeMultiDimShiftKalkbrenerHomPort (){
  homogenousPortEquivalent();
  solveforOneDimShiftHomogenousPortfolio();
  for(int i=0;i< d;i++){
    mu[i] = mu1 * a_h[i]  / pow(RSquare, 0.5);
    printf("%lf\n", mu[i]);
  }

}
```

B.4. Codes For The Better Confidence Intervals For IS

```
double thirdroot(double x){
  return GSL_SIGN (x)*pow(fabsf(x),1.0/3);
}

double inverseTransform(double x,double a, double b,
```

```
double skewness,int numbSim){
  double res;
  res = pow(numbSim,0.5)*1.0/(a*skewness)*(thirdroot(1+
  3*a*skewness*(pow(numbSim,-0.5)*x-1.0/numbSim*b*skewness))-1);
  return res;
}

gsl_vector * upperEndLowerEndConfInt (int n, gsl_vector *sample,
double exact, double beta){
  gsl_vector *res = gsl_vector_alloc(2);
  gsl_vector_set_zero(res);
  double sigma = pow(gsl_stats_variance(sample->data, 1, n),0.5);
  double skewness = gsl_stats_skew(sample->data,1,n);
  double temp  = inverseTransform(gsl_cdf_gaussian_Pinv(1-beta,1),
  1.0/3,1.0/6,skewness,n);
  double bound = gsl_stats_mean(sample->data, 1, n)-
  pow(n,-0.5)*sigma*temp;

  if(exact<bound){
    gsl_vector_set(res, 0, 1);
  }
  temp  = inverseTransform(gsl_cdf_gaussian_Pinv(beta,1),
  1.0/3,1.0/6,skewness,n);
  bound = gsl_stats_mean(sample->data,1,n)-pow(n,-0.5)*sigma*temp;

  if(exact>bound){
    gsl_vector_set(res, 1, 1);
  }
  return res;
}

gsl_vector * simulatemanyLossProb(int m, int n, double exact,
double beta){
  gsl_vector *res, *resUp, *resLower, *sample, *finResult, *restStat;
  resUp = gsl_vector_alloc(m);
  resLower = gsl_vector_alloc(m);
  restStat = gsl_vector_alloc (m);

  finResult = gsl_vector_alloc(4);

  for(int i=0;i<m;i++){
    //common sample for both methods.
    sample = simulateISProbofExceedanceforCI(n);
    //tstat
    gsl_vector_set(restStat, i, (gsl_stats_mean(sample->data, 1, n)-
    exact)/(gsl_stats_sd(sample->data, 1, n)/sqrt(n)));
    //Hall
    res = upperEndLowerEndConfInt(n,sample,exact,beta);
    gsl_vector_set(resUp,i,gsl_vector_get(res, 0));
    gsl_vector_set(resLower,i,gsl_vector_get(res, 1));
    gsl_vector_free(sample);
    gsl_vector_free(res);
```

```
  }
  //tstat
  double resL  = 0;
  double resU  = 0;
  for(int l=0;l<m;l++){
    if(gsl_vector_get(restStat,l) < gsl_cdf_gaussian_Pinv(0.95,1))
      resL = resL + 1.0 / m;
    if(gsl_vector_get(restStat,l) > gsl_cdf_gaussian_Pinv(0.05,1))
      resU = resU + 1.0 / m;
  }
  gsl_vector_set(finResult,0,resU);
  gsl_vector_set(finResult,1,gsl_stats_mean(resUp->data,1,m));
  gsl_vector_set(finResult,2,resL);
  gsl_vector_set(finResult,3,gsl_stats_mean(resLower->data,1,m));

  gsl_vector_free(resLower);
  gsl_vector_free(resUp);
  gsl_vector_free(restStat);
  return finResult;

}

gsl_vector * simulateISProbofExceedanceforCI(int numbSim){
  gsl_vector *res;
  res = gsl_vector_alloc(numbSim);
  double loss;
  double expectedLoss = 0;
  double z[d];
  double likelihoodRatio;

  double mu_muT = 0;
  for(int k=0;k< d;k++){
    mu_muT = mu_muT + mu[k]*mu[k];
  }

  for(int i=0;i<numbSim;i++){
    for(int j=0;j<d;j++){
      z[j] = gsl_ran_gaussian (r, 1) + mu[j];
    }

    computeDependentProb(z);

    expectedLoss = 0;
    for(int k=0;k< numbofOblgrs;k++){
      expectedLoss = expectedLoss + pZ[k]*c[k];
    }

    if(expectedLoss >= x_p)
      theta = 0;
    else{
//      theta = solveThetaXZ(z);
      theta = solveThetaXZ_Newton(z);
```

```
    }

    computeTwistedDependentProb(z);

    loss = computeLoss();

    double sum = 0;
    for(int k=0;k< numbofOblgrs;k++){
      sum = sum + log(1 + pZ[k]*(exp(theta*c[k])-1));
    }
    double mu_zT = 0;
    for(int k=0;k< d;k++){
      mu_zT = mu_zT + mu[k]*z[k];
    }

    gsl_vector_set(res, i, 0);
    if(loss > x_p){
      likelihoodRatio = exp(-theta * loss + sum) *
       exp(-mu_zT+mu_muT/2);
      gsl_vector_set(res, i, likelihoodRatio);
    }
  }

  double beta = 1-0.05;

  double prob = gsl_stats_mean(res->data, 1, numbSim);
  double variance = gsl_stats_variance(res->data, 1, numbSim);
  double confinter = gsl_cdf_gaussian_Pinv(beta,1) *
  pow(variance/numbSim, 0.5);
  double lowerLim = prob - confinter;
  double upperLim = prob + confinter;

  printf("prob GL %.8lf \n", prob);
  printf("upper bound t %.8lf \n", upperLim);
  printf("lower bound t %.8lf \n", lowerLim);

  // Hall CI
  double sigma = pow(gsl_stats_variance(res->data, 1, numbSim),0.5);
  double skewness = gsl_stats_skew(res->data,1,numbSim);

  double temp  = inverseTransform(gsl_cdf_gaussian_Pinv(1-beta,1),
  1.0/3,1.0/6,skewness,numbSim);
  double upBound = gsl_stats_mean(res->data, 1, numbSim)-
  pow(numbSim,-0.5)*sigma*temp;

  temp  = inverseTransform(gsl_cdf_gaussian_Pinv(beta,1),
  1.0/3,1.0/6,skewness,numbSim);
  double lowBound = gsl_stats_mean(res->data,1,numbSim)-
  pow(numbSim,-0.5)*sigma*temp;

  return res;
}
```

REFERENCES

1. Artzner, P., F. Delbaen, J. M. Eber and D. Heath, 1997, "Thinking coherently", *Risk*, Vol. 10, No. 11, pp. 68–71.

2. Artzner, P., F. Delbaen, J. M. Eber and D. Heath, 1999, "Coherent measures of risk", *Mathematical Finance*, Vol. 9, No. 3, pp. 203–228.

3. Basel Committee On Banking Supervision, 2005, "International Convergence of Capital Measurement and Capital Standards". www.bis.org.

4. Bluhm, C., L. Overbeck and C. Wagner, 2003, *An Introduction to Credit Risk Modeling*, Chapman & Hall/CRC.

5. Credit Suisse Financial Products, 1997, *CreditRisk$^+$: A CreditRisk Management Framework*, London.

6. Duffie, D. and. K. Singleton, 2003, *Credit Risk: Pricing, Measurement and Management*, Princeton University Press, Princeton.

7. Egloff, D., M. Leippold, S. Jöhri and C. Dalbert, 2005, "Optimal Importance Sampling for Credit Portfolios with Stochastic Approximation", *Working Paper*, No. 217, National Centre of Competence in Research Financial Valuation and Risk Management.

8. Frey, R., A. J. McNeil and M. Nyfeler, 2001, "Copulas and credit models", *RISK*, Vol. 8, pp. 111–114.

9. Galassi, M., J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth and F. Rossi, 2003, *GNU Scientific Library Reference Manual* (2nd Ed.), Network Theory Ltd.

10. Geweke, J., 1989, "Bayesian Inference in Econometric Models Using Monte Carlo Integration", *Econometrica*, Vol. 57, No. 6, pp. 1317–1339.

11. Glasserman, P., 2004, *Monte Carlo Methods in Financial Engineering*, Springer, New York.

12. Glasserman, P., 2005, "Measuring marginal risk contributions in credit portfolios", *Journal of Computational Finance*, Vol. 9, No. 2, pp. 1–41.

13. Glasserman, P. and J. Li, 2005, "Importance Sampling for Portfolio Credit Risk", *Management Science*, Vol. 51, pp. 1643–1656.

14. Glasserman, P., P. Heidelberger and P. Shahabuddin, 1999, "Asymptotically optimal importance sampling and stratication for pricing path dependent options", *Mathematical Finance*, Vol. 9, No. 2, pp. 117–152.

15. Glasserman, P. and S. Juneja, 2007, "Uniformly Efficient Importance Sampling for the Tail Distribution of Sums of Random Variables", *Mathematics of Operations Research*, Vol. 33, No. 1, pp. 36–50.

16. Glasserman, P., W. Kang and P. Shahabuddin, 2007, Fast Simulation of Multifactor Portfolio Credit Risk, *Working Paper*, Graduate School of Business, Columbia University.

17. Grundke, P., 2005, "Risk Measurement with Integrated Market and Credit Portfolio Models", *Journal of Risk*, Vol. 7, No. 3, pp. 63–94.

18. Grundke, P., 2006, "Importance Sampling for Integrated Market and Credit Portfolio Models", *Working Paper*, Department of Banking, University of Cologne.

19. Gupton, G., C. Finger and M. Bhatia, 1997, "CreditMetrics technical document", J.P. Morgan & Co., New York.

20. Gürtler, M., M. Hibbeln and C. Vöhringer, 2008, "Adjusting Multi-Factor Models for Basel II-Consistent Economic Capital", *Working Paper*, Technical University at Braunschweig Institute for Economics and Business Administration.

21. Hall, P., 1992, "On the removal of skewness by transformation", *Journal of the Royal Statistical Society*, Vol. 54, pp. 221–228.

22. Hesterberg, T., 1995, "Weighted Average Importance Sampling and Defensive Mixture Distributions", *Technometrics*, Vol. 37, No. 2, pp. 185-194.

23. Johnson, N. J., 1978, "Modied t tests and condence intervals for asymmetrical populations", *Journal of the American Statistical Association*, Vol. 73, No. 363, pp. 536–544.

24. Kalkbrener, M., A. Kennedy and M. Popp, 2007, "Efficient Calculation of Expected Shortfall Contributions in Large Credit Portfolios", *Journal of Computational Finance*, Vol. 11, pp. 1-33.

25. Kalkbrener, M., H. Lotter and L. Overbeck, 2004, "Sensible and efficient capital allocation for credit portfolios", *Risk*, Vol. 17, No. 1, pp. 19–24.

26. Kiesel, R., W. Perraudin and A. Taylor, 2003, "The structure of credit risk: spread volatility and ratings transitions", *Journal of Risk*, Vol. 6, No. 1, pp. 1–27.

27. McKinsey & Company, German Office, 2001, *CreditPortfolio View 2.0*, Technische Dokumentation.

28. Merino, S. and M. Nyfeler, 2004, "Applying importance sampling for estimating coherent credit risk contributions", *Quantitative Finance*, Vol. 4, pp. 199–207.

29. Merton, R. C., 1974, "On the pricing of corporate debt: The risk structure of interest rates", *Journal of Finance*, Vol. 29, No. 2, pp. 449–470.

30. R Development Core Team, 2007, "R: A language and environment for statistical computing", R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

31. Ross, S., 1997, *Simulation*, New York: Academic, pp. 167–78.

32. Wolfram Research, Inc., 2007, *Mathematica, Version 6.0*, Champaign, IL.

33. Yamai, Y. and T. Yoshiba, 2005, "Value-at-risk versus expected shortfall: A practical perspective", *Journal of Banking & Finance*, Vol. 29, No. 4, pp. 997–1015.

34. Zhou X. H. and S. Gao, 2000, "One-sided condence intervals for means of positively skewed distributions", *The American Statistician*, Vol. 54, No. 2, pp. 100–104.

REFERENCES NOT CITED

Bentkus, V. and F. Gotze, 1996, "The Berry-Esseen bound for student's statistic", *The Annals of Probability*, Vol. 24, No. 1, pp. 491–503.

Boyle, P., M. Broadie and P. Glasserman, 1997, "Monte Carlo methods for security pricing", *J. Econom. Dynam. Control*, Vol.21, No. 8–9, pp. 1267–1321.

Broadie, M. and P. Glasserman, 1997, "Pricing American-style securities by simulation", *Journal of Economic Dynamics and Control*, Vol. 21, No. 8–9, pp. 1323–1352.

Broadie, M., P. Glasserman and G. Jain, 1997, "Enhanced Monte Carlo Estimates for American Option Prices", *Journal of Derivatives*, Vol. 5, pp. 25–44.

Broadie, M., P. Glasserman, and S. Kou, 1997, "A Continuity Correction for Discrete Barrier Options", *Mathematical Finance*, Vol. 7, pp. 325–349.

Broadie, M., P. Glasserman and Z. Ha, 1997, "Pricing American Options by Simulation Using a Stochastic Mesh with Optimized Weights", *Probabilistic Constrained Optimization: Methodology and Applications* (S. P. Uryasev, Editor), Kluwer Academic Publishers, Norwell, Mass.

Bucklew, J. A., 2004, *Introduction to Rare Event Simulation*, Springer Series in Statistics.

Frey, R. and A. J. McNeil, 2003, "Dependent defaults in models of portfolio credit risk", *Journal of Risk*, Vol. 6, No. 1, pp. 59–92.

Frey, R. and A. J. McNeil, 2002, "VaR and expected shortfall in portfolios of dependent credit risks: conceptual and practical insights", *Journal of Banking and Finance*, Vol. 26, pp. 1317–1334.

Glasserman, P. and J. Li, 2003, "Importance Sampling for a Mixed Poisson Model of Portfolio Credit Risk", *Proceedings of the 2003 Winter Simulation Conference*, pp. 267–275.

Hörmann, W., J. Leydold and G. Derflinger, 2004, *Automatic Non-Uniform Random Variate Generation*, Springer-Verlag, Berlin Heidelberg.

Johns, M. V., 1988, "Importance Sampling for Bootstrap Confidence Intervals", *Journal of the American Statistical Association*, Vol. 83, No. 403, pp. 709–714.

Kalkbrener, M., A. Kennedy and M. Popp, 2007, "Efficient Calculation of Expected Shortfall Contributions in Large Credit Portfolios, *Journal of Computational Finance*, Vol. 11, No. 2, pp. 45–78.

Longstaff, F. A. and E. S. Schwartz, 2001, "Valuing American Options by Simulation: Simple Least-Squares Approach", *Review of Financial Studies*, Vol. 14, pp. 113–147.

Merino, S. and M. Nyfeler, 2002, "Calculating portfolio loss", *Risk*, August 2002, pp. 82-86.

Moody's Investors Service, 2002, *Default and Recovery Rates of Corporate Bond Issuers, A Statistical Review of Moody's Ratings Performance 1970-2001. Special Comment*, New York.

Overton, M. L., 2001, *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia.

Ritchken, P., 1995, "On Pricing Barrier Options", *Journal of Derivatives*, Vol. 3, No. 2, pp. 19–28.

Smith, P. J., 2001, "Underestimation of rare event probabilities in importance sampling simulations", *SIMULATION*, Vol. 76, No. 3, pp. 140–150.

Xiao, J. Y., 2000, "Importance sampling for credit portfolio simulation", *RiskMetrics Journal*, Vol. 2, No. 2, pp. 23–28.

Yamai, Y. and T. Yoshiba, 2002, "Comparative analyses of expected shortfall and VaR (2): Expected utility maximization and tail risk", *Monetary and Economic Studies*, Vol. 20, No. 2, pp. 95–115.