MASTER'S DEGREE THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: Portfolio Credit Risk: Models and Numerical Methods

Author: Guillermo Navas Palencia

Advisor: Argimiro Arratia Quesada

Department: Department of Computer Science (LARCA)

University: Universitat Politècnica de Catalunya

Academic year: 2015/2016

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master's Thesis
Master in Statistics and Operations Research

Portfolio Credit Risk: Models and Numerical Methods

Guillermo Navas Palencia

Advisor: Argimiro Arratia Quesada

Department of Computer Science (LARCA) and Barcelona Graduate School of Mathematics (BGSMath)

Preface

After my time in Frankfurt working at Moody's and these months writing this thesis, I am pleased to have acquired the required knowledge to complete this work in portfolio credit risk and I hope I have successfully shown the importance of this area in the financial industry and the possible improvements to increase the reliability of the methods within the pages of this thesis.

I owe a great deal of thanks to my thesis supervisor, Argimiro Arratia. He immediately became interested in the topic I suggested and accepted my research proposal. He has devoted many hours and invaluable guidance throughout the development on this work and definitely it has been a pleasure to work on this thesis with his supervision.

I would also like to thank Luis Ortiz from Centre de Recerca Matemàtica for his master class about Haar wavelet approximation, the interesting discussions about possible enhancements and for providing me his implementation of the method.

Most of the members of my family have been very supportive, especially my mother. I appreciate the fruitful discussions with them helping me to clarify my own ideas and take the right decisions.

Last, but no least, I thank the Numerical Algorithms Group, which provided me flexibility to moving to Oxford and start my new job, giving me time to finish the writing of this work.

Oxford, December 2015
Guillermo Navas Palencia

Abstract

The purpose of this thesis is the study of portfolio credit risk models and the numerical methods applied for their computation. Portfolios credit risk models are used for quantifying the portfolio loss distribution and risk measures, such as Value-at-Risk and Expected Shortfall.

The Vasicek one-factor model will provide a point of departure, allowing us to study its generalization and the development of a numerical method for its computation. Subsequently, we present the large portfolio approximation and its generalization. All these credit risk methodologies and especially their generalizations will require the use of advanced and efficient numerical methods whose implementation will be explained in detail. Therefore, we initially focus on the efficiency and accuracy of numerical methods for the computation of several statistical distributions and special functions. Furthermore, we include other more sophisticated methodologies, such as the Fourier transform method or the Haar wavelet approximation, which consider portfolios with exposure concentrations and loss given default. The former might be viewed as a general-purpose methodology, which can be applied on the computation of the portfolio loss distribution, whereas the latter is intended for the computation of risk measures. A detailed study of their respective implementations and computational improvements will be presented for both methodologies.

Finally, we present a comparative study of methods, using a set of portfolios with different characteristics in order to identify the most appropriate method for each case. The results reveal that there is no ideal method applicable for all portfolios but rather that different methods need to be applied for different portfolios.

Keywords: credit risk, quantitative finance techniques, mathematical finance, risk measures, computation of special functions, oscillatory functions, numerical quadrature

MSC2000: 30E20, 33C10, 33C15, 33F05, 41A10, 42A38, 44A10, 62Gxx, 62P05, 65D20, 65D25, 65D30, 65D32, 68M20, 68W25, 91B28, 91B30

Resumen

El objetivo de esta tesis es el estudio de modelos de riesgo de crédito en carteras y los métodos numéricos aplicados para su cálculo. Los modelos de riesgo de crédito en carteras son empleados para cuantificar la distribución de pérdidas de la cartera y medidas de riesgo, tales como el Valor en Riesgo (VaR) y la Pérdida Esperada (ES).

El modelo de Vasicek de un factor nos proporcionará un punto de partida, permitiéndonos el estudio de su generalización y el desarrollo de un método numérico para su cómputo. Seguidamente, presentamos la aproximación para grandes carteras y su generalización. Todas estas metodologías para la cuantificación del riesgo de crédito y especialmente sus generalizaciones, requerirán el uso de métodos numéricos eficientes y precisos cuya implementación será explicada en detalle. Por consiguiente, nos centraremos inicialmente en la eficacia y precisión de métodos numéricos para el cálculo de varias distribuciones estadísticas y de funciones especiales. Además, incluimos otras metodologías de riesgo de crédito más sofisticadas, tales como el método de la transformada de Fourier o la aproximación por medio de wavelets de Haar, que consideran carteras con concentraciones de exposición y severidad. El primer método puede ser visto como una metodología de propósito general, la cual puede aplicarse para el cálculo de la distribución de pérdidas en la cartera, mientras que el segundo está destinado al cálculo de medidas de riesgo. Un estudio detallado y sus respectivas implementaciones y mejoras computacionales serán presentados para ambas metodologías.

Finalmente presentamos un estudio comparativo de los métodos, empleando un conjunto de carteras con diferentes características con el fin de identificar el método más apropiado en cada caso. Los resultados revelan que no existe un método ideal aplicable a todos los portfolios, sino que métodos distintos deben ser aplicados en diferentes carteras.

Palabras clave: riesgo de crédito, técnicas financieras cuantitativas, matemáticas financieras, medidas de riesgo, cálculo de funciones especiales, funciones oscilatorias, cuadratura numérica

MSC2000: 30E20, 33C10, 33C15, 33F05, 41A10, 42A38, 44A10, 62Gxx, 62P05, 65D20, 65D25, 65D30, 65D32, 68M20, 68W25, 91B28, 91B30

Notation

Spaces

$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers

Complex plane

$\Re, \Im$	Real, imaginary part
$\bar{x}$	Complex conjugate

Special functions

J_ν, K_ν	Bessel function order ν
erfc	Complementary erf
Γ	Gamma function
P_k	kth Legendre polynomial
H_k	kth Hermite polynomial
erf	Error function
erfi	Imaginary error function
$(\mu)_j$	Pochhammer symbol
L_k	kth Laguerre polynomial
${}_1F_1$, ${}_2F_0$	Hypergeometric functions

Transforms

$\hat{f}, \mathcal{F}\{f\}$	Fourier transform of f
$\mathcal{L}\{f\}$	Laplace transform of f

Asymptotic

$f \sim \sum_{j=0}^{\infty} \cdots$	Asymptotic expansion
$f(x) \sim g(x), x \to a$	f is asymptotically equal to g as x approaches a
$\mathcal{O}(\cdot)$	Big-O notation

Products and norms

$\langle x, y \rangle$	Vector inner product
$\lVert\cdot\rVert_1$	l^1-norm
$\lVert\cdot\rVert_2^2$	l^2-norm

Other functions

$W(x)$	Weighting function
$\psi(t)$	Change of variable
x_j, w_j	nodes, weights numerical integration

Measures

$E[X]$	Expectation
$Var[X]$	Variance
$corr[X]$	Correlation

Acronyms

ABS	Asset-Backed Security
CDF	Cumulative Distribution Function
CF	Characteristic Function
CLT	Central Limit Theorem
CVaR	Conditional Value-at-Risk
DE	Double Exponential transformation
EAD	Exposure At Default
EMG	Exponentially Modified Gaussian distribution
ES	Expected Shortfall
FFT	Fast Fourier Transform
FTM	Fourier Transform Method
GSL	GNU Scientific Library
LGD	Loss Given Default
LHHS	Latin Hypercube-Hammersley Sequence sampling
LHS	Latin Hypercube Sampling
LLN	Law of Large Numbers
MBS	Mortgage-Backed Security
MC	Monte Carlo
MGF	Moment Generating Function
MLHS	Median Latin Hypercube Sampling
MRA	Multi-Resolution Analysis
NAG	Numerical Algorithms Group
NIG	Normal Inverse Gaussian distribution
RLHS	Random Latin Hypercube Sampling
TDD	Test-Driven Development
VaR	Value-at-Risk
WA	Wavelet Approximation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Credit risk is a critical area in banking and is of concern to a variety of stakeholders: financial institutions, consumers and regulators. Credit risk is the risk of loss resulting from an obligor's inability to meet its legal obligation according to the debt contract. This circumstance is called the default event. The examples of default event include the bond default, the corporate bankruptcy, the credit card charge-off, the mortgages foreclosure or the unexpected change of credit rating, amongst others.

For financial institutions it is essential to quantify the credit risk at a portfolio level. Portfolio credit loss modelling requires the default dependence among obligors. A common approach is utilizing one or multiple factor models, such as the obligors are independent conditional on some latent common factor, which in some cases are assumed to follow a standard normal distribution. Some common factors are the state of the economy, changes of a market index or interest rates, for instance. Furthermore, it is usually considered that an obligor might incur in default in a fixed time. More realistic approaches take into account concentration risk in credit portfolios, which arise from an unequal distribution of loans to single obligors (name concentration) or different industry or regional sectors (sector or country concentrations). One of the main implications given by these concentrations is the effect called default contagion, which may lead to an increase in the portfolio credit risk due to the existing dependencies between different obligors. Financial institutions are also interested in the computation of common risk measures, such as Value-at-Risk (VaR) and Expected Shortfall (ES), which are typically used to determine the reserve capital to cover potential extreme losses. VaR is the measure chosen in the Basel II Accord (Basel Committee on Bank Supervision).

We can distinguish between two types of default probability, actual and implied. Actual default probability corresponds to the direct observations of the defaults. On the other hand, implied default probability refers to the risk-neutral default probability implied from the credit market data, e.g. corporate bond yields. This thesis focuses on actual defaults, the study of default probability bottom up from the actual credit performance.

Credit rating agencies such as Moody's, Standard & Poor's and Fitch focus their business in credit risk management and are particularly interested in the portfolio loss distribution. Credit ratings represent the creditworthiness of individual corporations and consumers. The final rating of Asset-Backed Securities (ABS), Residential Mortgage-Backed Securities (RMBS), Commercial Mortgage-Backed Securities (CMBS) or Collateralized Debt Obligations (CDO) are ultimately based on statistical models of the expected default probability, as well as judgement by rating specialists, i.e. credit risk analysts for the quantitative and qualitative derivation of the rating notes. The rating notes are classified by letters, Aaa and Baa are examples of

Moody's rating system, which uses Aaa, Aa, A, Baa, Ba, B, Caa, Ca, C to represent the likelihood of default from the lowest to the highest. The speculative-grade corporate bonds are sometimes said to be high-yield or junk.

Finally, although we focus on the most widely used methods, there exits other important methods for quantifying portfolio credit risk, such as the Binomial Expansion Technique or the Lognormal Method, which can be successfully applied for specific portfolios. Additionally, literature in credit risk has been fostered by academics in finance, statistics and computational mathematics and professionals in industry.

1.1 Contributions

Several methods and improvements are presented throughout this thesis, however we consider that the main contributions are those listed below:

- Chapter 2
  1. New approximation of the binomial distribution based on Necdet Batir's approximation for the factorial function.
  2. More precise computation of the exponentially modified Gaussian inverse cumulative distribution by using Newton-Raphson algorithm instead of optimization routines and faster methods for computing the normal inverse Gaussian distribution by means of numerical quadrature methods.
  3. New integral representation of the confluent hypergeometric function for argument z with large imaginary part.
- Chapter 4
  1. Numerical method for the generalized Vasicek model.
  2. Closed-form formulas for large portfolio approximation with special factor models.
  3. Fourier transform method with beta distributed loss given default computed using a new algorithm for the characteristic function of the beta distribution.

1.2 Numerical software

Most of contents of this thesis is oriented towards an efficient implementation of numerical methods and computational models in portfolio credit risk, as it becomes evident along its pages. Therefore, much of the effort has been devoted to coding and testing, as shown in Figure 1.1, where the total project is over 12000 lines of source code. Regarding integrated development environments, the code in C/C++ has been developed in Microsoft Visual Studio® 2012. Code in R has been developed using Rstudio and code in Python using Pycharm Community Edition.

Finally, we have used SVN as a version control system for two different project hosted in Assembla.com:

1. **CreditRating**: https://www.assembla.com/spaces/creditrating/subversion/source/HEAD/branches/CreditRatingTesting/CreditRatingTesting
2. **FTM_HaarWavelets**: https://www.assembla.com/spaces/ftm-haarwavelets/subversion/source/HEAD/branches/HaarWavelets

This code can only be used for research purposes. For other purposes or inquiries about it, please contact the author (guillermo.navas@estudiant.upc.edu).

1.2.1 Software development process

The main code in C++ has been coded following a software development process called Test-driven development (TDD). TDD is an advanced technique of using automated unit tests to drive the design of software and force decoupling of dependencies. This technique allows the creation of a comprehensive suite of unit tests that can be run to make sure that the required functionalities and specifications are met. Additionally, a set of tests with minimum dependency facilitates the isolation and detection of errors or performance issues. Therefore, this technique is heavily emphasized by developers using Agile development methodologies.

The main concept of TDD centers on two basic rules: Never write a single line of code unless you have a failing automate test and eliminate duplication. The development cycle is simple and can be described in 5 short steps:

1. **Add a new test**: Write code to include new feature trying to cover all the specifications. When writing new test is important to check expected fails.

2. **Run all tests**: Test the added feature and make sure that does not break or degrade other existing features.

3. **Write code if test failed**: If the test returns an unexpected fail write the minimum amount of code to pass the test without taking into account performance or coding style.

4. **Run all tests**: If all tests pass, we can be confident about the integration of the new feature in the main code. Otherwise, go to step 3.

5. **Code refactoring**: Refactoring is a technique for restructuring an existing body of code, altering its internal structure without changing its external behaviour. Therefore, rewrite the code written in step 3 in accordance with the project design patterns and avoiding duplication in order to clean up the code, since large projects without a clear structure might be problematic. We recommend Visual Assist or ReSharper-JetBrains as a refactoring tool and CodeMaid to cleanup code.

In order to create, run and customize a series of unit tests for each project, we have used the Microsoft unit test framework. However, other free unit test framework can be used in Visual Studio, we recommend NUnit.

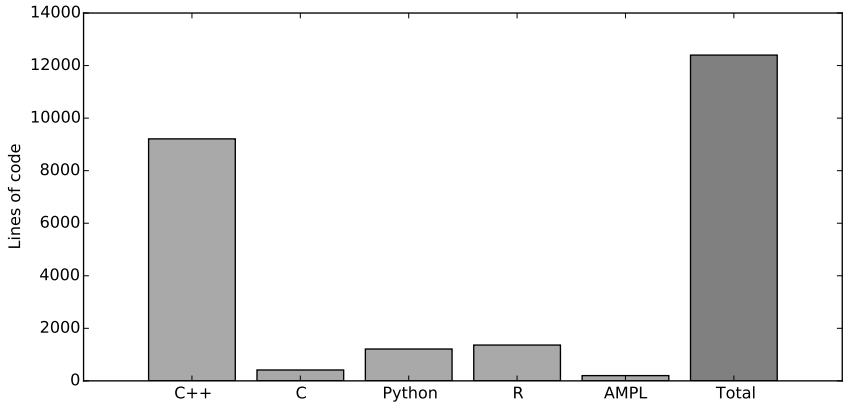


Figure 1.1: Lines of code for each programming language. Total = 12399 lines.

Chapter 2

Numerical Methods

This Chapter presents numerical methods employ to compute various probability distributions and special functions that are needed throughout this work. We review some common methods to compute the binomial distribution, and present some numerical improvements, being one of them a new approach based on Necdet's approximation to the factorial function. Subsequently, we focus on the computation of the Exponentially modified Gaussian distribution and the Normal Inverse Gaussian distribution, providing numerical methods that clearly outperform those used in some of the main R packages. Finally, we present the state-of-art on the computation of the confluent hypergeometric function and a novel method to compute the confluent hypergeometric function for values of large imaginary z applying methods used to compute highly oscillatory integrals.

2.1 Efficient computation of the binomial distribution

The binomial distribution with parameters k, n and p is the discrete probability distribution of obtaining exactly k successes out of n Bernoulli trials.

Probability mass function:

$$f(k;n,p) = P[X=k] = \binom{n}{k} p^k (1-p)^{n-k}, \;\; k = 0,1,2,...,n \tag{2.1.1}$$

Cumulative distribution function

$$F(k;n,p) = P[X \leq k] = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1-p)^{n-i} \tag{2.1.2}$$

where $\binom{n}{k}$ is the binomial coefficient. To compute it efficiently let us recall first

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \prod_{i=1}^{k} \frac{n-(k-i)}{i} = \prod_{i=1}^{k} \left(\frac{n-k}{i} + 1 \right) \tag{2.1.3}$$

We should use two identities which are easily obtained from the definition (Eq. 2.1.3).

$$\binom{n}{k} = \binom{n}{n-k} \quad \text{and} \quad \binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1} \tag{2.1.4}$$

by convention: $\binom{n}{0} = \binom{n}{n} = 1$.

Now, from identity (2.1.4) we can conveniently use either form of the binomial to do less than $n/2$ multiplications as follows: if $k \leq n-k$ then use $\binom{n}{k}$ and if $k > n-k$ then use $\binom{n}{n-k}$ (in the latter case we will apply $\prod_{i=1}^{n-k}\left(\frac{k}{i}+1\right)$). However, one must be aware that these products contain non-integer value fractions and for n big (e.g., $n >> 1000$), computing these continuous fractions will cause floating point error.

Thus, to avoid errors produced by continuous floating point division combine the previous strategy with the recursion suggested by identity (2.1.4), which can **keep all computations of integer type**, since $n * \binom{n-1}{k-1}/k = \frac{n}{k}\binom{n-1}{k-1}$ is an integer. And note that it is important for C++ to first multiply $\binom{n-1}{k-1}$ by n and then divide by k to get an integer without **explicit type conversion**. See implementation `BinomialCoeff`.

Listing 2.1: BinomialCoeff - computes the exact value using recursion.

```
template<typename value_type>
value_type BinomialCoeff(value_type n, value_type k)
{
  if (k > n)
    return 0; // undefined

  if (k == 0 || k == n)
    return 1;

  if (k > n - k)
    return BinomialCoeff(n, n - k);

  return n * BinomialCoeff(n - 1, k - 1) / k;
}
```

One can easily see that the time complexity of this algorithm is $\mathcal{O}(n)$, and if all arithmetic is performed in the integer domain we can avoid floating point calculations that can lead to precision errors (particularly with divisions). The above code allows to set the data type, so we can use `unsigned long long` to store an integer number till approximately $2^{64}-1$ or `double` if the required accuracy is not too strict. Furthermore, by using `double` we can reach values $\approx 1.79769 \cdot 10^{308}$.

Another approach for computing the binomial coefficient is by applying logarithmic properties.

$$\ln \binom{n}{k} = \ln n! - \ln(n-k)! - \ln k! \tag{2.1.5}$$

Writing the factorial as a gamma function allows the binomial coefficient to be generalized to noninteger arguments, if n is positive integer,

$$\ln \Gamma(n+1) = \ln n! \tag{2.1.6}$$

where $\ln \Gamma()$ is the logarithm of the gamma function.

$$Let\ \ G(n,k) = \ln \Gamma(n+1) - \ln \Gamma(n-k+1) - \ln \Gamma(k+1) = \ln \binom{n}{k} \tag{2.1.7}$$

Finally we can easily recover the original value,

$$\binom{n}{k} = e^{\ln \binom{n}{k}} = e^{G(n,k)} \tag{2.1.8}$$

The previous result does not eliminate the limitations in terms of arithmetic overflow, nevertheless this result can be applied for computing the binomial distribution as follows:

$$\ln f(k;n,p) = \ln \binom{n}{k} p^k (1-p)^{n-k} = G(n,k) + k \ln p + (n-k)\ln(1-p)$$

Hence,

$$f(k;n,p) = e^{(G(n,k)+k\ln p+(n-k)\ln(1-p))} \tag{2.1.9}$$

and analogously:

$$F(k;n,p) = \sum_{i=0}^{\lfloor k \rfloor} e^{(G(n,i)+i\ln p+(n-i)\ln(1-p))} \tag{2.1.10}$$

Equations (2.1.9) and (2.1.10) permit the computation of binomial coefficients with large n and k without arithmetic overflow, unlike the previous method that although setting the data type to `double` will be limited at $\binom{1028}{514} \approx 7.1561 \cdot 10^{307}$.

A not negligible point is the situation when $p = 0$ or a value very close to 0, for example $1e{-}17$. In general a `double` has a precision of 53 bits or about 16 decimal digits (this is reduced to 24 bits or about 7 decimal digits for `float`)[1], so values below this threshold will be considered as 0. Therefore, when $p = 0$ the function will not return $-\infty$ but must return 0 as the original function would do.

Listing 2.2: Binomialmf - Probability Mass function.

```
double Binomialmf(double p, int n, int k)
{
  // n: number of trials
  // k: number of success
  // p: probability of success
  // P[X = k]

  if (p != 0.0)
  {
    double q = 1 - p;
    double bmf = GammaLn(n+1);
    bmf -= GammaLn(n-k+1) + GammaLn(k+1);
    bmf += k*log(p) + (n-k)*log(q);

    return exp(bmf);
  }
  return 0.0;
}
```

The accuracy of this method depends on the precision achievable with the implementation of $\ln\Gamma()$, but apart from that, the method produces highly accurate results avoiding iterative procedures which are more computationally expensive. This method will be implemented to compute the default probability distribution so that is possible to compute large portfolios (1-5 million assets, for instance) in a very reasonable time.

Function	**n=100, k=50, p=0.5**	**n=10000, k=150, p=0.01**
Binomial`<int>`	$1.6364528993 \cdot 10^{-23}$	0
Binomial`<double>`	0.0795892373871788	$1.\#INF000000000000$
Binomialmf	0.0795892373871834	$5.78112710034635 \cdot 10^{-7}$
Mathematica(Benchmark)	0.0795892373871788	$5.78112710035172 \cdot 10^{-7}$

Table 2.1.1: Binomal mass function test. First two methods implement the `BinomialCoeff` function for computing the binomial mass function.

As shown in Table 2.1.1, as n increases the computation of the binomial probability mass function using the first method becomes infeasible for standard data types. However, with the first method we can obtain more accurate result for small values, as shown in Table 2.1.2.

[1] For higher precision use *Quadruple precision*, a binary of 128 bits with about 34 decimal digits of precision.

Function	**Elapsed inclusive Time %**
Binomial	6.1
Binomialmf	5.0

Table 2.1.2: Profiling functions to compute the binomial probability mass function. $n = 1000$, $k = 500$, $p = 0.5$. The first method uses the `BinomialCoeff` function to compute the binomial distribution.

2.1.1 Numerical approximations

Since $\ln\Gamma()$ is a difficult function to implement, it is worth studying possible approximations to $\ln n!$ and test if these can speed up the routine while maintaining a sufficient accuracy for large n. The first alternative is to use **Stirling's approximation**, an asymptotic approximation for factorials. Two Stirling's formulas are compared. Stirling's formula to one order and a second one including 4 terms:

$$n! \approx n^n e^{-n}\sqrt{2\pi n} \tag{2.1.11}$$

$$n! \approx n^n e^{-n}\sqrt{2\pi n}\Big(1 + \frac{1}{12n} + \frac{1}{288n^2} - \frac{139}{51840n^3}\Big) \tag{2.1.12}$$

We are more interested in the previous equations after applying logarithms,

$$\ln n! \approx n\ln n - n + \tfrac{1}{2}\ln(2\pi n) = \big(n + \tfrac{1}{2}\big)\ln n - n + \tfrac{1}{2}\ln(2\pi) \tag{2.1.13}$$

$$\ln n! \approx n\ln n - n + \tfrac{1}{2}\ln(2\pi n) + \frac{1}{12n} - \frac{1}{360n^3} + \frac{1}{1260n^5} \tag{2.1.14}$$

Now we can obtain from Equation (2.1.13) the following approximation for logarithm of the binomial coefficient,

$$\begin{aligned}\ln\binom{n}{k} &\approx \big(n + \tfrac{1}{2}\big)\ln n - \big(k + \tfrac{1}{2}\big)\ln k - (n - k + \tfrac{1}{2})\ln(n-k) - \tfrac{1}{2}\ln(2\pi)\\ &= \tfrac{1}{2}(\ln n - \ln k - \ln(n-k) - \ln(2\pi)) + n\ln n - k\ln k - (n-k)\ln(n-k)\end{aligned} \tag{2.1.15}$$

A similar result is achieved for the approximation with 4 terms using Equation (2.1.14), where we have reduced the resulting expression by applying Horner's rule for polynomial computation [38].

$$\begin{aligned}\ln\binom{n}{k} \approx{}& \tfrac{1}{2}(\ln n - \ln k - \ln(n-k) - \ln(2\pi)) + n\ln n - k\ln k - (n-k)\ln(n-k)\\ &+ \frac{1}{12}\Bigg[\frac{1}{n}\Bigg(1 + \frac{1}{n^2}\Bigg(-\frac{1}{30} + \frac{1}{105n^2}\Bigg)\Bigg) - \frac{1}{(n-k)}\Bigg(1 + \frac{1}{(n-k)^2}\Bigg(-\frac{1}{30} + \frac{1}{105(n-k)^2}\Bigg)\Bigg)\\ &- \frac{1}{k}\Bigg(1 + \frac{1}{k^2}\Bigg(-\frac{1}{30} + \frac{1}{105k^2}\Bigg)\Bigg)\Bigg]\end{aligned} \tag{2.1.16}$$

Recently, **Necdet Batir** [4] established a new Stirling-type approximation formula for the factorial function, which so far is one of the most accurate approximations to $n!$

$$n! \approx \sqrt{2\pi}n^n e^{-n}\sqrt[4]{n + \frac{1}{6} + \frac{1}{72n} - \frac{31}{6480n^2} - \frac{139}{155520n^3} + \frac{9871}{6531840n^4}} \tag{2.1.17}$$

We make use of (2.1.17) to write an expression for the logarithm of the binomial coefficient,

$$\ln n! \approx n\ln n - n + \tfrac{1}{2}\ln(2\pi) + \tfrac{1}{4}\ln\Big(n + \tfrac{1}{6} + \tfrac{1}{72n}\Big(1 + \tfrac{1}{90n}\Big(-31 + \tfrac{1}{24n}\Big(-139 + \tfrac{9871}{42n}\Big)\Big)\Big)\Big) \tag{2.1.18}$$

and having written $\ln n!$ we can easily write $\ln \binom{n}{k}$ as follows,

$$\begin{aligned}\ln \binom{n}{k} \approx\; & n \ln n - (n-k)\ln(n-k) - k \ln k - \tfrac{1}{2}\ln(2\pi) \\ & + \tfrac{1}{4}\Big[\ln\Big(n + \tfrac{1}{6} + \tfrac{1}{72n}\Big(1 + \tfrac{1}{90n}\Big(- 31 + \tfrac{1}{24n}\Big(- 139 + \tfrac{9871}{42n}\Big)\Big)\Big)\Big) \\ & - \ln\Big((n-k) + \tfrac{1}{6} + \tfrac{1}{72(n-k)}\Big(1 + \tfrac{1}{90(n-k)}\Big(- 31 + \tfrac{1}{24(n-k)}\Big(- 139 + \tfrac{9871}{42(n-k)}\Big)\Big)\Big)\Big) \\ & - \ln\Big(k + \tfrac{1}{6} + \tfrac{1}{72k}\Big(1 + \tfrac{1}{90k}\Big(- 31 + \tfrac{1}{24k}\Big(- 139 + \tfrac{9871}{42k}\Big)\Big)\Big)\Big)\Big] \end{aligned} \tag{2.1.19}$$

Table 2.1.3 shows that by reducing the computational time[2] using the simplest Stirling's approximation, the accuracy is substantially reduced. In order to reach a similar precision we need to use an approximation with several additional terms, worsening the computational time with respect to *StirlingLogBinomialCoeff* although not significantly due to a efficient implementation. Note that *NecdetBinomialCoeff* reduces the computational time comparing to *StirlingLogBinomialCoeff4* and obtain the same results as *GammaLnBinomialCoeff*. Therefore, the routine *NecdetBinomialCoeff* stands out to be the best one in terms of computational time and accuracy, at least for n and k small. *GammaLnBinomialCoeff* is more expensive to compute due to the function $\ln \Gamma()$ and as a result of not applying Horner's rule.

Function	**n=100, k=50**	**Rel.error**	**Time(10^{-6} s)**
log(BinomialCoeff`<double>`*)*	66.783841652017429	$4.47 \cdot 10^{-17}$	4.34
StirlingLogBinomialCoeff	66.786341610355805	$3.74 \cdot 10^{-5}$	0.62
StirlingLogBinomialCoeff4	66.783841652017472	$8.89 \cdot 10^{-16}$	0.64
GammaLnBinomialCoeff	66.783841652017486	$8.98 \cdot 10^{-16}$	0.72
NecdetBinomialCoeff	66.783841652017486	$8.89 \cdot 10^{-16}$	0.53
Mathematica (Benchmark)	66.783841652017426		

Table 2.1.3: Comparing logarithm of binomial coefficient approximations.

Function	**n=10000, k=150, p=0.01**	**Rel.error**
StirlingBinomial	$5.78112710040402 \cdot 10^{-7}$	$9.05 \cdot 10^{-12}$
Gamma-Binomialmf	$5.78112710034635 \cdot 10^{-7}$	$9.29 \cdot 10^{-13}$
NecdetBinomial	$5.78112710040402 \cdot 10^{-7}$	$9.05 \cdot 10^{-12}$
Mathematica (Benchmark)	$5.78112710035172 \cdot 10^{-7}$	

Table 2.1.4: Comparing accuracy of Binomial distribution approximations.

As shown in Table 2.1.4 as n and k increases the accuracy of the approximation methods gets worse, but not excessively. Table 2.1.3 shows a trade-off between accuracy and computation time, thus, depending on requirements one could choose either *Gamma-Binomialmf* or *NecdetBinomial*. An important point is the fact that Stirling's approximation improves as n increases, but for small n its accuracy is not sufficient even considering the approximation with 4 terms, whereas Necdet's approximation has a superior accuracy for a wider range of n and k. For comparing frequency and duration of function calls we have used the Microsoft Visual Studio® Profiler. Profiling aims to find performance bottlenecks in the code, and as we are dealing with high performance calculations, we think is worth to investigate about possible performance issues.

[2] Intel CPU i5-3317U, 1.70GHz processor and 4GB RAM, and using Microsoft® C/C++ Optimizing Compiler Version 17.00.61030 for x86.

2.2 Computing the Exponentially modified Gaussian Inverse cumulative distribution

In general the inverse cumulative distribution function or quantile function can be described for a continuous distribution as $F^{-1}(p)$ where $p \in [0,1]$ and x^* is the unique real number such that $F(x^*) = p$, so the problem can be formalized as

$$F^{-1}(p) = \inf\{x \in \mathbb{R} : F(x) \geq p\} \tag{2.2.1}$$

For some distributions there exist an analytical expression (the Exponential distributions or Weibull distribution, for instance) whose inverse can be obtained straightforward. However, for some other distributions like the Exponentially Modified Gaussian distribution (EMG), its cumulative distribution function is so complicated that its inverse needs to be computed numerically.

$$F(x;\mu,\sigma,\lambda) = \Phi(\lambda(y-\mu),0,\lambda\sigma) - e^{-\lambda(y-\mu)+\frac{(\lambda\sigma)^2}{2}+\ln(\Phi(\lambda(y-\mu),(\lambda\sigma)^2,\lambda\sigma))} = p \tag{2.2.2}$$

The R package `emg` computes the inverse cumulative distribution by solving the following unconstrained optimization problem using the L-BFGS-B algorithm [8], which is a correct general approach for all continuous distributions, where we want to minimize the absolute error.

$$\underset{x}{min} \quad \|F(x;\mu,\sigma,\lambda) - p\| = \underset{x}{min} \quad |F(x;\mu,\sigma,\lambda) - p| \tag{2.2.3}$$

The code below is the current implementation in the R package `emg`, which uses the function `optim`, usually applied for general-purpose optimization.

Listing 2.3: minimization function implemented in emg.

```
mu <- 0;sigma <- 1; lambda <- 1; p <- 0.5

opt1 <- optim(c(mu), fn=abs(p - pemg(y, mu, sigma, lambda)), lower=-Inf, upper
    =Inf, method="L-BFGS-B")

> opt1$message
[1] "ERROR: ABNORMAL_TERMINATION_IN_LNSRCH"
```

Note that the absolute value in the objective function produces a termination error as shown in the output provided by the `optim` function. In order to remove the absolute function we can change the problem formulation, since it is known that ℓ^1-norm unconstrained optimization problems can be reformulated as a constrained linear programming problems,

$$\begin{aligned} &min && z \\ &\text{subject to} && z \geq F(x;\mu,\sigma,\lambda) - p \\ & && z \geq p - F(x;\mu,\sigma,\lambda) \end{aligned} \tag{2.2.4}$$

Nevertheless, in this simple case we can use a square instead, obtaining a ℓ^2-norm squared optimization problem.

$$\underset{x}{min} \quad \|F(x;\mu,\sigma,\lambda) - p\|_2^2 = \underset{x}{min} \quad (F(x;\mu,\sigma,\lambda) - p)^2 \tag{2.2.5}$$

Equation (2.2.5) in contrast to Equation (2.2.3) does provide an analytical solution due to it is continuously differentiable nature, and it can be computed efficiently. Therefore, we compute its gradient and equate to 0, such that the optimal value is the polynomial root that minimizes the square error.

$$2(F(x;\mu,\sigma,\lambda) - p)\frac{\partial}{\partial x}F(x;\mu,\sigma,\lambda) = 2(F(x;\mu,\sigma,\lambda) - p)f(x;\mu,\sigma,\lambda) = 0 \tag{2.2.6}$$

Listing 2.4: minimization function using a square.

```
opt2 <- optim(c(mu), fn=(p - pemg(y, mu, sigma, lambda))^2, lower=-Inf, upper=
    Inf, method="L-BFGS-B")

> opt2$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Furthermore, as shown in Section 4.2, this type of one-dimensional equations can be solved by using the Brent's method or the Newton-Raphson algorithm. Therefore, we need to solve a root-finding problem with Equation (2.2.5). Table 2.2.1 and Figure 2.1 show the comparison in terms of computational time[3] and accuracy of these four methods. The benchmark is performed using the R package microbenchmark, which is a more accurate replacement of the often seen `system.time(replicate(1000, expr))` expression. The number of evaluations is set to 100.

Method	Iterations	$F^{-1}(p,\mu,\sigma,\lambda)$	$F(x^*)$	Rel.error
`optim` $\lvert F(x;\mu,\sigma,\lambda)-p\rvert$	58	0.8757984043	0.5000000188	$3.75\cdot 10^{-8}$
`optim` $(F(x;\mu,\sigma,\lambda)-p)^2$	6	0.8757984042	0.5000000187	$3.74\cdot 10^{-8}$
Newton-Raphson	5	0.8757983437	0.5	$< 10^{-22}$
Brent	8	0.8757983437	0.50	$1.94\cdot 10^{-11}$

Table 2.2.1: Comparison in terms of accuracy of the four methods for computing the EMG inverse cumulative distribution function. $\mu = 0$, $\sigma = \lambda = 1$ and $p = 0.5$. $x_0 = \mu$, tol=eps=$1e{-}08$.

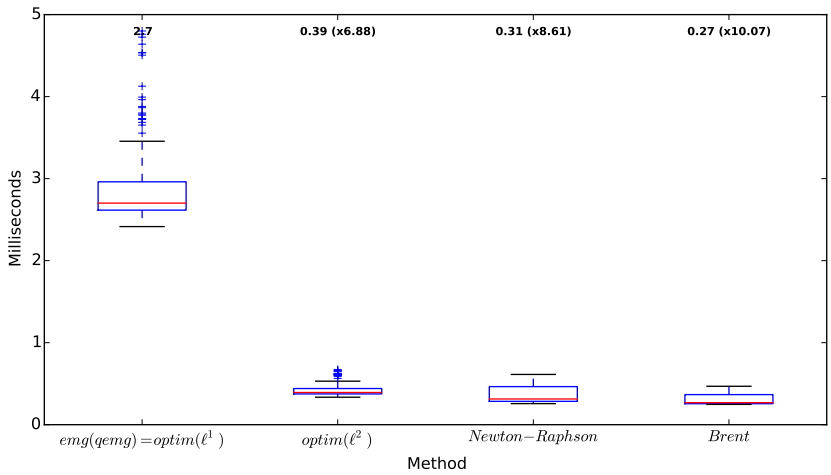


Figure 2.1: Benchmark of the four methods for computing the EMG inverse cumulative distribution function. Median computational time and speed up with respect to the `emg` routine on the upper X-axis.

Thus, we can conclude that the inverse cumulative distribution of the Exponentially modified Gaussian distribution should be computed using either the function `optim` to minimize the ℓ^2-norm squared optimization problem or applying Newton-Raphson, being the latter the best option. Solving by using Brent's method is a good alternative, since in practice is faster, however the result is not as accurate as the obtained by means of Newton-Raphson, where we take the first derivative or probability density function

$$\frac{\partial F(x;\mu,\sigma,\lambda)}{\partial x} = f(x;\mu,\sigma,\lambda) = \frac{\lambda}{2} e^{\frac{\lambda}{2}(2\mu+\lambda\sigma^2-2x)} \operatorname{erfc}\left(\frac{\mu+\lambda\sigma^2-x}{\sqrt{2}\sigma}\right) \tag{2.2.7}$$

[3]Intel CPU i5-3317U, 1.70GHz processor and 4GB RAM, and using R-3.1.2

2.3 Computing the Normal Inverse Gaussian (NIG) distribution.

The Normal-Inverse Gaussian distribution (NIG) was introduced in mathematical finance by Barndorff-Nielsen in [3]. The class of NIG distributions is a subclass of the generalized hyperbolic distributions obtained for $\lambda = -\frac{1}{2}$. The density of the NIG distribution is given by

$$f_{NIG}(x;\alpha,\beta,\mu,\delta) = \frac{\alpha\delta}{\pi} e^{(\delta\sqrt{\alpha^2-\beta^2}+\beta(x-\mu))} \frac{K_1(\alpha\sqrt{\delta^2+(x-\mu)^2})}{\sqrt{\delta^2+(x-\mu)^2}} \tag{2.3.1}$$

where K_1 is the modified Bessel function of the third kind of order 1 (see [1]), whose integral representation is as follows

$$K_1(z) = \frac{1}{2}\int_0^\infty e^{-\frac{1}{2}z(y+y^{-1})}\, dy \tag{2.3.2}$$

The NIG distribution has support $x \in \mathbb{R}$ and parameters $(\mu, \delta, \alpha, \beta)$ with domain of variation $0 \leq |\beta| < \alpha, \quad \mu \in \mathbb{R}, \quad \delta > 0$. The parameters μ and δ determine the location and scale, respectively, whereas α and β control the shape of the density. In particular, $\beta = 0$ generates a NIG symmetric distribution. The mean and variance of the NIG distribution are given by

$$E[X] = \mu + \frac{\delta\beta}{\sqrt{\alpha^2-\beta^2}} \quad \text{and} \quad Var[X] = \frac{\delta\alpha^2}{(\alpha^2-\beta^2)^{3/2}} \tag{2.3.3}$$

In addition, the normal-inverse distribution is a mixture of normal and inverse Gaussian distribution (IG)

$$f_{IG}(y;\alpha,\beta) = \begin{cases} \frac{\alpha}{\sqrt{2\pi\beta}} y^{-3/2} \exp\left(-\frac{(\alpha-\beta y)^2}{2\beta y}\right) & \text{if } y > 0 \\ 0 & \text{if } y \leq 0 \end{cases} \tag{2.3.4}$$

where $\alpha > 0$ and $\beta > 0$.

2.3.1 Cumulative distribution function

The calculation of the density function is straightforward, but the cumulative distribution function (CDF) has to be numerically integrated from (2.3.1)

$$F_{NIG}(x;\alpha,\beta,\mu,\delta) = \int_{-\infty}^{x} f_{NIG}(y;\alpha,\beta,\mu,\delta)\, dy \tag{2.3.5}$$

Alternatively, we can use the fact that the NIG distribution arises from a convolution of the normal and the inverse Gaussian distribution as remarked in [23]

$$F_{NIG}(x;\alpha,\beta,\mu,\delta) = \int_0^\infty \Phi\left(\frac{x-(\mu+\beta y)}{\sqrt{y}}\right) f_{IG}(y;\delta\gamma,\gamma^2)\, dy \tag{2.3.6}$$

where $\gamma = \sqrt{\alpha^2-\beta^2}$. The latter expression for the NIG CDF avoids the computation of the Bessel function, which results in a more efficient method.

We compute the NIG CDF applying two quadrature rules, Gauss-Legendre quadrature and Tanh-Sinh quadrature. These quadrature rules have the property that doubling the number of evaluations points roughly doubles the accuracy, so both are ideal for high precision quadrature. The advantages of the Tanh-Sinh algorithm are that it tends to handle endpoint singularities well, and that the nodes are cheap to compute on the first run. On the other hand, Gauss-Legendre quadrature often requires fewer function evaluations, and is therefore often faster for

repeated use, but the algorithm does not handle endpoints singularities as well and the nodes are more expensive to compute. Gauss-Legendre can be a better choice if the integrand is smooth. Finally, we compare our results to those computed by the R package `GeneralizedHyperbolic` in terms of accuracy and computational time.

Gauss-Legendre quadrature

The Gauss-Legendre quadrature is a Gaussian quadrature over the interval $[-1, 1]$ with weighting function $W(x) = 1$. The abscissas for quadrature order n are given by the roots of the Legendre polynomials $P_n(x)$. The general procedure is listed below,

1. We can transform any integral over interval $[a, b]$ to $[-1, 1]$ by

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}y + \frac{b+a}{2}\right) dy \tag{2.3.7}$$

2. An integral of $f(x)$ can be approximated with

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{2.3.8}$$

   Hence,

$$\int_a^b f(x)\, dx \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}x_i + \frac{b+a}{2}\right) \tag{2.3.9}$$

3. Legendre polynomials are defined by the following recursive rule, called *Bonnet's recursion formula*

$$P_0(x) = 1, \quad P_1(x) = x, \quad (n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \tag{2.3.10}$$

   Furthermore, there exist a recursive equation for their derivative

$$P_n'(x) = \frac{n}{x^2-1}(xP_n(x) - P_{n-1}(x)) \tag{2.3.11}$$

   In order to find the roots of those polynomials, numerical approximations such as the Newton-Raphson method is applied.

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2} \tag{2.3.12}$$

We implement in C++ and R the routine, due to G.B. Rybicki in [33]. This routine includes a range scaling from $[a, b]$ to $[-1, 1]$, that unfortunately cannot be applied directly in this case, since we are computing an improper integral. In general, we consider the following transformations in order to scale a given interval with infinite endpoints to $[-1, 1]$. Throughout this thesis, we use these transformations and other variants when solving integrals by quadrature rules. A simple pre-check routine was added to perform a transformation when needed.

- $[a, \infty)$:

$$y = \psi(t) = a + \frac{t+1}{t-1}, \quad \psi'(t) = -\frac{2}{(t+1)^2} \tag{2.3.13}$$

- $(-\infty, b]$:

$$y = \psi(t) = b + \frac{t-1}{t+1}, \quad \psi'(t) = \frac{2}{(t+1)^2} \tag{2.3.14}$$

- $(-\infty, \infty)$:

$$y = \psi(t) = \frac{1}{1-t^2}, \quad \psi'(t) = \frac{1+t^2}{(1-t^2)^2} \tag{2.3.15}$$

We apply the second change of variable. Note that $\psi(y)$ has the property that $\psi(y) \to -\infty$ as $y \to -1$ and $\psi(y) \to x$ as $y \to 1$, obtaining the following integral on the interval $(-1, 1)$.

$$F_{NIG}(x) = \int_{-1}^{1} f_{NIG}\left(x + \frac{t-1}{t+1}\right) \frac{2}{(t+1)^2}\, dt \tag{2.3.16}$$

For the second integral representation, the authors in [23] use a logarithmic transformation, $t = \exp(-y)$ such that integral (2.3.6) is transformed to

$$F_{NIG}(x) = \int_{0}^{1} \Phi\left(\frac{x - (\mu - \beta \log(t))}{\sqrt{\log(t)}}\right) f_{IG}(-\log(t); \delta\gamma, \gamma^2) \frac{1}{t}\, dt \tag{2.3.17}$$

On the other hand, we can apply the previous transformation for a interval $[a, \infty]$ to obtain suitable limits for Gauss-Legendre quadrature, such that

$$F_{NIG}(x) = \int_{-1}^{1} \Phi\left(\frac{x - (\mu - \beta\tilde{t})}{\sqrt{\tilde{t}}}\right) f_{IG}(\tilde{t}; \delta\gamma, \gamma^2) \frac{-2}{(t+1)^2}\, dt \tag{2.3.18}$$

where $\tilde{t} = \frac{t+1}{t-1}$

Tanh-Sinh quadrature

The tanh-sinh quadrature rule is based on the Euler-Maclaurin integral formula. By performing a change of variable involving nested exponentials/hyperbolic functions, the derivatives at the endpoints vanish rapidly. The basic tanh-sinh quadrature scheme is described as follows

$$I = \int_{-1}^{1} f(x)\, dx = \int_{-\infty}^{\infty} f(g(t))g'(t)\, dt \approx h \sum_{j=-N}^{N} w_j f(x_j) \tag{2.3.19}$$

This principle is utilized in the tanh-sin scheme by transforming the integral of $f(x)$ on the interval $[-1, 1]$ to an integral on $(-\infty, \infty)$ using the next change of variable

$$x = g(j) = \tanh(\frac{\pi}{2} \sinh hj), \quad w_j = g'(hj) = \frac{\cosh hj}{\cosh^2(\frac{\pi}{2} \sinh hj)} \tag{2.3.20}$$

We have implemented two different versions of the algorithm in R, a straightforward implementation and a more sophisticated version mainly based on [2, 6]. The latter implementation computes the integral for different increasing levels until the desired accuracy is reached or the final level has been completed. Given an initial level k, h is computed as $h = 2^{-k}$ until a maximum level $k = m$, which is typically set to $m = 12$.

If the complexity of the integrand can be estimated, we can fix a desired level of accuracy, fact that avoids the computation for multiple levels. Finally, an estimate of the error in tanh-sinh quadrature is given by

$$E(h) = h\left(\frac{h}{2\pi}\right)^2 \sum_{j=-N}^{N} F''(hj) \tag{2.3.21}$$

where $F(t) = f(g(t)g'(t))$ and $g(t) = \tanh(\pi/2 \sinh(t))$. A straightforward implementation uses the trapezoidal rule to evaluate the integrand on N points.

$$F_{NIG}(x) = \int_{-1}^{1} f_{NIG}\left(x + \frac{t-1}{t+1}\right) \frac{2}{(t+1)^2}\, dt \approx h \sum_{i=1}^{N} w_i f_{NIG}\left(x + \frac{y_i - 1}{y_i + 1}\right) \frac{2}{(y_i+1)^2}$$

where

$$y_i = \tanh(\frac{\pi}{2}\sinh(t_i)) \quad w_i = \frac{\frac{\pi}{2}\cosh(t_i)}{\cosh^2(\frac{\pi}{2}\sinh(t_i))}, \quad t_i = -t_a + (i-1)h \quad \text{and} \quad h = \frac{2t_a}{N-1}$$

The value of t_a is approximated by means of the expression $t_a = (p+1)^{0.46}$ obtained experimentally, which depends on the digits of precision p. We found that $p = 11$ is sufficient to evaluate the NIG distribution with 20-digit accuracy. See below a vectorized implementation in R.

Listing 2.5: Simple implementation in R of the tanh-sinh quadrature

```
quadts_trap <- function(f, p, n=100) {
  ta = (p+1)^0.46
  pi2 <- pi*0.5
  h = 2*ta/(n-1); t <- seq(-ta, ta, by=h)
  y <- tanh(pi2*sinh(t))
  w <- pi2*cosh(t)/(cosh(pi2*sinh(t))^2)*h
  s <- sum(f(y)*w); return (s)
}
```

Table 2.3.1 and Figure 2.2 compare the three methods introduced in this Section with the method implemented in the R package `GeneralizedHyperbolic`. The results indicate that tanh-sinh implementation using the trapezoidal rule produces the most accurate results, being less computationally expensive.

Method	**N**	$\boldsymbol{F(x^*)}$	**Rel.error**
$GeneralizedHyperbolic(pnig)$	30	0.6502646092594934756903	$< 10^{-22}$
$Gauss - Legendre$	30	0.6502646092914177167188	$4.91 \cdot 10^{-11}$
$Tanh - sinh$	$**$	0.6502646092594868143522	$1.02 \cdot 10^{-14}$
$Tanh - sinh^*(p = 11)$	200	0.6502646092594934756903	$< 10^{-22}$
$Benchmark$	100	0.6502646092594934756903	

Table 2.3.1: Comparison in terms of accuracy of the four methods for computing the NIG cumulative distribution function. $\mu = 0$, $\delta = 1$, $\alpha = 1$, $\beta = 0$ and $q = 0.3$. $Tanh - sinh^*$: Straightforward implementation. $**$: min_level=4, max_level=6. Benchmark: `pnig(q=0.3, intTol=1e-14, subdivisions = 100)`

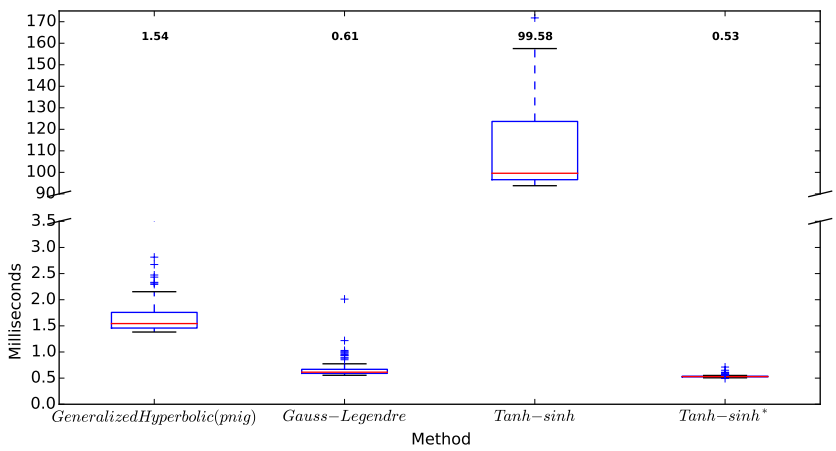


Figure 2.2: Benchmark of the four methods for computing the NIG cumulative distribution function. Median computational time on the upper X-axis.

2.3.2 Inverse cumulative distribution function

To compute the inverse cumulative distribution function of the NIG distribution we follow the same approach explained in Section 2.2, but we apply the Newton-Raphson method with numerical differentiation to diminish the computational effort introduced by a differentiation in summation form, see pseudocode below. Several methods can be applied, but we solely present two of the possible methods. The first one is a simple discretization of the integral by the midpoint rule, the second is a discretization based on the Gauss-Legendre quadrature, previously introduced.

$$\frac{4}{n}\sum_{i=1}^{n} f_{NIG}\left(q+\frac{x_i-1}{x_i+1}\right)\frac{2}{(x_i+1)^2}=p; \quad \sum_{i=1}^{n} w_i^{GL} f_{NIG}\left(q+\frac{x_i^{GL}-1}{x_i^{GL}+1}\right)\frac{2}{(x_i^{GL}+1)^2}=p$$

where $x_i = -1 + (i-0.5)\frac{2}{n}$

Algorithm 1 Newton-Raphson method with numerical differentiation

```
function NEWTON_METHOD(f, x_0, tol = 1e−12, maxiter = 50, h = 0.001)
    x = x_0
    i = 1
    while i ≤ maxiter do
        if |f(x)| < tol then
            STOP ("Optimal found")
        end if
        f'(x) = (f(x+h) − f(x−h)) / 2h                ▷ centered differentiation
        x = x − f(x) / f'(x) ; i = i + 1
    end while
    return x
end function
```

The obtained results highlight that the method implemented in `GeneralizedHyperbolic` based on adaptive quadrature from QUADPACK is significantly slower than both methods presented in this Subsection. We also notice that Brent's algorithm does not improve Newton-Raphson with numerical differentiaton.

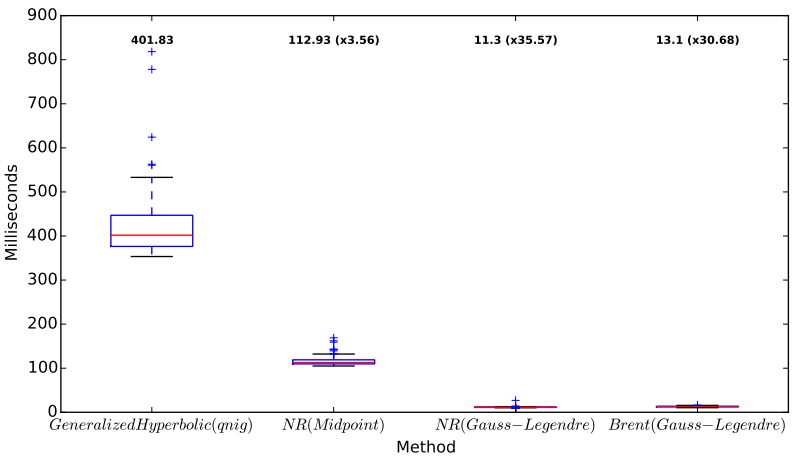


Figure 2.3: Benchmark of the four methods for computing the inverse NIG cumulative distribution function. Median computational time and speed up increase on the upper X-axis.

2.4 Computing the Confluent Hypergeometric function

The confluent hypergeometric function of the first kind ${}_1F_1(a;b;z)$ or Kummer's function $M(a,b,z)$ is a degenerate form of the hypergeometric function ${}_2F_1(a;b;c;z)$ which arises as a solution of the confluent hypergeometric differential equation

$$z\frac{d^2\,M(a,b,z)}{dz^2} + (b-z)\frac{d\,M(a,b,z)}{dz} - aM(a,b,z) \tag{2.4.1}$$

Special functions are also expressible as special cases of ${}_1F_1(a;b;z)$, including the incomplete gamma function, Bessel functions, Hermite polynomials and Laguerre polynomials. Furthermore, the computation of confluent hypergeometric functions is important in mathematical finance, for example to compute the characteristic function of the beta distribution, which is shown in Subsection 4.4.1.

In general, computing hypergeometric functions is difficult. Amongst the several methods for computing ${}_1F_1(a;b;z)$, Abramowitz and Stegun ([1], eq. 13.2.1) give a useful integral form for $\Re(b) > \Re(a) > 0$

$${}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\int_0^1 e^{zt}t^{a-1}(1-t)^{b-a-1}\,dt \tag{2.4.2}$$

A recent survey of numerical methods for computing the confluent hypergeometric function can be found in [29] and [30], the authors provide 'roadmaps' with recommendation for which methods should be used in each situation, see ([30], Table B.1). See below two of the methods with a superior average accuracy for all cases.

Taylor series: In practice the simplest method for computing the confluent hypergeometric function is to truncate the Taylor series defined as follows

$${}_1F_1(a;b;z) \approx S_N = \sum_{j=0}^{N}\frac{(a)_j}{(b)_j}\frac{1}{j!}z^j \tag{2.4.3}$$

and then use the relation ${}_1F_1(a;b;z) = \Gamma(b)\mathbf{M}(a;b;z)$ where

$$\mathbf{M}(a;b;z) = \sum_{j=0}^{\infty}\frac{(a)_j}{\Gamma(b+j)}\frac{z^j}{j!} \tag{2.4.4}$$

and the **Pochhammer symbol** $(\mu)_j$ is

$$(\mu)_0 = 1, \quad (\mu)_j = \mu(\mu+1)\times\cdots\times(\mu+j-1), \quad j = 1,2,\ldots$$

The following three-term recurrence relation can be used to obtain approximations of $\mathbf{M}(a;b;z)$ recursively in terms of previous approximations:

$$S_{-1} = S_0 = 1, \quad S_1 = 1 + \frac{a}{b}z$$

$$r_j = \frac{a+j-1}{j(b+j-1)}, \quad S_j = S_{j-1} + (S_{j-1} - S_{j-2})r_j z, \quad j = 2,3,\ldots$$

The authors consider a stopping criterion based on the difference of two successive terms ($|S_N - S_{N-1}| < \epsilon$), where $\epsilon \approx 2.2\times 10^{-16}$. As remarked in [30] as a or z becomes larger, the coefficients of the Taylor series become large, slowing down the convergence rate.

Gauss-Jacobi quadrature: Another popular method is the use of quadrature methods, when the relation $\Re(b) > \Re(a) > 0$ holds. Integral (2.4.2) can be approximated by means of a Gauss-Jacobi quadrature, which is appropriate to approximate integrals of the form

$$\int_{-1}^{1} f(x)(1-x)^{\alpha}(1+x)^{\beta}\,dx \tag{2.4.5}$$

Applying transformation $t = \psi(\tilde{t}) = \frac{1}{2}\tilde{t} + \frac{1}{2}$ and using Jacobi parameters $\tilde{\alpha} = b - a - 1$ and $\tilde{\beta} = a - 1$

$$ {}_1F_1(a;b;z) \approx \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \left[\frac{e^{z/2}}{2^{b-1}} \sum_{j=1}^{N_{mesh}} w_j^{GJ} e^{z x_j^{GJ}/2} \right] \tag{2.4.6}$$

where x_j^{GJ} and w_j^{GJ} are the Gauss-Jacobi abscissas and weights on $[-1, 1]$. As remarked in [30] and after our experiments, problems arise when $|z|$ is large, especially if $\Im(z) \gtrsim 200$ or when either a or b are fairly large. Therefore this method is appropriate for cases where $a, b \leq 50$ and $\Im(z) < 200$. Figure 2.4 shows ${}_1F_1(0 \leq a \leq 100, 0 \leq b \leq 100, x = 2.5)$, plotted as $\log({}_1F_1)$ to emphasize the behaviour.

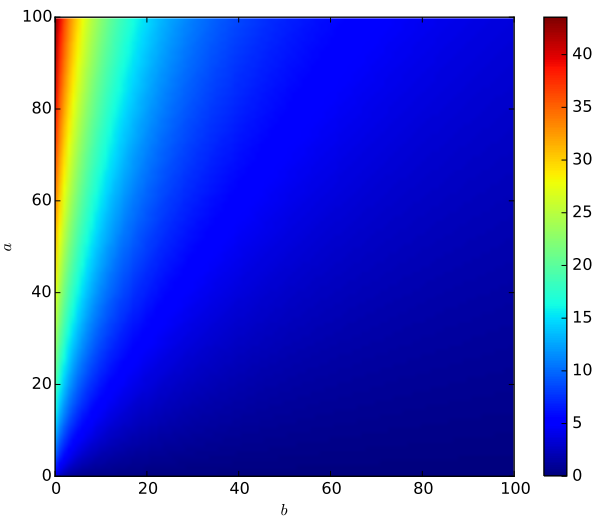


Figure 2.4: ${}_1F_1(a, b, x)$ for $a \in [0, 100]$, $b \in [0, 100]$ and $x = 2.5$.

Various numerical packages include the confluent hypergeometric function. Table 2.4.1 compares several numerical packages, commercial and open-source, and our implementation in C++. See ([30], Table 3.1) for recommendations for which methods to use for computing the confluent hypergeometric function when the parameters and the variable are real.

Package	**Value (z_1)**	**Rel.error**	**Value (z_2)**	**Rel.error**
NAG (`s22bac`)	0.013878573542982072	$1.25 \cdot 10^{-16}$	76.166022025278835	$1.87 \cdot 10^{-16}$
GSL	0.013878573542983349	$9.19 \cdot 10^{-14}$	76.166022025278849	$3.73 \cdot 10^{-16}$
Custom(GJ*)	0.013878573542988935	$4.94 \cdot 10^{-13}$	76.166022025314646	$4.70 \cdot 10^{-13}$
Custom(Taylor**)	0.013878573542982112	$2.75 \cdot 10^{-15}$	76.166022025278252	$7.46 \cdot 10^{-15}$
Mathematica	**0.013878573542982075**		**76.166022025278817**	

Table 2.4.1: Computation of the confluent hypergeometric function for parameters $a = 13.6$, $b = 14.2$ and $z_1 = -4.5$ and $z_2 = 4.5$. (*) 64 Gauss-Jacobi nodes, (**) $\epsilon = 1e - 12$.

2.4.1 Computing the Confluent Hypergeometric function in the complex plane: A new approach

The computation of the confluent hypergeometric function is much more difficult when considering complex arguments. In fact, neither the GNU Scientific Library (GSL) nor the Numerical Algorithms Group (NAG) Library have implementations for complex arguments. In this Subsection we focus on a particular case of interest, the characteristic function of the beta distribution, which can be expressed in terms of the confluent hypergeometric function

$$CF(t) = {}_1F_1(\alpha; \alpha + \beta; it) \tag{2.4.7}$$

where $\alpha, \beta > 0$ are the parameters of the beta distribution. In this particular case the relation $b > a > 0$ always holds, therefore the integral representation can be applied. The motivation behind this case is the computation of the portfolio Fourier transform with beta distributed loss given default, where $L_n = E_n \cdot \Lambda_n \cdot D_n$, details of this formula can be found in Section 4.4. The portfolio Fourier transform is given by

$$\hat{f}(t) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y) {}_1F_1(\alpha; \alpha + \beta; -itE_n)] f(y)\, dy \tag{2.4.8}$$

Given a Fourier resolution $M = 2^p,\ p \in [6, 12]$ and the step of distribution $\Delta x \lesssim 10^{-3}$, the Fourier vector required, which allows the use of the Fast Fourier Transform (FFT) algorithm for the computation of the Discrete Inverse Fourier Transform (DIFT), is computed as follows,

$$\hat{f}(t_j),\ \text{where } t_j = j\frac{T_{max}}{M},\ j = 0, \ldots, M \text{ and } T_{max} = \frac{2\pi}{\Delta x} \tag{2.4.9}$$

Hence, if T_{max} is large and $\Delta x \ll E_n \Lambda_n,\ \forall n \in \{1, \ldots, N\}$, then $(0 - it_j E_n \Lambda_n)$ is a value with a very large imaginary part as $j \to M$. A schema of methods is shown in ([30], Figure 3.2) with the main methods to compute the confluent hypergeometric function with complex arguments. For the computation of the characteristic function of beta distribution we can, in principle, use either Taylor series if $a, b \leq 50$, Gauss-Jacobi for small $|z|$ or asymptotic series, however asymptotic series cannot be apply when $\Re(z) = 0$, which discards the method from the list. On the other hand, we can apply recurrence relations (see [30], 3.6) that can reduce the problem of computation with large parameter values. We have tested the code developed by the authors, which can be found in http://datashare.is.ed.ac.uk/handle/10283/607, with unsatisfactory results when $|\Im(z)|$ is very large or a and b increase considerably. Furthermore, the computation time required by recurrence relations can vary significantly depending on the parameter values.

Table 2.4.2 shows several examples with different parameter values. Two methods have been considered, Taylor series with $\epsilon = 10^{-15}$ and Gauss-Jacobi quadrature with 200 nodes.

(a, b, z)	Mathematica	Taylor (10^{-15})	G-J(200)
$(2, 5, 2i)$	0.6447684839850957	3.58×10^{-17}	2.26×10^{-13}
	$+0.6597210696430052i$	$3.66 \times 10^{-17}i$	$2.31 \times 10^{-13}i$
$(5, 10, 200i)$	$-3.4058072327447802 \times 10^{-8}$	E	-1.25×10^{-8}
	$+1.9999373994098487 \times 10^{-8}i$	E	$-7.34 \times 10^{-9}i$
$(2, 5, -4800i)$	$-5.2075818310913531 \times 10^{-7}$	E	E
	$-6.3766435989570370 \times 10^{-10}i$	E	E
$(5, 10, 100 - 1000i)$	$3.9444325322222503 \times 10^{32}$	E	1.77
	$+3.1270029337251067 \times 10^{31}i$	E	$1.40 \times 10^{-1}i$

Table 2.4.2: Relative errors of Taylor series and Gauss-Jacobi quadrature when computing the confluent hypergeometric function on the complex plane. Relative error: $|sol^M - sol^*|/sol^M$, where sol^M is the solution provided by *Mathematica*. E: error due to overflow.

As shown in Table 2.4.2, the previous methods might fail when the parameter $|z|$ is large, therefore we need a different method to compute the confluent hypergeometric function for values of large imaginary part z. Next we present two methodologies developed throughout this thesis to compute the confluent hypergeometric function with large imaginary parts of the variable z.

Computation of the confluent hypergeometric function by adaptive quadrature for oscillatory integrals

The starting point for these methods is the transformation of the integral representation (2.4.2) into an oscillatory integral as follows

$$ {}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \int_0^1 e^{zt} t^{a-1}(1-t)^{b-a-1}\, dt $$

This is carried out by observing that $e^z = e^{\Re(z)} e^{i\Im(z)}$ and substituting accordingly

$$ {}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \int_0^1 \underbrace{e^{\Re(z)t} t^{a-1}(1-t)^{b-a-1}}_{r(t)} \underbrace{e^{i\Im(z)t}}_{e^{i\omega t},\omega=\Im(z)}\, dt \tag{2.4.10} $$

It is well known that Gauss-Jacobi quadrature in particular and more generally other classical quadratures based on polynomial interpolation, deteriorates as $\omega = \Im(z) \to \infty$, therefore other methods for highly oscillatory integrals are required. Now we can write the integrand $r(t)e^{i\omega t}$ in terms of its real and imaginary parts to obtain two separated integrals, (2.4.12) and (2.4.13)

$$ \int_0^1 r(t)e^{i\omega t}\, dt = \int_0^1 r(t)\cos(\omega t)\, dt + i \int_0^1 r(t)\sin(\omega t)\, dt \tag{2.4.11} $$

$$ \int_0^1 r(t)\cos(\omega t)\, dt = \int_0^1 e^{\Re(z)t} t^{a-1}(1-t)^{b-a-1}\cos(\Im(z)t)\, dt \tag{2.4.12} $$

$$ i\int_0^1 r(t)\sin(\omega t)\, dt = i\int_0^1 e^{\Re(z)t} t^{a-1}(1-t)^{b-a-1}\sin(\Im(z)t)\, dt \tag{2.4.13} $$

Thus, we obtain the following integral representation of ${}_1F_1(a;b;z)$

$$ \begin{aligned} {}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \Bigg[& \int_0^1 e^{\Re(z)t} t^{a-1}(1-t)^{b-a-1}\cos(\Im(z)t)\, dt \\ & + i\int_0^1 e^{\Re(z)t} t^{a-1}(1-t)^{b-a-1}\sin(\Im(z)t)\, dt \Bigg] \end{aligned} \tag{2.4.14} $$

Several scientific libraries have routines to calculate an approximation to the sine or cosine transform of a function f over $[a,b]$

$$ I = \int_a^b f(x)\cos(\omega x)\, dx \quad \text{or} \quad I = \int_a^b f(x)\sin(\omega x)\, dx \tag{2.4.15} $$

We use the routine `gsl_integration_qawo` from the GSL. This function implements an adaptive algorithm designed to integrate functions of the form (2.4.15). The results are extrapolated using the epsilon-algorithm [40] to accelerate the convergence of the integral. If a sub-interval has length $L = |b-a|2^l$ then the integration over this sub-intervals is performed by means of modified Clenshaw-Curtis procedure [31], if $L\omega > 4$ and $l \leq 20$ the sub-intervals are computed using a 25-points Clenshaw-Curtis integration rule, which handles the oscillatory behaviour. Sub-intervals where the above condition do not hold are computed using a 15-point Gauss-Kronrod integration. An alternative is the routine `d01snc` from the NAG Library, which is based upon the QUADPACK routine QFOUR [32].

Computation of the confluent hypergeometric function by steepest descent method

A more recent approach that entirely differs from most classical integration approaches, based on polynomial interpolation, is the so-called steepest descent method or numerical steepest descent method [21, 10]. This approach derives from the classical method of steepest descent; if $r(t)$ is an analytic function, the interval of integration is substituted by a union of contours on the complex plane, such that along these contours the integrand is non-oscillatory and exponentially decaying. Thus, $\int_a^b r(t)e^{i\omega t}\,dt$ can be written as

$$\begin{aligned}\int_a^b r(t)e^{i\omega t}\,dt &= e^{i\omega a}i\int_0^\infty r(a+ip)e^{-\omega p}\,dp - e^{i\omega b}i\int_0^\infty r(b+ip)e^{-\omega p}\,dp \\ &= \frac{e^{i\omega a}i}{\omega}\int_0^\infty r\Big(a+i\frac{q}{\omega}\Big)e^{-q}\,dq - \frac{e^{i\omega b}i}{\omega}\int_0^\infty r\Big(b+i\frac{q}{\omega}\Big)e^{-q}\,dq \end{aligned} \tag{2.4.16}$$

This method achieves high asymptotic order, using ν points for each integral, the error behaves as $\mathcal{O}(\omega^{-2\nu-1})$, which is twice the asymptotic decay for the same amount of function evaluations as Filon-type and Levin-type methods [10], whose absolute error behaves as $\mathcal{O}(\omega^{-n-1})$. Note that the asymptotic decay grows as the frequency parameter ω increases. Both integrals can be evaluated by Gauss-Laguerre integration [9], since both integrals (2.4.16) have the form

$$\int_0^\infty f(x)e^{-x}\,dx \approx \sum_{i=1}^n w_i f(x_i) \tag{2.4.17}$$

were the abscissas x_i are the zeros of the Laguerre Polynomial

$$L_n(x) = L_n^0(x) = \sum_{i=0}^n \frac{(-1)^i}{i!}\binom{n}{i}x^j; \quad x_i : L_n^0(x_i) = 0 \tag{2.4.18}$$

with recurrence relation $(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x)$ and weights w_i are defined as

$$w_i = \frac{\Gamma(n+1)x_i}{n![(n+1)L_{n+1}^0(x_i)]^2} \tag{2.4.19}$$

We employ the widely used Golub-Welsch algorithm [17] to compute Gauss-Laguerre abscissas and weights. This algorithm exploits the three-term recurrence relations satisfied by all real orthogonal polynomials. Alternatively, the Hale-Townsend algorithm [18] accurately computes Gauss-Legendre and Gauss-Jacobi quadrature nodes and weights by means of Newton's algorithm with initial guesses and function evaluations computed via asymptotic formulae. To our knowledge, the latter algorithm has not been modified in order to compute Gauss-Laguerre nodes although the paper suggests that the algorithm could be applied, since Laguerre polynomials also have well-known asymptotic expansions. The extension of this algorithm is beyond the scope of this thesis, but it is part of future research to improve the numerical steepest descent method.

Also mention that Equation (2.4.16) is simplified if $a = 0$ and $b = 1$

$$\int_0^1 r(t)e^{i\omega t}\,dt = \frac{i}{\omega}\int_0^\infty r\Big(i\frac{q}{\omega}\Big)e^{-q}\,dq - \frac{e^{i\omega}i}{\omega}\int_0^\infty r\Big(1+i\frac{q}{\omega}\Big)e^{-q}\,dq \tag{2.4.20}$$

Now we cast the integral (2.4.2) into (2.4.20) obtaining the following integral representation of ${}_1F_1(a;b;z)$

$$\begin{aligned}{}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\Bigg[&\frac{i}{\omega}\int_0^\infty e^{\Re(z)i\frac{q}{\omega}}\Big(i\frac{q}{\omega}\Big)^{a-1}\Big(1-i\frac{q}{\omega}\Big)^{b-a-1}e^{-q}\,dq \\ &- \frac{ie^{i\omega}}{\omega}\int_0^\infty e^{\Re(z)(1+i\frac{q}{\omega})}\Big(1+i\frac{q}{\omega}\Big)^{a-1}\Big(-i\frac{q}{\omega}\Big)^{b-a-1}e^{-q}\,dq\Bigg]\end{aligned} \tag{2.4.21}$$

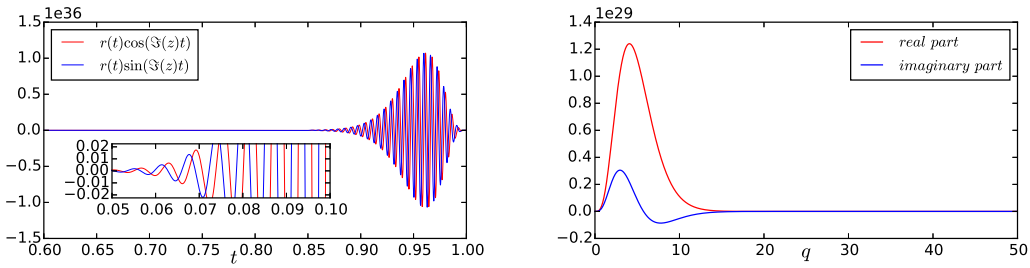
which is computed using N_{mesh} integration points

$$
{}_1F_1(a;b;z) \approx \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\Bigg[\frac{i}{\omega}\sum_{j=1}^{N_{mesh}} w_j e^{\Re(z)i\frac{x_j}{\omega}}\Big(i\frac{x_j}{\omega}\Big)^{a-1}\Big(1-i\frac{x_j}{\omega}\Big)^{b-a-1}
$$
$$
-\frac{ie^{i\omega}}{\omega}\sum_{j=1}^{N_{mesh}} w_j e^{\Re(z)(1+i\frac{x_j}{\omega})}\Big(1+i\frac{x_j}{\omega}\Big)^{a-1}\Big(-i\frac{x_j}{\omega}\Big)^{b-a-1}\Bigg] \tag{2.4.22}
$$

and where the error associated with a quadrature order n is $E = \frac{(n!)^2}{(2n)!}f^{(2n)}(\xi)$, for $\leq \xi \leq \infty$. If we consider the characteristic function of the beta distribution, Equation (2.4.22) is simplified as follows,

$$
{}_1F_1(a;b;z) \approx \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\Bigg[\frac{i}{\omega}\sum_{j=1}^{N_{mesh}} w_j \Big(i\frac{x_j}{\omega}\Big)^{a-1}\Big(1-i\frac{x_j}{\omega}\Big)^{b-a-1}
$$
$$
-\frac{ie^{i\omega}}{\omega}\sum_{j=1}^{N_{mesh}} w_j \Big(1+i\frac{x_j}{\omega}\Big)^{a-1}\Big(-i\frac{x_j}{\omega}\Big)^{b-a-1}\Bigg]
$$

Table 2.4.3 shows some examples and compares both presented methodologies. As shown, we encounter many problems applying the method based on adaptive quadrature when computing ${}_1F_1(a;b;z)$ with $\Re(z) > 0$ and in general the computation time required is highly superior compare to the time required by the steepest descent method to obtain a similar level of accuracy, thereby we focus on the second method although the following improvements can also be applied to the first method. Figure 2.5 illustrates the behaviour of an integrand evaluated for both methods. Observe that the reformulation of the integral by the steepest descent method removes the oscillatory nature on the real and imaginary part, a fact that facilitates the use of numerical integration techniques.



(a) *Integrand by adaptive quadrature.* (b) *Integrand by steepest descent method.*

Figure 2.5: ${}_1F_1(5, 10, 100 - 1000i)$.

Once the imaginary part of z is controlled, we need to put the focus on the values of a and b when $a, b \gtrsim 120$, since this order of magnitude leads to overflow problems. The most exposed expression to overflow is the ratio of Gamma functions, so we next study its asymptotic behaviour. Given

$$
\frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} = \frac{(b-1)!}{(a-1)!(b-a-1)!} \tag{2.4.23}
$$

we can use Stirling's approximation $k! \sim \sqrt{2\pi k}(\frac{k}{e})^k$ as $k \to \infty$

$$
\lim_{a,b\to\infty}\frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \sim \frac{\sqrt{b-1}(\frac{b-1}{e})^{b-1}}{\sqrt{2\pi(a-1)(b-a-1)}(\frac{a-1}{e})^{a-1}(\frac{b-a-1}{e})^{b-a-1}} \tag{2.4.24}
$$

Equation (2.4.24) shows that $\frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \to \infty$ as a and b increase and $b \gg a$. Furthermore, the integrand is also affected by a similar behaviour

$$\lim_{a,\omega\to\infty} \left(i\frac{x_j}{\omega}\right)^{a-1}\left(1-i\frac{x_j}{\omega}\right)^{b-a-1} = 0,\ b \gg a \tag{2.4.25}$$

$$\lim_{a,\omega\to\infty} \left(1+i\frac{x_j}{\omega}\right)^{a-1}\left(-i\frac{x_j}{\omega}\right)^{b-a-1} = 0,\ b \gg a \tag{2.4.26}$$

Bear in mind that there exist an important issue with computing the Gamma function of large arguments, since $\Gamma(172) \approx 1.241 \times 10^{309}$ and exceeds the double-precision 1.798×10^{308}. Hence, to avoid the above causes of overflow we can apply logarithm properties, obtaining the following modified integrands

$$\begin{aligned}\psi_1(a,b,z;q) = \log\Gamma(b) - \log\Gamma(a) - \log\Gamma(b-a) + \Re(z)i\frac{q}{\omega} \\ + (a-1)\log\left(i\frac{q}{\omega}\right) + (b-a-1)\log\left(1-i\frac{q}{\omega}\right)\end{aligned} \tag{2.4.27}$$

$$\begin{aligned}\psi_2(a,b,z;q) = \log\Gamma(b) - \log\Gamma(a) - \log\Gamma(b-a) + \Re(z)\left(1+i\frac{q}{\omega}\right) \\ + (a-1)\log\left(1+i\frac{q}{\omega}\right) + (b-a-1)\log\left(-i\frac{q}{\omega}\right)\end{aligned} \tag{2.4.28}$$

These integrands are used to rewrite the integral representation in such a way that overflow is well controlled. In practice, we should not encounter problems with most values of reasonable magnitude, as it will be shown later.

$${}_1F_1(a;b;z) = \frac{i}{\omega}\int_0^\infty e^{\psi_1(a,b,z;q)}\,dq - \frac{ie^{i\omega}}{\omega}\int_0^\infty e^{\psi_2(a,b,z;q)}\,dq \tag{2.4.29}$$

$${}_1F_1(a;b;z) \approx \frac{i}{\omega}\sum_{j=1}^{N_{mesh}} w_j e^{\psi_1(a,b,z;x_j)} - \frac{ie^{i\omega}}{\omega}\sum_{j=1}^{N_{mesh}} w_j e^{\psi_2(a,b,z;x_j)} \tag{2.4.30}$$

Furthermore, these integrands can be simplified when considering the characteristic function of beta distribution, since $\Re(z)i\frac{q}{\omega} = 0$ as shown above. For our experiments we have fixed the number of integration points to 64 Gauss-Laguerre weights and nodes. Therefore, the accuracy can be improved by increasing the number of points with the consequently increment in terms of computation time.

(a,b,z)	Mathematica	Adap.(10^{-10})	SD(64)	Time	Time
$(5,10,200i)$	$-3.4058072327447802 \times 10^{-8}$	-2.57×10^{-11}	-4.24×10^{-13}	20.7	0.17
	$+1.9999373994098487 \times 10^{-8}i$	$-1.51 \times 10^{-11}i$	$2.49 \times 10^{-13}i$		
$(5,10,100-1000i)$	$3.9444325322222503 \times 10^{32}$	E	4.91×10^{-13}	-	0.13
	$+3.1270029337251067 \times 10^{31}i$	E	$3.89 \times 10^{-14}i$		
$(2,20,50-2500i)$	$-5.4691850777561475 \times 10^{-5}$	-1.07×10^{-5}	-2.78×10^{-12}	18.6	0.15
	$+1.4441287455005657 \times 10^{-6}i$	$2.82 \times 10^{-7}i$	$7.35 \times 10^{-14}i$		
$(50,80,-4800i)$	$-5.3648791122771850 \times 10^{-57}$	E	-5.58×10^{-12}	-	0.16
	$-2.5405319387094543 \times 10^{-58}i$	E	$2.64 \times 10^{-13}i$		
$(90,100,-8000i)$	$2.3963908394271643 \times 10^{-21}$	E	2.81×10^{-13}	-	0.17
	$+5.2568330819871682 \times 10^{-20}i$	E	$-6.16 \times 10^{-12}i$		
$(2,20,-20000i)$	$-8.5499825580152617 \times 10^{-7}$	-2.62×10^{-12}	-2.78×10^{-12}	18.2	0.16
	$-1.4534982558011904 \times 10^{-9}i$	$4.45 \times 10^{-15}i$	$4.73 \times 10^{-15}i$		

Table 2.4.3: Comparison of oscillatory integration methods to compute the confluent hypergeometric function on the complex plane. Time in milliseconds.

(a, b, z)	**Mathematica**	**SD2(64)**	**Rel.error**
$(500, 510, 100 - 1000i)$	$-6.2228244518594344 \times 10^{39}$	$-6.2228244518589960 \times 10^{39}$	-8.10×10^{-14}
	$+6.5628468755115970 \times 10^{38}i$	$+6.5628468755089959 \times 10^{38}i$	$-8.54 \times 10^{-15}i$
$(700, 800, -10^5 i)$	$-1.2009316709301143 \times 10^{-113}$	$-1.2009316709306315 \times 10^{-113}$	-1.22×10^{-13}
	$+1.9151353691419210 \times 10^{-113}i$	$+1.9151353691419011 \times 10^{-113}i$	$-1.94 \times 10^{-13}i$
$(900, 1000, 100 - 10^5 i)$	$6.0430633105995064 \times 10^{-60}$	$6.0430633106045338 \times 10^{-60}$	3.73×10^{-13}
	$-6.8982723514693311 \times 10^{-60}i$	$-6.8982723514706421 \times 10^{-60}i$	$4.26 \times 10^{-13}i$
$(999, 1000, 700 - 10^5 i)$	$-4.6020552283174788 \times 10^{302}$	$-4.6020552283192628 \times 10^{302}$	-1.82×10^{-13}
	$-8.8660573516817291 \times 10^{302}i$	$-8.8660573516852506 \times 10^{302}i$	$3.51 \times 10^{-13}i$
$(999, 1000, -700 + 10^5 i)$	$-3.2868461235577501 \times 10^{-306}$	$-3.2868461235590394 \times 10^{-306}$	-1.32×10^{-13}
	$+9.2806995639762948 \times 10^{-306}i$	$+9.2806995639799548 \times 10^{-306}i$	$-3.72 \times 10^{-13}i$
$(900, 930, -10^{10} i)$	$-5.9703815795271339 \times 10^{-212}$	$-5.9703815795709398 \times 10^{-212}$	-1.12×10^{-11}
	$-3.3335392705314068 \times 10^{-212}i$	$-3.3335392704550403 \times 10^{-212}i$	$6.28 \times 10^{-12}i$
$(2, 20, 10^{10} i)$	$-3.4199999999999999 \times 10^{-18}$	$-3.4200000000000307 \times 10^{-18}$	-8.99×10^{-15}
	$+1.1627999999999999 \times 10^{-26}i$	$+1.1628000225087330 \times 10^{-26}i$	$3.05 \times 10^{-23}i$
$(4000, 4200, 50000i)$	$-7.2188661794436351 \times 10^{-219}$	$-7.2230501187607908 \times 10^{-219}$	-0.33
	$+2.9349000867097359 \times 10^{-218}i$	$+2.9323949592509309 \times 10^{-218}i$	$1.33i$
$(2, 5, -10^{20} i)$	$-1.1999999999999999 \times 10^{-39}$	$-1.199999999999999 \times 10^{-39}$	-1.76×10^{-16}
	$-6.6335289706601479 \times 10^{-59}i$	$-7.3462416784363379 \times 10^{-56}i$	B

Table 2.4.4: Results by applying the numerical steepest descent method to compute the confluent hypergeometric function on the complex plane for large values of a, b and $|z|$.

Finally, a few important aspects to keep in mind:

- if $r(t)$ grows exponentially large in the complex plane, the accuracy of the steepest descent method diminishes rapidly, as experienced with the last example. Furthermore, we have noticed a decrease of correct decimals as a and b increase and $|z|$ is not various orders of magnitude higher.

- the numerical steepest descent method experiences convergence problems when $|\Im(a)| > |\Re(a)|$ or $|\Im(b)| > |\Re(b)|$. We have not found any paper about the numerical steepest descent method where complex integrands were considered, but it seems reasonable to think that as $|\Im(a)|$ and $|\Im(b)|$ grow the total imaginary part of the integrand is split between the oscillatory term and the integrand itself causing numerical instability. This does not apply for the computation of the characteristic function, but restricts the regime of parameters and might not be reliable in some particular cases.

To summarize the key points: in this Subsection we presented a novel method to compute the confluent hypergeometric function for values of large imaginary z applying techniques used on the computation of highly oscillatory integrals. The computation by means of the numerical steepest descent method stands out as the fastest and more accurate technique and clearly superior to the Gauss-Jacobi quadrature or Taylor series for values of large imaginary z. Part of our ongoing research is based on the extension of this method to compute different integral representations of other hypergeometric functions.

Other methods

An alternative method to compute the confluent hypergeometric function based on a transformation to a oscillatory integral is presented in [29], where the transformation applied to integral (2.4.2) is given by

$$\int_{-1}^{1} e^{z\left(\frac{t}{2}+\frac{1}{2}\right)}(1-t)^{b-a-1}(1+t)^{a-1}\,dt = e^{z/2}\int_{-1}^{1} \underbrace{e^{\Re(z)t/2}(1-t)^{b-a-1}(1+t)^{a-1}}_{r(t)}\ \underbrace{e^{i\Im(z)t/2}}_{e^{i\omega t},\ \omega=\frac{\Im(z)}{2}}\,dt$$

This integral is computed by means of the method described in [28], which gives the following expression

$$\int_{-1}^{1} r(t)e^{i\omega t}\,dt \approx \sum_{k=1}^{N}\left[i^{k-1}(2k-1)\sqrt{\frac{\pi}{2\omega}}J_{k-\frac{1}{2}}(\omega)\times\sum_{i=1}^{N}w_{j-1}P_{k-1}(x_{j-1})r(x_{j-1})\right],\ \omega\neq 0$$

where J_ν is the Bessel function of first order, $P_k(x)$ are the Legendre polynomials, w_k are the weights for Gauss-Legendre quadrature and x_k are the nodes for Gauss-Legendre quadrature defined as the $k-$th root of $P_k(x)$. As remarked in [29], the number of points N_{mesh} needed to generate accurate results is very high, which increases the computation time significantly due to the expensive procedure of generating Legendre polynomials.

Chapter 3

Monte Carlo Methods

The purpose of this Chapter is to survey the Monte Carlo techniques and their variants. The first Section introduces the main principles of Monte Carlo. The second focuses on a variance reduction techniques called Latin Hypercubes. In general, Monte Carlo methods can be used in various problem classes, but we focus on numerical integration. An excellent reference for this Chapter is Glasserman [16].

3.1 Principles of Monte Carlo

Monte Carlo (MC) methods are based on the aggregation of results of n experiments obeying some property, where the experiments are suitable numbers randomly generated. In the simplest case, this means sampling randomly from a possible domain and observing the fraction of numbers that fall in a given set as an estimate. A common example typically used to illustrate this procedure is the approximation of π considering a circle inscribed in a unit square. More formally, the law of large numbers (LLN) ensures that this estimate converges to the correct value as the number of draws increases, whereas the central limit theorem (CLT) provides information about the likely magnitude of the error in the estimate after generating a finite number of draws.

As stated before, we are mainly interested in Monte Carlo Integration, usually applied in order to integrate a function over a complicated domain D. Monte Carlo integration throws points over a simplified domain D', such that $D' \supset D$, and the estimate is the fraction of points falling within D. Let us consider the simplest problem of estimating the integral of a function f over a finite interval.

$$I = \int_a^b f(x)\,dx \tag{3.1.1}$$

The previous integral can be represented as an expectation $E[f(U)]$, with U uniformly distributed between a and b. The Monte Carlo estimate is the average value of n function evaluations of n independent and uniformly distributed points over the interval $[a, b]$.

$$\hat{I}_n = \frac{b-a}{n} \sum_{i=1}^{n} f(U_i) \tag{3.1.2}$$

If f is integrable over $[a, b]$ then, by the strong law of large numbers, $\hat{I} \to I$ as $n \to \infty$. If f is square integrable and we set

$$\sigma_f^2 = \int_a^b (f(x) - I)^2\,dx \tag{3.1.3}$$

then the error $\hat{I}_n - I$ in the Monte Carlo estimate is approximately normally distributed with mean 0 and standard deviation or standard error $\sigma_f/\sqrt{n}$, the accuracy of this approximation improves as n increases. Furthermore, σ_f can be estimated using the sample standard deviation

$$s_f = \sqrt{\frac{b-a}{n-1}\sum_{i=1}^{n}(f(U_i) - \hat{I}_n)^2} \tag{3.1.4}$$

In numerical integration, Monte Carlo over one dimension if not competent, since its convergence rate is $\mathcal{O}(n^{-1/2})$, whereas the error in the simple trapezoidal rule is $\mathcal{O}(n^{-2})$, at least for smooth twice continuously differentiable integrands [1]. However, deterministic integration rules like trapezoidal quadrature suffer the curse of dimensionality when the tensor product approach for constructing multi-dimensional integration rules is applied. In particular, in Monte Carlo integration the $\mathcal{O}(n^{-1/2})$ convergence rate holds for all d dimensions, because the normalization $1/\sqrt{n}$ of the CLT does not depend on the dimension. In contrast, the error in a product trapezoidal rule (tensor product) is $\mathcal{O}(n^{-r/d})$, with $r = 2$ meaning twice differentiable. Other techniques such as sparse grids methods can avoid the curse of dimensionality for integrating multidimensional functions, and even outperform Monte Carlo for very specific smooth integrands, see [14].

3.2 Variance reduction techniques

Variance reduction techniques are improvements to increase the efficiency of Monte Carlo simulation by reducing the variance of simulation estimates. There exist two main strategies for reducing variance: exploiting specific features of a model to adjust simulation outputs, and reducing the variability in simulation inputs. Well known techniques are: Control variates, antithetic variates, stratified sampling, Latin hypercube sampling, moment matching methods, and importance sampling. However, due to the fact that we want methods that can be applied for a wide range of problems arising in portfolio credit risk, we focus on a generic method, Latin hypercube sampling, and some variants capable of increasing the gain in efficiency.

3.2.1 Latin Hypercube Sampling

Latin hypercube sampling (LHS) is one form of stratified sampling that can yield more precise estimates of the distribution function. In Latin hypercube sampling, the range of each uncertain parameter X_i is subdivided into non-overlapping intervals of equal probability. One value from each interval is selected at random with respect to the probability distribution in the interval. The n values thus obtained for X_i are paired in a random manner with n values of X_2. The main drawback of this stratification scheme is that it is uniform in one dimension and does not provide uniformity properties in d-dimensions.

Latin hypercube sampling is an extension of stratification for sampling in multiple dimensions. The difficulty is apparent even in the simple case of sampling from the d-dimensional hypercube $[0,1)^d$. Latin hypercube sampling treats all coordinates equally and avoids the exponential growth in sample size resulting from full stratification by stratifying only the one-dimensional marginals of a multidimensional joint distribution.

Table 3.2.1 shows a comparison of methods for sampling a standard normal distribution. As a measure of error we compute the sampling error for 100 samples and this is repeated 1000 times. The results show a sampling error of μ for MC that is 20 times larger than for Random

[1] $\mathcal{O}(n^{-2})$ is in most cases an upper bound, as shown in [22] for many functions, the error decreases much faster than that.

Latin Hypercube Sampling (RLHS) or simply Latin Hypercube Sampling (LHS) and this ratio is substantially higher if we compare against the Median Latin Hypercube Sampling (MLHS). Similar ratios are appreciable with regards to the sampling error of σ.

Theoretical results [16] for the univariate case show that the sampling error of MC decreases as $\mathcal{O}(1/\sqrt{N})$, whereas the sampling error for LHS is $\mathcal{O}(1/N)$, therefore it is quadratically faster. In general, computing an univariate integral using MC is not recommended due to the required number of samples to obtain a decent accuracy. However, if the number of samples if N instead of N^2, the computation time needed for sampling can be less than the time for computing nodes and weight for numerical integration methods, especially if the integrand is not well behaved.

Method	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$	**Sampling error ($\boldsymbol{\mu}$)**	**Sampling error ($\boldsymbol{\sigma}$)**
Pseudo-random	-0.001329	0.99850	0.09701	0.07022
Random LHS	-0.000103	1.00424	0.00477	0.01248
Median LHS	$-1.84 \cdot 10^{-17}$	0.99864	$1.95 \cdot 10^{-17}$	$2.17 \cdot 10^{-14}$
$\mathcal{N} \sim (0, 1)$	0	1	-	-

Table 3.2.1: Sampling error for each sampling method. Samples=100. Tests=1000.

Remark 3.1 *MC and LHS are both unbiased estimation techniques, therefore computed statistics approach their theoretical values as $N \to \infty$.*

From a developer perspective, we are interested in the efficient computation of both sampling techniques, otherwise the theoretical improvements will not be reflected. To do so, one should avoid loops and rather use vectorized functions like STL algorithms in C++ that allow `lambda` expressions. Especially in C++ it is recommended the usage of `std::vector` instead of native arrays. See below the pseudocode for both sampling techniques. The complete code in C++ can be found in the repository in Chapter 1.

Algorithm 2 Random Latin Hypercube Samplig

procedure RLHS(samples, dimensions, lbound, ubound)
  $a \leftarrow lbound$
  $b \leftarrow ubound$
  for $j := 1$ to dimensions **do**
    vector of indexes: $\pi_i, \quad i = 1, \ldots, samples$
    vector of independent random variables: $u_i \sim U[0, 1], \quad i = 1, \ldots, samples$
    random shuffle of π_i
    $v_{ij} = (b - a)\dfrac{\pi_i - 1 + u_i}{samples} + a$
    to extend to not uniform distribution, where F is the cumulative distribution:
    $z_{ij} = F^{-1}(v_{ij})$
  end for
  return z
end procedure

In order to sample other statistical distribution, one can apply the inverse transform sampling method from the generated random sample v. See below a few examples

- Case Normal distribution: $z \sim \mathcal{N}(\mu, \sigma)$: $z = \mu + \Phi^{-1}(v)\sigma$
- Case Exponential distribution: $z \sim Exp(\lambda)$: $z = -\frac{1}{\lambda}\ln(1 - v)$
- Case Weibull distribution: $z \sim Weibull(\alpha, \beta)$: $z = -\alpha(\ln(1 - v)^{1/\beta})$

Unlike in RLHS where one value from each interval is selected at random with respect to the probability density in the interval, in MLHS the value is chosen as the midpoint of the interval.

Algorithm 3 Median Latin Hypercube Samplig

procedure MLHS(samples, dimensions, lbound, ubound)
  $a \leftarrow lbound$
  $b \leftarrow ubound$
  for $j := 1$ to dimensions **do**
    vector of indexes: $\pi_i, \quad i = 1, \ldots, samples$
    random shuffle of π_i
    $v_{ij} = (b - a)\dfrac{\pi_i - 0.5}{samples} + a$
    to extend to not uniform distribution, where F is the cumulative distribution:
    $z_{ij} = F^{-1}(v_{ij})$
  end for
  return z
end procedure

Figure 3.1 shows the generated samples from RLHS and MLHS and compares each other to a pseudo-random standard normal distribution and the theoretical distribution. The differences with respect to pseudo-random sample are evident. Furthermore, if we compare both LHS methods, some small differences are detected, such as MLHS generates a symmetric sample, whereas this does not occur in RLHS.

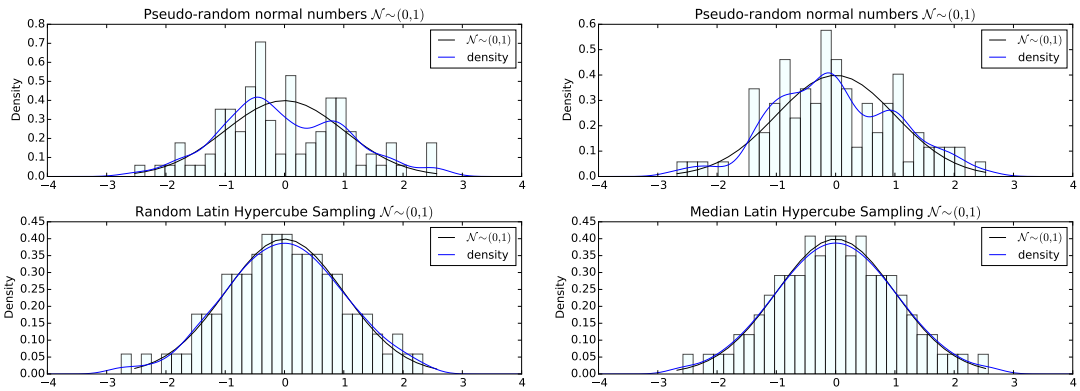


Figure 3.1: Random and Median Latin Hypercube sampling. Samples=100, variables=1.

3.2.2 Numerical examples

In this Subsection we compare LHS methods to crude MC by solving 5 integrals of different kind. Furthermore, we show several changes of variable applicable when solving improper integrals.

Integral 1 Let us first consider a very simple example, which serves us as a first comparison. As expected, the MLHS relative error decreases faster, especially for a small number of samples.

$$I = \int_0^1 x^2\,dx = \frac{x^3}{3}\Big|_0^1 = \frac{1}{3}$$

$$I \approx \frac{1}{N}\sum_{i=1}^{N}(x_i)^3 \quad \text{where } X \sim U(0,1) \tag{3.2.1}$$

N	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
10	0.3639118	0.3288905	0.3324999	9.17355	1.33286	0.25000
100	0.3519684	0.3331533	0.3333250	5.59052	0.05401	0.00250
1000	0.3361249	0.3333358	0.3333333	0.83747	0.00075	0.00003
10000	0.3324780	0.3333334	0.3333333	0.25659	0.00002	0.00000

Table 3.2.2: Definite integral 1. Tests=100. Relative error to exact value.

Integral 2 Let us consider now a more difficult integral where we can appreciate the power of Monte Carlo techniques.

$$I = \int_0^1 \cos(x)^2 e^{-x^2/2}\,dx = \frac{\sqrt{\frac{\pi}{2}}(2e^2\,\mathrm{erf}\left(\frac{1}{\sqrt{2}}\right) + \mathrm{erf}\left(\frac{1+2i}{\sqrt{2}}\right) - i\,\mathrm{erfi}\left(\frac{2+i}{\sqrt{2}}\right)}{4e^2} \approx 0.649353310769$$

where $\mathrm{erf}(x)$ is the error function and $\mathrm{erfi}(x)$ is the imaginary error function.

$$I \approx \frac{1}{N}\sum_{i=1}^{N}\cos(x_i)^2 e^{-x_i^2/2} \quad \text{where } X \sim U(0,1) \tag{3.2.2}$$

N	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
10	0.6000654	0.6447246	0.6496570	7.59031	0.71281	0.04677
100	0.6663541	0.6493676	0.6493563	2.61811	0.00223	0.00047
1000	0.6390092	0.6493560	0.6493533	1.59299	0.00041	0.00000
10000	0.6488693	0.6493533	0.6493533	0.07454	0.00000	0.00000

Table 3.2.3: Definite integral 2. Tests=100. Relative error to exact value.

Integral 3 Can be computed using Gauss-Hermite quadrature

$$I = \int_{-\infty}^{\infty} \cos(x)^2 e^{-x^2/2}\,dx = \left(1 + \frac{1}{e^2}\right)\sqrt{\frac{\pi}{2}} \approx 1.4229317610735444$$

This integral can be computed by MC after rewriting the integral and recognizing the standard normal probability density function,

$$\sqrt{2\pi}\int_{-\infty}^{\infty} \cos(x)^2 \frac{e^{-x^2/2}}{\sqrt{2\pi}}\,dx \Leftrightarrow E[\cos(x)^2] \quad \text{where } X \sim \mathcal{N}(0,1)$$

Hence,

$$I \approx \frac{\sqrt{2\pi}}{N}\sum_{i=1}^{N}\cos(x_i)^2 \tag{3.2.3}$$

N	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
10	1.4377003	1.5187798	1.3625425	1.03790	6.73595	4.24400
100	1.4196959	1.4227157	1.4216536	0.22741	0.01519	0.08983
1000	1.4227226	1.4227560	1.4232940	0.01470	0.01235	0.02546
10000	1.4228905	1.4229461	1.4229630	0.00290	0.00101	0.00220

Table 3.2.4: Improper integral 1. Tests=100. Relative error to exact value.

These results do not show a considerable advantage of LHS over crude MC. However, we can perform a change of variable in order to transform the previous integral over infinite intervals into a definite integral.

$$x = \psi(t) = \frac{t}{1-t^2}; \quad \text{Jacobian factor} = \psi'(t) = \frac{1+t^2}{(1-t^2)^2}\mathrm{d}t \tag{3.2.4}$$

$$I = \int_{-1}^{1} \cos\left(\frac{t}{1-t^2}\right)^2 e^{-\frac{\left(\frac{t}{1-t^2}\right)^2}{2}} \frac{1+t^2}{(1-t^2)^2}\, dt \tag{3.2.5}$$

Note that the Jacobian factor diverge as the limits are approached. If the function vanishes at least as fast as $1/x^2$ then the limit of the integrand is finite at the limits. However, when that is not the case, the function may not be absolutely convergent. An interesting point is that depending on the sampling methods this can be avoided, for instance, Latin hypercube do not generate the exact endpoints, but Monte Carlo sampling cannot guarantee the function evaluation at the limits. To overcome this drawback it is possible to implement a special handling for $|x| = 1$.

N	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
100	1.4565774	1.4254693	1.4229319	2.36453	0.17833	0.00001
1000	1.4208958	1.4229227	1.4229318	0.14308	0.00063	0.00000

Table 3.2.5: Transformed integral 3. Tests=100. Relative error to exact value.

Actually, the result obtained by MLHS with 1000 samples is 1.42293176107354, which implies an accuracy of magnitude 10^{-14}. For this sort of improper integrals another possible approach considers the use of other techniques like Gaussian quadrature. Applying Gauss-Hermite quadrature with 20 intervals and the proper change of variable we can obtain a similar level of precision.

$$\int_{-\infty}^{\infty} cos(x)^2 e^{-x^2/2}\, dx \approx \sum_{i=1}^{k} w_i^{GH} cos(\sqrt{2}y)^2\sqrt{2} \approx 1.4229317610735444$$

Integral 4 Finally, an integral considering a normal density function is tested

$$I = \int_{-\infty}^{\infty} \Phi\left(\frac{\alpha - \sqrt{\rho}\, x}{\sqrt{1-\rho}}\right) \frac{e^{-x^2/2}}{\sqrt{2\pi}}\, dx = 0.15$$

where $\alpha = \Phi^{-1}(0.15)$ and $\rho = 0.5$.

$$I \approx \frac{1}{N}\sum_{i=1}^{N} \Phi\left(\frac{\alpha - \sqrt{\rho}\, x_i}{\sqrt{1-\rho}}\right) \quad where\ X \sim \mathcal{N}(0,1) \tag{3.2.6}$$

N	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
10	0.1391898	0.1630717	0.1460529	7.20680	8.71447	2.63141
100	0.1478378	0.1499722	0.1498757	1.44148	0.01855	0.08289
1000	0.1493713	0.1500001	0.1499968	0.41915	0.00007	0.00211
10000	0.1499828	0.1500000	0.1499999	0.01146	0.00002	0.00005

Table 3.2.6: Improper integral 4. Tests=100.

Let us consider the previous change of variable to transform this integral,

$$I = \frac{1}{\sqrt{2\pi}} \int_{-1}^{1} \Phi\left(\frac{\alpha - \sqrt{\rho}\,\frac{t}{1-t^2}}{\sqrt{1-\rho}}\right) e^{-\frac{\left(\frac{t}{1-t^2}\right)^2}{2}} \frac{1+t^2}{(1-t^2)^2}\, dt \tag{3.2.7}$$

and a second possible approach using the transformation $\psi(t) = \tan(t)$ such that

$$\operatorname{atan}(\infty) = \tfrac{\pi}{2}, \quad \operatorname{atan}(-\infty) = -\tfrac{\pi}{2} \quad \text{and} \quad \psi'(t) = \frac{2}{\cos(2t)+1}$$

$$I = 2\int_{-\pi/2}^{\pi/2} \Phi\left(\frac{\alpha - \sqrt{\rho}\,\tan(t)}{\sqrt{1-\rho}}\right) e^{-\frac{\tan(t)^2}{2}} \frac{1}{\cos(2t)+1}\, dt \tag{3.2.8}$$

Method	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
approach 1	0.1325625	0.1503859	0.1499999	11.6249	0.25729	0.00000
approach 2	0.1829069	0.1503859	0.1499999	21.9379	0.53917	0.00000

Table 3.2.7: Transformations on integral 4. N=100 and tests=100.

Both transformation provide a similar accuracy. Using MLHS with 100 samples we can obtain an accuracy of order of magnitude 10^{-8} or 10^{-9} depending on the transformation. By contrast, using Gauss-Hermite quadrature with 100 intervals the achievable order of magnitude is 10^{-8}. Regarding computational time, both routines are similar, expending less than $1ms$ in average, since compute Hermite weights is also computationally extensive, hence the most important remark is the fact that Monte Carlo continue improving as the samples increase and not the Gaussian quadrature, which is typically limited to 100-150 intervals.

Integral 5 The purpose of the following integral is to demonstrate the inefficiency of the presented methods when solving multidimensional integrals.

$$\int_0^{\pi/2}\int_0^1 x\cos(xy)\,dy\,dx = \int_0^{\pi/2} \sin(xy)\Big|_{y=0}^{y=1} dx = \int_0^{\pi/2} \sin(xy)\,dx = -\cos(x)\Big|_{x=0}^{x=\pi/2} = 1$$

N	MCS	RLHS	MLHS	Err MCS %	Err RLHS %	Err MLHS %
200	1.0722116	1.0188677	1.0191900	7.22116	1.88677	1.91901
2000	0.9974967	0.9996955	0.9998356	0.25032	0.03045	0.01643
20000	1.0002159	1.0001869	1.0000815	0.02159	0.01869	0.00815

Table 3.2.8: Double integral 5. Tests=100.

No relevant differences between methods. In multiple integrals the good properties of LHS in one dimension start vanishing. Other methods such as Latin Hypercube-Hammersly Sequence Sampling (LHSS) has been considered, but without significant improvements, therefore it was not included. The results suggest that multidimensional integrals might require other methods like Quasi-Monte Carlo.

Chapter 4

Calculation of Default Probabilities

This Section introduces the basic theoretical model to calculate a distribution of default probabilities. This is the so-called firm's value model due to Vasicek (1987) and which has become to be the internal-rating-based approach of the Basel II. The Vasicek's model is the one-factor model built under several assumptions of homogeneity, which will later lead to the Large Homogeneous Portfolio [39]. We will also deal with the numerical calculations for a generalized version of Vasicek's model which considers, in general, any continuous distribution with support on the whole real line. A good reference for this Chapter is Schönbucher [34] ch.10.

4.1 One-factor model

The general approach in the factor model is that the default times in a portfolio are driven by several risk factors. The portfolio is composed by N debtors whose default is driven by the variation of its firm's value. The value of the assets of the nth debtor at time t is denoted by $V_n(t)$. Debtor n defaults if its firm's value falls below a default threshold $V_n(T) \leq \alpha_n$. The asset values of different debtors are assumed to be correlated with each other. The variance-covariance matrix of the V_i, $i \in N$ is denoted by Σ.

The covariance matrix Σ has $\frac{1}{2}N(N-1)$ (symmetric upper diagonal matrix) elements which characterize the joint default probability between the N debtors. In order to simplify and reduce computational complexity, M factors are chosen reducing the matrix dimensions to $N \cdot M$. In this first Section, a single factor is considered such that the number of coefficients is equal to N, which is reduced to just one parameter under the assumption of equal correlation coefficient. To clarify the previous concepts, a more detailed situation that will be used as a framework for the next theoretical aspects is presented.

Definition 4.1 *Consider a portfolio of N assets (structured finance products, for instance) where the nth asset has an initial value of V_n and its default probability p_n over a known time horizon T_i for which the portfolio's default distribution is determined. Some assumptions are considered in order to simplify the model:*

1. *Each asset in the portfolio corresponds to a different debtor. This is modelled if assets belonging to the same debtor are aggregated.*

2. *All assets have bullet amortisation(i.e. The payment of the principal balance is due at the end of the loan term)*

Note that in this framework we can still calibrate the model to reflect different individual default probabilities p_n over the time horizon by setting the barrier level to that level which replicates the given individual default probability. Assuming $V_n(t)$ is normally distributed, this level is

$$\alpha_n = \Phi^{-1}(p_n) \tag{4.1.1}$$

where $\Phi(\cdot)$ is the cumulative normal distribution function.

4.1.1 The distributions of defaults

The values of the assets of the obligors are driven by a common *systematic* factor Y and an *idiosyncratic* factor ϵ_n:

$$V_n(t) = \sqrt{\rho_n}\, Y + \sqrt{1-\rho_n}\, \epsilon_n \tag{4.1.2}$$

where Y and ϵ_n, $n \leq N$ are independent normally distributed random variables with mean 0 and variance 1 and $\rho \in [0,1]$. Using this approach the values of the assets of two obligors n and $m \neq n$ are correlated with linear correlation coefficient ρ. The important point is that *conditional on the realisation of the systematic factor Y, the firm's values and the defaults are independent.*

The systematic factor Y can be viewed as an indicator of the state of the business cycle, and the idiosyncratic factor ϵ_n as a firm-specific effects factor. The threshold α_n of the firm is mainly determined by the firm's reserves and balance-sheet structure. The relative sizes of the idiosyncratic and systematic components are controlled by the correlation coefficient ρ. If $\rho = 0$, then the business cycle has no influence on the fates of the firms, if $\rho = 1$, then it is the only driver of defaults, and the individual firm has no control whatsoever.

First, the business cycle variable Y materialises, and conditional on the general state of the economy, the individual defaults occur independently from each other, but with a default probability $p_n(y)$ which depends on the state of economy. This default probability is

$$p_n(y) = \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right) \tag{4.1.3}$$

Indeed, the individual conditional default probability $p(y)$ is the probability that the firm's value $V_n(t)$ is below the threshold α_n, given that the systematic factor Y takes the value y:

$$\begin{aligned} p_n(y) &= P[V_n(t) < \alpha_n \,|\, Y = y] \\ &= P[\sqrt{\rho_n}\, y + \sqrt{1-\rho_n}\, \epsilon_n < \alpha_n] \\ &= P\left[\epsilon_n < \frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right] \\ &= \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right) \end{aligned} \tag{4.1.4}$$

It can be proved that the expected value of the random variable $p_n(y)$ is the default probability p_n. We use two well known facts in Statistics:

1. The probability of the event[1] $[X \in A]$ where X is a random variable with distribution F is the expectation of its indicator function:

$$P[[X \in A]] = \int_A dF = \int 1_A(x) dF(x) = E[1_A(x)]$$

[1] Iverson bracket $[X \in A]$: This notation denotes that is 1 if the condition in square brackets is satisfied and 0 otherwise.

2. Law of total expectation: The expectation of a conditional expectation is the expectation (see [12, p. 162]):
$$E[E[X|Y]] = E[X]$$

Proposition 4.2 (Expected value of a conditional probability) *Given the expected value of a conditional probability $P[X < \alpha \,|\, Y = y]$, the unconditional probability is $P[X < \alpha]$:*

$$E[P[X < \alpha \,|\, Y = y]] = P[X < \alpha]$$

Proof: *With the two facts previously stated, we obtain:*

$$E[P[X < \alpha \,|\, Y = y]] = E[E[1_{(X<\alpha|Y=y)}]] = E[1_{(X<\alpha)}] = P[X < \alpha]$$

□

In particular from this theorem it follows immediately

$$\begin{aligned} E[p_n(y)] &= E[P[V_n(t) < \alpha_n \,|\, Y = y]] \\ &= P[V_n(t) < \alpha_n] = p_n \end{aligned}$$

This result remarks the fact that the expected value of the default probability conditional to a realisation of Y does not depend on the factor distribution; therefore, this statement holds for any factor which is not normally distributed. In case that the assumption about the factor distribution was non normal (NIG distribution, for instance), one must carefully determine the default threshold α_n since $\alpha_n = \Phi^{-1}(p_n)$ does not apply. The default threshold is obtained by solving for α_n the next equation, where $\phi(\cdot)$ denotes the density function for the factor:

$$E[p_n(y)] = \int_{-\infty}^{\infty} \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\phi(y)\, dy = p_n \tag{4.1.5}$$

The next step is to obtain the distribution function of the defaults. The probability of having exactly n defaults is the average of the conditional probabilities of n defaults, averaged over the possible realisations of Y and weighted with the probability density function $\phi(y)$,

$$P[X = n] = \int_{-\infty}^{\infty} P[X = n \,|\, Y = y]\phi(y)\, dy \tag{4.1.6}$$

Conditional on the realisation $Y = y$ of the systematic factor, the probability of having n defaults is given by the binomial probability mass function, which is a discrete probability function since defaults are discrete random variables.

$$P[X = n \,|\, Y = y] = \binom{N}{n}(p(y))^n(1-p(y))^{N-n} \tag{4.1.7}$$

where we use the conditional independence of the defaults in the portfolio. Substituting this and (4.1.4) into Equation (4.1.6) we obtain that the probability of having exactly n defaults in the underlying portfolio is given by

$$P[X = n] = \int_{-\infty}^{\infty} \binom{N}{n}\left(\Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\right)^n \left(1 - \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\right)^{N-n} \phi(y)\, dy \tag{4.1.8}$$

Therefore, the default distribution function, under the normality assumption is:

$$P[X \leq m] = \sum_{n=0}^{m} \binom{N}{n} \int_{-\infty}^{\infty} \left(\Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\right)^n \left(1 - \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\right)^{N-n} \phi(y)\, dy \tag{4.1.9}$$

4.1.2 Computing the default probability function

The presented method to solve the previous integrals is the Gauss-Hermite Quadrature. The Gauss-Hermite quadrature is a form of Gaussian quadrature for approximating the value of integrals of the next form:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)\, dx \approx \sum_{i=1}^{k} w_i f(x_i) \tag{4.1.10}$$

Where k is the number of sample points used. The x_i are the abscissas of the Hermite function ($W(x) = e^{-x^2}$), and the associated weights w_i are given by

$$w_i^{GH} = \frac{2^{k-1} k!\, \sqrt{\pi}}{k^2 [H_{k-1}(x_i)]^2},\ i = 1, 2, \ldots, k \tag{4.1.11}$$

Which to avoid computational overflows due to large N, it is turned into the following expression by using the orthonormal set of polynomials $\widetilde{H}_j$:

$$\widetilde{H}_{-1} = 0, \quad \widetilde{H}_0 = \frac{1}{\pi^{1/4}}, \quad \widetilde{H}_{i+1} = x\sqrt{\frac{2}{i+1}}\widetilde{H}_i - \sqrt{\frac{i}{i+1}}\widetilde{H}_{i-1} \quad \widetilde{H}_i' = \sqrt{2i}\, \widetilde{H}_{i-1} \tag{4.1.12}$$

$$w_i^{GH} = \frac{2}{[\widetilde{H}_N'(x_i)]^2} \tag{4.1.13}$$

In order to compute the distribution function of the defaults with this method, some amendments are required:

$$P[X \leq m] = \sum_{n=0}^{m} \binom{N}{n} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} h(y)\, dy$$

Let us take $h(y)$ as

$$h(y) = \left(\Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\right)^n \left(1 - \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right)\right)^{N-n}$$

We need the following change of variable

$$x^2 = \frac{y^2}{2} \Leftrightarrow y = x\sqrt{2}$$

applying integration by substitution

$$P[X \leq m] = \sum_{n=0}^{m} \binom{N}{n} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2} h(x\sqrt{2})\sqrt{2} dx = \sum_{n=0}^{m} \binom{N}{n} \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-x^2} h(x\sqrt{2}) dx$$

and finally with (4.1.10) and (4.1.13)

$$P[X \leq m] \approx \sum_{n=0}^{m} \binom{N}{n} \frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} w_i^{GH} \left(\Phi\left(\frac{\alpha_n - \sqrt{2\rho_n}\, x_i}{\sqrt{1-\rho_n}}\right)\right)^n \left(1 - \Phi\left(\frac{\alpha_n - \sqrt{2\rho_n}\, x_i}{\sqrt{1-\rho_n}}\right)\right)^{N-n} \tag{4.1.14}$$

Similarly, we can transform the probability distribution of the defaults, leading to the following expression

$$P[X = n] \approx \frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} w_i^{GH} \binom{N}{n} \left(\Phi\left(\frac{\alpha_n - \sqrt{2\rho_n}\, x_i}{\sqrt{1-\rho_n}}\right)\right)^n \left(1 - \Phi\left(\frac{\alpha_n - \sqrt{2\rho_n}\, x_i}{\sqrt{1-\rho_n}}\right)\right)^{N-n} \tag{4.1.15}$$

As shown in Section 2.1, some computational problems arise when evaluating large binomial coefficients. The obtained results in (2.1.9) are used for solving (4.1.14) and (4.1.15), getting the final expression for computing the default probability distribution,

$$P[X = n] \approx \frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} w_i^{GH} e^{(G(N,n)+n \ln t+(N-n)\ln(1-t))} \tag{4.1.16}$$

where $t = \Phi\big(\frac{\alpha_n - \sqrt{2\rho_n}\, x_i}{\sqrt{1-\rho_n}}\big)$. See Section 2.1 for further details about implementation and computational issues.

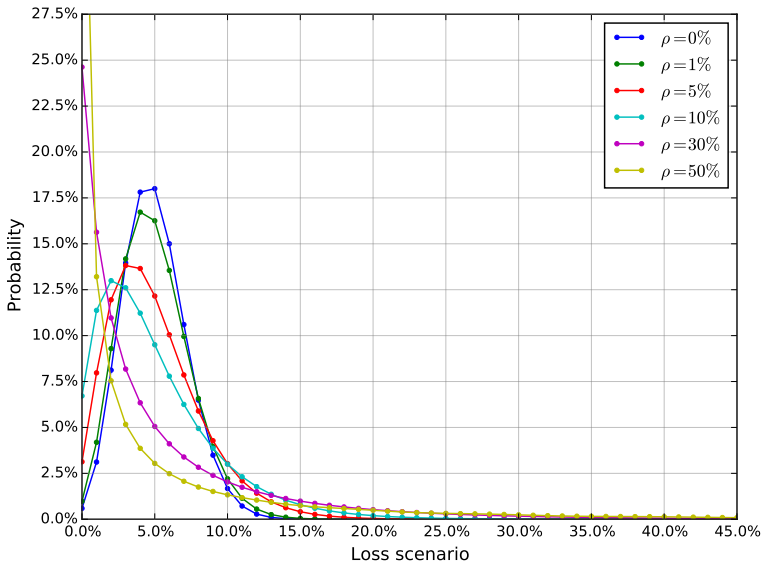


Figure 4.1: Default probabilities distributions under different asset correlation. Parameters: $N = 100$ and $p = 5\%$. Asset correlation ρ in percentage points.

ρ	Mean	Stdev	Stdev/Mean	$VaR_{0.999}$
0	0.0500000000	0.0217944947	0.435889	0.134638
0.01	0.0500000000	0.0241191949	0.482384	0.147820
0.05	0.0500000000	0.0322146401	0.644293	0.202831
0.10	0.0500000000	0.0409348413	0.818697	0.271646
0.30	0.0499999935	0.0711567420	1.423135	0.556961
0.50	0.0499974079	0.1003254738	2.006614	0.814539

Table 4.1.1: Default probabilities distributions computed using a Gauss-Hermite quadrature with 20 nodes.

From Table 4.1.1 we can notice that the number of correct figures diminishes as we consider more extreme cases, $\rho = 0.3$ or $\rho = 0.5$. We have included a metric called **standard deviation-over-mean ratio**, which is used to calibrate the factor loadings ($w = \sqrt{\rho}$) for ABS and MBS transactions and whose value rarely exceeds 1.5.

4.1.3 Approximations

For the moment we have presented a robust method to compute the distribution function of the defaults based on the efficient computation of the binomial distribution. In principle, we could approximate the binomial distribution function by using a Normal approximation or a Poisson approximation, which is useful for calculations with default or loss rates (actuarial approach).

Theorem 4.3 (Poisson limit theorem) *If $n \to \infty$ and $p \to 0$, such that the expectation of binomial distribution $np = \lambda$ then the Binomial distribution converges to the Poisson distribution.*

$$\binom{n}{k} p^k q^{n-k} \approx \frac{e^{-\lambda}\lambda^k}{k!} \tag{4.1.17}$$

Thus the distribution function of the defaults can be rewritten in terms on the Poisson approximation, given the following expression

$$P[X = n] \approx \int_{-\infty}^{\infty} exp\Big(- n\Phi\Big(\frac{\alpha_n - \sqrt{\rho}\, y}{\sqrt{1-\rho}}\Big)\Big)\Big(n\Phi\Big(\frac{\alpha_n - \sqrt{\rho}\, y}{\sqrt{1-\rho}}\Big)\Big)^k k!^{-1} \frac{e^{\frac{-y^2}{2}}}{\sqrt{2\pi}}\, dy \tag{4.1.18}$$

We use again the Gauss-Hermite quadrature for computing the distribution of the defaults,

$$P[X = n] \approx \frac{1}{\sqrt{\pi}} \sum_{i=1}^{k} w_i^{GH} exp\Big(- n\Phi\Big(\frac{\alpha_n - \sqrt{2\rho}\, x_i}{\sqrt{1-\rho}}\Big)\Big)\Big(n\Phi\Big(\frac{\alpha_n - \sqrt{2\rho}\, x_i}{\sqrt{1-\rho}}\Big)\Big)^k k!^{-1} \tag{4.1.19}$$

Furthermore, in order to reduce computational time and numerical overflow produced when $n > 171$, we compute the Poisson distribution as follows,

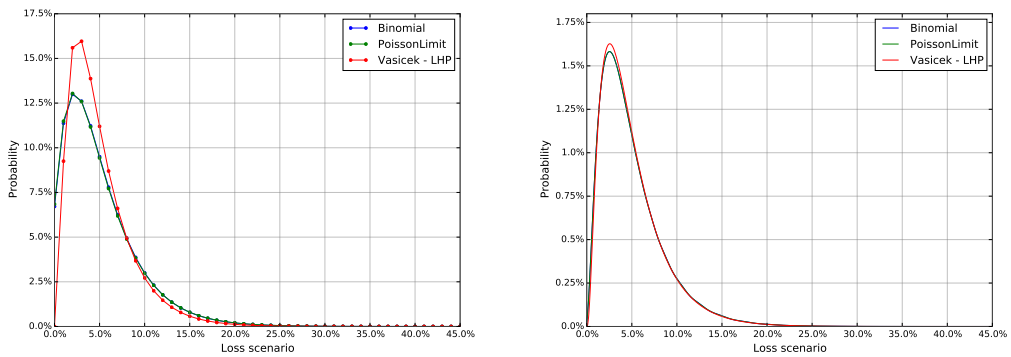
$$\frac{e^{-\lambda}\lambda^k}{k!} = e^{(-\lambda + k \ln k - \ln \Gamma(k+1))} \tag{4.1.20}$$

For completeness we include the Normal approximation of the binomial distribution, although the Poisson approximation behaves significantly better for $p \leq 0.1$.

Theorem 4.4 (de Moivre-Laplace theorem) *As n grows larger, for k in the neighbourhood of np we can approximate*

$$\binom{n}{k} p^k q^{n-k} \approx \frac{1}{\sqrt{2\pi npq}} e^{-\frac{(k-np)^2}{2npq}}, \qquad q = 1 - p,\ p, q > 0 \tag{4.1.21}$$

Figure 4.2 compares the portfolio computed with the Poisson approximation with respect to the portfolio using the binomial distribution and the portfolio computed by means of the Vasicek distribution explained in Section 4.3.



(a) *Assets=100, ρ=5, p=10.*

(b) *Assets=1000, ρ=5, p=10.*

Figure 4.2: Poisson approximation and Vasicek distribution approximation.

4.2 Generalized Vasicek model

In this Section we develop various numerical methods for computing the loss distribution with a general distribution function. Schönbucher in [35] proposed the generalized one-factor model, but does not provide a solution method.

Assumption 4.1 (Generalized One-Factor Model) *The values of the assets of the obligors are driven by a common factor Y which has distribution function $G(y)$, and a idiosyncratic noise component ϵ_n which is distributed according to the distribution function $H(\epsilon_n)$*

$$V_n(t) = \sqrt{\rho_n}\, Y + \sqrt{1-\rho_n}\, \epsilon_n \quad n \leq N \tag{4.2.1}$$

where $Y \sim G$ and ϵ_n, $n \leq N$ are independent and identically $H(\epsilon_n)$-distributed with mean 0 and variance 1 and $\rho \in [0,1]$.

Therefore, the generalized default probability or conditional default risk is given by

$$p_n(y) = H\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right) \tag{4.2.2}$$

and must satisfy

$$E[p_n(y)] = \int_{-\infty}^{\infty} H\left(\frac{\alpha_n - \sqrt{\rho_n}\, y}{\sqrt{1-\rho_n}}\right) G'(y)\, dy = p_n \tag{4.2.3}$$

For the sake of simplicity, and without loss of generality, let us assume $p_n = p$ for all $n \in \{1, \ldots, N\}$, and consequently, $p_n(y) = p(y)$ for all $n \in \{1, \ldots, N\}$. Furthermore, let us consider a portfolio with a common asset correlation ρ, such that $\rho_n = \rho$ for all $n \in \{1, \ldots, N\}$. We first propose a numerical method for solving the previous equation for $\alpha_n = \alpha$ and finally compute the generalized default probability function given by

$$P[X = n] = \int_{-\infty}^{\infty} \binom{N}{n} \left(H\left(\frac{\alpha - \sqrt{\rho}\, y}{\sqrt{1-\rho}}\right)\right)^n \left(1 - H\left(\frac{\alpha - \sqrt{\rho}\, y}{\sqrt{1-\rho}}\right)\right)^{N-n} G'(y)\, dy \tag{4.2.4}$$

For some cases (4.2.3) can be easily solved for α, for example when considering the one-factor Gaussian model. In this special case we can make use of the following identity,

$$\int_{-\infty}^{\infty} \Phi\left(\frac{z-\mu}{\sigma}\right) \phi(z)\, dz = \Phi\left(\frac{-\mu}{\sqrt{\sigma^2+1}}\right) \tag{4.2.5}$$

$$\mu = \frac{-\alpha}{\sqrt{\rho}}, \quad \sigma = \frac{\sqrt{1-\rho}}{\sqrt{\rho}}$$

$$\begin{aligned} E[p_n(y)] &= \Phi\left(-\frac{-\alpha/\sqrt{\rho}}{(\sqrt{1-\rho}/\sqrt{\rho})^2+1)^{1/2}}\right) = \Phi\left(\frac{\alpha}{\sqrt{1-\rho/\rho+1}\sqrt{\rho}}\right) = \Phi\left(\frac{\alpha}{\sqrt{1/\rho}\sqrt{\rho}}\right) \\ &= \Phi(\alpha) = p_n \Leftrightarrow \alpha = \Phi^{-1}(p) \end{aligned}$$

Thus, in the one-factor Gaussian model, α only depends on the individual probability of default p, for example $\alpha = \Phi^{-1}(0.15) = -1.036433389$.

4.2.1 Numerical method for computing the threshold

In general solving Equation (4.2.3) is difficult when dealing with non-normal distributions. We next propose several steps for solving it numerically. We start by applying a change of variable and a posterior discretization scheme, which allows the use of root-finding algorithms such as Newton-Raphson to find α. In this first part we present the quadrature rule called Double-Exponential transformation.

The double-exponential transformation (DE transformation)

The double-exponential (DE) transformation was first proposed by Takahasi and Mori in [36] for the efficient evaluation of integrals of an analytic function with end-point singularities. Application of the double-exponential transformation on Sinc numerical methods can be found in [26]. This transformation is characterized by having double exponential asymptotic behaviour of the integrands in the resulting infinite integrals. For integrals over an infinite interval $(-\infty, \infty)$,

$$I = \int_{-\infty}^{\infty} f(x)\, dx \tag{4.2.6}$$

the double-exponential transformation consist on applying the following transformation

$$x = \psi(t) = \sinh\left(\frac{\pi}{2}\sinh t\right) \tag{4.2.7}$$

$$w = \psi'(t) = \frac{\pi}{2}(\cosh t)\cosh\left(\frac{\pi}{2}\sinh t\right) \tag{4.2.8}$$

Then one obtains the approximation

$$I_h = \frac{\pi}{2}h\sum_{k=-\infty}^{\infty} f\Big(\sinh\left(\frac{\pi}{2}\sinh kh\right)\Big)(\cosh kh)\cosh\left(\frac{\pi}{2}\sinh kh\right) \tag{4.2.9}$$

The double exponential transformation is one of the best quadrature formulas for integrals over $(-\infty, \infty)$. Formulas (4.2.7) and (4.2.8) are used in order to change a slowly convergent integral into a rapidly convergent one. As explained in [36], there is a loss of figures due to cancellation occurring as x tends to the endpoints of the interval, however this can be mitigated for some cases. In actual computation of (4.2.9) we truncate the infinite summation at $k = -N_-$ and $k = N_+$ so the total number of function evaluations is $-N_- + N_+ + 1$. Applying this truncation in (4.2.9),

$$I \approx I_h^N = \frac{\pi}{2}h\sum_{k=-N_-}^{N_+} w_k f(x_k) \tag{4.2.10}$$

Note, that the value of the integral I depends on two parameters N and h. N is the number of discretizations and h is the mesh size. As described in [36] a good approximation for h is

$$h \approx N^{-\frac{m}{m+1}} \tag{4.2.11}$$

and the error formula in terms of N

$$|\Delta I_N| \approx e^{-2\pi(\sin|\tau_1|/m)N^{-\frac{m}{m+1}})}, \quad |\tau_1| < \pi/2 \tag{4.2.12}$$

where m is a value such that the truncation error $|\Delta I_t|$ tends to 0 as m tends to infinity.

$$\Delta I_t \approx e^{-N^m h^m} \tag{4.2.13}$$

Alternatively we can use a similar DE transformation given by

$$I \approx I_h^N = \frac{\pi}{2}h\sum_{k=-ta}^{ta} w_k f(x_k) \tag{4.2.14}$$

where $ta = (d+1)^{0.46}$, d=digits of precision and $h = \frac{2ta}{N-1}$. We now apply trapezoidal rule obtaining the following integral approximation

$$I_h^N = \frac{\pi}{2}h\sum_{k=1}^{N} w_k f(x_k) \tag{4.2.15}$$

where $t_k = -ta + (k-1)h$ and

$$x_k = \sinh\left(\frac{\pi}{2}\sinh(t_k)\right) \quad w_k = \cosh(t_k)\cosh\left(\frac{\pi}{2}\sinh(t_k)\right) \tag{4.2.16}$$

We tested both quadrature rules and the latter might be slightly faster in some cases, nevertheless both DE quadratures tend to converge fast. As shown later, we might encounter some problem when solving integrals with functions with less precision, since these function are limiting the total achievable accuracy. Nevertheless, we are working with a global accuracy within the range $(1e{-}10, 1e{-}8)$, therefore, a similar order of magnitude is expected. In general, for most integrals $m = 12$ is sufficient to achieve a high accuracy, as suggested in [2].

Let us solve two integrals using the DE quadrature to test the efficiency of the method. In both cases an analytical result is known, which facilitates the checking of results.

$$I_1 = \int_{-\infty}^{\infty} \frac{1}{1+x^4}\,dx = \frac{\pi}{\sqrt{2}} \approx 2.221441469079183123507$$

$$I_2 = \int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi} \approx 1.772453850905516027298$$

Integral	**N**	**Rel.error**
I_1	121	$1.19 \cdot 10^{-8}$
I_1	241	$1.99 \cdot 10^{-16}$
I_2	121	$1.92 \cdot 10^{-8}$
I_2	241	$1.75 \cdot 10^{-14}$
I_2	361	$1.25 \cdot 10^{-16}$

Table 4.2.1: DE quadrature accuracy. N is the number of function evaluations. $m = 12$.

By only 241 function evaluations the approximated value has an absolute error of $1.99{\cdot}10^{-16}$, which is quite remarkable.

Steps of the calculation method

1. Apply DE transformation (4.2.15) to expression (4.2.3)

$$E[p(y)] \approx \frac{\pi}{2}h\sum_{k=1}^{N} w_k H\left(\frac{\alpha - \sqrt{\rho}x_k}{\sqrt{1-\rho}}\right)g(x_k) = p \tag{4.2.17}$$

where $t_k = -ta + (k-1)h$ and

$$x_k = \sinh\left(\frac{\pi}{2}\sinh(t_k)\right) \quad w_k = \cosh(t_k)\cosh\left(\frac{\pi}{2}\sinh(t_k)\right) \tag{4.2.18}$$

As shown in Table 4.2.2, a discretization with $N = 100$ and $d = 11$ digits of precision equivalent to $ta \approx 3.14$ is sufficient to achieve high accuracy. Other discretization schemes are possible, we might also choose the midpoint Riemann sum for two reasons; its simplicity and the fact that does not evaluate the integrand on the endpoints, avoiding singularities, but its convergence is slower. Hence, we use the latter discretization scheme during the validation performed by AMPL in Subsection 4.2.1.

$$E[p(y)] \approx \frac{2\pi}{N}\sum_{i=1}^{N} H\left(\frac{\alpha - \sqrt{\rho}\,\tan(x_i)}{\sqrt{1-\rho}}\right)\frac{g(\tan(x_i))}{\cos(2x_i)+1} = p \tag{4.2.19}$$

where

$$\Delta x = \frac{b-a}{N} = \frac{\pi}{N} \quad x_i = \frac{-\pi}{2} + (i-0.5)\Delta x \tag{4.2.20}$$

2. Root-finding algorithm.

   In order to solve the previous equation for α numerically, we use root-finding algorithms. The first one is a polyalgorithm formed by the Bisection algorithm and the Newton-Raphson algorithm, which is described in detail. In the second approach, we use Brent's method. Finally, a benchmark in terms of computational time and accuracy is presented.

   We apply the Bisection algorithm to obtain an initial guess, α_0, followed by the Newton-Raphson method that converges quadratically. The lower bound and upper bound selected as inputs to the Bisection algorithm are:

$$lb = \Phi^{-1}(1e{-}9) \approx -6, \qquad ub = \Phi^{-1}(1 - 1e{-}9) \approx 6 \tag{4.2.21}$$

   We assume that these quantiles represent a sufficient range so that the optimal α^* is within the interval. We can assume this, since both Y and ϵ_n are centered and standardized random variables. As shown in (4.2.24) the required accuracy for finding the initial guess α_0 is $1e{-}01$ due to we just need an initial point sufficiently close to the optimal α^*. The Bisection algorithm converges linearly; therefore, one might want to minimize the number of iterations during the searching procedure while guaranteeing a good initial guess. In order to apply Newton's method we need the function and its first derivative. The derivative with respect to α is straightforward giving a final expression that can be easily modified by changing the distribution function, whose derivative is the corresponding density function.

$$f(\alpha;\rho,p) = \frac{\pi}{2} h \sum_{k=1}^{N} w_k H\left(\frac{\alpha - \sqrt{\rho} x_k}{\sqrt{1-\rho}}\right) g(x_k) - p \tag{4.2.22}$$

$$f'(\alpha;\rho,p) = \frac{\partial f(\alpha,\rho,p)}{\partial \alpha} = \frac{\pi h}{2\sqrt{1-\rho}} \sum_{k=1}^{N} w_k H'\left(\frac{\alpha - \sqrt{\rho} x_k}{\sqrt{1-\rho}}\right) g(x_k) \tag{4.2.23}$$

$$\alpha_{k+1} = \alpha_k - \frac{f(\alpha_k)}{f'(\alpha_k)} \qquad \alpha_0 = \texttt{Bisection}(f(\alpha), lb, ub, 1e{-}01) \tag{4.2.24}$$

   In this case the derivative $f'(\alpha,\rho,p)$ is expensive to compute, as much as the function $f(\alpha,\rho,p)$. Due to the derivative is not an analytical expression, we can calculate the derivative numerically by finite difference. Therefore, $f'(\alpha,\rho,p)$ can be approximated by a second order centered finite difference approximation [2].

$$f'(\alpha;\rho,p) \approx \frac{f(a+h) - f(a-h)}{2h} + \mathcal{O}(h^2) \tag{4.2.25}$$

   or a more accurate fourth order finite difference approximation

$$f'(\alpha;\rho,p) \approx \frac{-f(a+2h) + 8f(a+h) - 8f(a-h) + f(a-2h)}{12h} + \mathcal{O}(h^4) \tag{4.2.26}$$

   The Brent's method is a root-finding algorithm which combines root bracketing, bisection and inverse quadratic interpolation. This method find a zero of the function f on the sign changing interval $[a,b]$. The algorithm switches between the potentially fast-converging secant method or inverse quadratic interpolation and a more robust bisection method,

[2] The value of the parameter h is $u^{2/3}$ where u is the **unit roundoff** typically 10^{-16} in double-precision arithmetic. See ([27], chap. 7) for further details about numerical differentiation.

depending on which is performing better. The interval is $[-6, 6]$ as previously stated. See [7] for a complete description of the algorithm.

Discretization	Method	Iterations	Abs.Error	Time (milliseconds)
Riemann	B+NR	11	$5.23 \cdot 10^{-11}$	8.70
	B+NRFD	11	$5.23 \cdot 10^{-11}$	10.71
	Brent	15	$5.23 \cdot 10^{-11}$	5.99
DE	B+NR	23	$1.78 \cdot 10^{-10}$	5.45
	B+NRFD	11	$1.78 \cdot 10^{-10}$	3.28
	Brent	15	$\mathbf{1.78 \cdot 10^{-11}}$	**1.98**

Table 4.2.2: Benchmark for computing the threshold α for two discretization schemes and 3 root-finding algorithms. B+NR = Bisection + Newton-Raphson, B+NRFD = Bisection + Newton-Raphson with centered finite difference order 2. Tolerance=$1e{-}15$. $\alpha_0 = -1.03125$. DE: $N = 100$, $d = 11$. Riemann midpoint: $N = 100$. Gaussian factor model: $Y \sim \mathcal{N}(0, 1)$ and $\epsilon_n \sim \mathcal{N}(0, 1)$ where $\alpha^* = \Phi^{-1}(0.15)$.

As shown in Table 4.2.2 the double exponential discretization scheme + Brent's method is the best method in terms of accuracy and computational time [3]. In general the DE transformation allows a substantial reduction in terms of computational time, which is desired when computing large homogeneous portfolios. Furthermore, the centered finite difference of order four does not improve the accuracy but increases the computation time, therefore was not included. The method presented applies to a wide class of continuous distribution with support $x \in \mathbb{R}$.

Validation: unconstrained nonlinear optimization problem

In order to check the obtained result applying the presented method, we solve the same problem by means of optimization techniques. Equation (4.2.3) can be transformed into an unconstrained nonlinear optimization problem, which is solved using AMPL + Ipopt, a software package for large-scale nonlinear optimization. We consider Equation (4.2.19)

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \left(\frac{2\pi}{N} \sum_{i=1}^{N} H\left(\frac{\alpha - \sqrt{\rho}\,\tan(x_i)}{\sqrt{1-\rho}} \right) \frac{g(\tan(x_i))}{\cos(2x_i) + 1} - p \right)^2 \tag{4.2.27}$$

We use the AMPL Extended Function library. This library extends AMPL with over 300 function chosen from the GNU Scientific library, including functions for generating random variates and computing their probability distributions.

Listing 4.1: Example Normal case

```
load amplgsl_64.dll; function gsl_ran_gaussian_pdf; function
    gsl_cdf_gaussian_P;

param pi := atan(1)*4;
param rho := 0.0;
param p := 0.15;
param lbound := -pi/2; # tan(-infinity)
param ubound := pi/2;  # tan(+infinity)
param N := 100;
param y {i in 1..N} = lbound + (i-0.5)*(ubound - lbound)/N;
var alpha := -1.03125;
```

[3] Intel CPU i5-3317U, 1.70GHz processor and 4GB RAM, and using R-3.1.2

```
minimize I: (2*pi/N * sum{i in 1..N} (gsl_cdf_gaussian_P((alpha - sqrt(rho)*
    tan(y[i]))/sqrt(1-rho), 1) * gsl_ran_gaussian_pdf(tan(y[i]), 1) * 1/(cos
    (2*y[i]) + 1)) - p)^2
```

iteration	objective	infeasibility
0	$1.25 \cdot 10^{-1}$	$2.79 \cdot 10^{-1}$
1	$1.61 \cdot 10^{-3}$	$2.18 \cdot 10^{-2}$
2	$4.45 \cdot 10^{-5}$	$3.20 \cdot 10^{-3}$
3	$7.30 \cdot 10^{-8}$	$1.26 \cdot 10^{-4}$
4	$2.35 \cdot 10^{-13}$	$2.26 \cdot 10^{-7}$
5	$2.46 \cdot 10^{-24}$	$7.32 \cdot 10^{-13}$

Table 4.2.3: Ipopt solver. Optimality tolerance = $1e{-}12$. Total iterations = 5, total number of objective function evaluations = 6. $\alpha_0 = 0$ and $\alpha^* = -1.036433389$.

In AMPL, if a variable is not assigned an initial value, then it has an initial value of zero ($\alpha_0 = 0$) [13]. As shown in Table 4.2.4, if we provide the guess obtained by the Bisection method, the number of iterations is reduced. Given an optimality tolerance, the number of iterations required by Ipopt is slightly lower compared to the presented numerical method. On the other hand, our method is easier to implemented than the interior-point algorithm implemented in Ipopt, so the use of solvers is likely to be excessive. Nevertheless, these results are used as confirmation that our method is correct.

iteration	objective	infeasibility
0	$1.47 \cdot 10^{-6}$	$5.68 \cdot 10^{-4}$
1	$9.26 \cdot 10^{-11}$	$4.49 \cdot 10^{-6}$
2	$3.81 \cdot 10^{-19}$	$2.88 \cdot 10^{-10}$
3	$7.70 \cdot 10^{-34}$	$1.29 \cdot 10^{-17}$

Table 4.2.4: Ipopt solver. Optimality tolerance = $1e{-}12$. Total iterations = 3, total number of objective function evaluations = 4. $\alpha_0 = -1.03125$ and $\alpha^* = -1.036433389$.

4.2.2 Special factor models

Until now we have computed portfolios assuming a systematic factor and idiosyncratic factor both normally distributed. In order to extend the one-factor model, we test the presented method for other continuous distributions with support $x \in \mathbb{R}$. Hence, we choose some well known distributions such as the logistic distribution and exponentially modified Gaussian distribution.

Logistic distribution

The probability density function and the cumulative distribution function which is the logistic function,

$$f(y;\mu,s) = \frac{e^{-\frac{y-\mu}{s}}}{s(1+e^{-\frac{y-\mu}{s}})^2} = \frac{1}{4s}\text{sech}^2\left(\frac{y-\mu}{2s}\right) \tag{4.2.28}$$

$$F(y;\mu,s) = \frac{1}{1+e^{-\frac{y-\mu}{s}}} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{y-\mu}{2s}\right) \tag{4.2.29}$$

As previously stated, Y and ϵ_n are centered and standardized random variables, then we have to choose the parameters for each factor such that they have zero mean and unit variance.

$$E[Y] = \mu = 0; \quad Var[Y] = 1 \Leftrightarrow \frac{s^2\pi^2}{3} = 1 \Leftrightarrow s = \frac{\sqrt{3}}{\pi} \tag{4.2.30}$$

Let us consider a systematic factor Y following a logistic distribution and idiosyncratic factor ϵ normally distributed, hence $Y \sim Logistic(0, \sqrt{3}/\pi)$ and $H \sim \mathcal{N}(0,1)$. Substituting in equations (4.2.22) and (4.2.23) the resulting equation and its derivative are given by

$$\begin{aligned} f(\alpha;\rho,p) &= \frac{\pi}{2}h\sum_{k=1}^{N} w_k \Phi\left(\frac{\alpha-\sqrt{\rho}x_k)}{\sqrt{1-\rho}}\right)\frac{\pi}{4\sqrt{3}}\text{sech}^2\left(\frac{x_k\pi}{2\sqrt{3}}\right) - p \\ &= \frac{\pi^2 h}{8\sqrt{3}}\sum_{k=1}^{N} w_k \Phi\left(\frac{\alpha-\sqrt{\rho}x_k)}{\sqrt{1-\rho}}\right)\text{sech}^2\left(\frac{x_k\pi}{2\sqrt{3}}\right) - p \end{aligned} \tag{4.2.31}$$

$$f'(\alpha;\rho,p) = \frac{\pi^{3/2}h}{8\sqrt{6-6\rho}}\sum_{k=1}^{N} w_k e^{-\frac{(\alpha-\sqrt{\rho}x_k)^2}{2-2\rho}}\text{sech}^2\left(\frac{x_k\pi}{2\sqrt{3}}\right) \tag{4.2.32}$$

			Newton-Raphson		**Ipopt Solver**	
p	ρ	B+NR iter [4]	NR iter ($\alpha_0 = 0$)	α	iter ($\alpha_0 = 0$)	α
0.05	0	12	7	-1.644853627	7	-1.644853627
0.05	0.1	11	7	-1.644556971	7	-1.644556971
0.05	0.5	11	7	-1.636491034	7	-1.636491034
0.05	0.9	11	7	-1.624527893	7	-1.624527893
0.15	0	11	6	-1.036433389	5	-1.036433389
0.15	0.1	11	6	-1.035483079	5	-1.035483079
0.15	0.5	11	6	-1.015408800	5	-1.015408800
0.15	0.9	12	6	-0.971268644	5	-0.971268644
0.5	0	11	2	$9.019154e{-12}$	1	$9.019571e{-12}$
0.5	0.1	11	2	$9.006674e{-12}$	1	$9.006813e{-12}$
0.5	0.5	11	2	$8.754275e{-12}$	1	$8.753869e{-12}$
0.5	0.9	11	2	$8.164863e{-12}$	1	$8.164485e{-12}$

Table 4.2.5: Threshold α computed by both methods. Small differences when the theoretical values is 0. The number of iterations with the same starting point are almost equal.

An important point is the situation when $p = 0.5$, the theoretical value of α is 0 or a very close number for $\rho \neq 0$ (if $\rho = 0$ then $\alpha = \Phi^{-1}(0.5) = 0$), however, both algorithms stop at the required optimality tolerance, as expected. Another way of checking the obtained results is by solving Equation (4.2.3) when $\rho = 0$. This result might be useful for more complicated H distributions.

$$\int_{-\infty}^{\infty} H\left(\frac{\alpha-\sqrt{\rho}\,y}{\sqrt{1-\rho}}\right)g(y)\,dy = H(\alpha)\int_{-\infty}^{\infty} g(y)\,dy = H(\alpha) = p \Longrightarrow \alpha = H^{-1}(p) \tag{4.2.33}$$

Furthermore, we can consider a idiosyncratic factor also following a centered and standardized logistic distribution, $Y \sim Logistic(0, \sqrt{3}/\pi)$ and $H \sim Logistic(0, \sqrt{3}/\pi)$. We refer to this model as *Logistic-factor model*.

$$f(\alpha;\rho,p) = \frac{\pi^2 h}{16\sqrt{3}}\sum_{k=1}^{N} w_k\left(1+\tanh\left(\frac{\pi(\alpha-\sqrt{\rho}x_k)}{2\sqrt{3-3\rho}}\right)\right)\text{sech}^2\left(\frac{x_k\pi}{2\sqrt{3}}\right) - p \tag{4.2.34}$$

$$f'(\alpha;\rho,p) = \frac{\pi^3 h}{96\sqrt{1-\rho}}\sum_{k=1}^{N} w_k\text{sech}^2\left(\frac{\pi(\alpha-\sqrt{\rho}x_k)}{2\sqrt{3-3\rho}}\right)\text{sech}^2\left(\frac{x_k\pi}{2\sqrt{3}}\right) \tag{4.2.35}$$

[4] B + NR = Bisection algorithm + Newton-Raphson method.

ρ	iterations	α
0	12	-0.956335684
0.1	12	-0.970082643
0.2	11	-0.980138229
0.3	11	-0.986974875
0.4	11	-0.990947388
0.5	11	-0.992250996
0.6	11	-0.990947388
0.7	11	-0.986974875
0.8	11	-0.980138229
0.9	12	-0.970082643

Table 4.2.6: Method: Newton-Raphson. $p = 0.15$. 5 Ipopt iterations for all cases. When $\rho = 0 \Leftrightarrow \alpha = H^{-1}(p = 0.15, 0, \frac{\sqrt{3}}{\pi}) = -0.956335684$.

Exponentially modified Gaussian distribution (EMG)

The EMG is a mixture of an exponential and Gaussian distribution and it has a characteristic positive skew from the exponential distribution. The distribution has three parameters,

- $\mu \in \mathbb{R}$: mean of the normal distribution
- $\sigma > 0$: standard deviation of the normal distribution
- $\lambda > 0$: Rate of the exponential component

The probability density function and the cumulative distribution are as follows,

$$f(y; \mu, \sigma, \lambda) = \frac{\lambda}{2} e^{\frac{\lambda}{2}(2\mu + \lambda\sigma^2 - 2y)} \text{erfc}\left(\frac{\mu + \lambda\sigma^2 - y}{\sqrt{2}\sigma}\right) \tag{4.2.36}$$

$$F(y; \mu, \sigma, \lambda) = \Phi(\lambda(y - \mu), 0, \lambda\sigma) - e^{-\lambda(y-\mu) + \frac{(\lambda\sigma)^2}{2} + \ln(\Phi(\lambda(y-\mu), (\lambda\sigma)^2, \lambda\sigma))} \tag{4.2.37}$$

Since we have three parameters available, we can choose one of them and determine the other two such that the required assumptions are satisfied, zero mean and unit variance

$$E[Y] = \mu + \frac{1}{\lambda} = 0 \Leftrightarrow \lambda = -\frac{1}{\mu} > 0 \tag{4.2.38}$$

$$Var[Y] = \sigma^2 + \frac{1}{\lambda^2} = 1 \Leftrightarrow \sigma = \sqrt{1 - \mu^2} > 0 \tag{4.2.39}$$

Hence,

$$f(y; \mu) = -\frac{1}{2\mu} e^{-\frac{1}{2\mu}(3\mu - \frac{1}{\mu} - 2y)} \text{erfc}\left(\frac{2\mu - \frac{1}{\mu} - y}{\sqrt{2(1 - \mu^2)}}\right) \tag{4.2.40}$$

$$F(y; \mu) = \Phi(u, 0, v) - e^{-u + \frac{v^2}{2} + \ln(\Phi(u, v^2, v))} \tag{4.2.41}$$

where $u = \lambda(y - \mu) = 1 - \frac{y}{\mu}$, $v = \lambda\sigma = -\frac{\sqrt{1-\mu^2}}{\mu}$ and $v^2 = \frac{1-\mu^2}{\mu^2}$.

Given the domain of σ and λ, μ is bounded such that $\mu \in (-1, 0)$. Figure 4.3 shows various possible probability density functions satisfying this open interval. We can see how the EMG distribution is closer to the standard normal distribution as μ tends to 0. Table 4.2.7 shows this behaviour. Therefore, we have gained flexibility to correctly fit the factor data (stock returns,

for instance). On the other hand, we can choose the Gaussian factor when μ and ρ are close to 0 as a valid simplification.

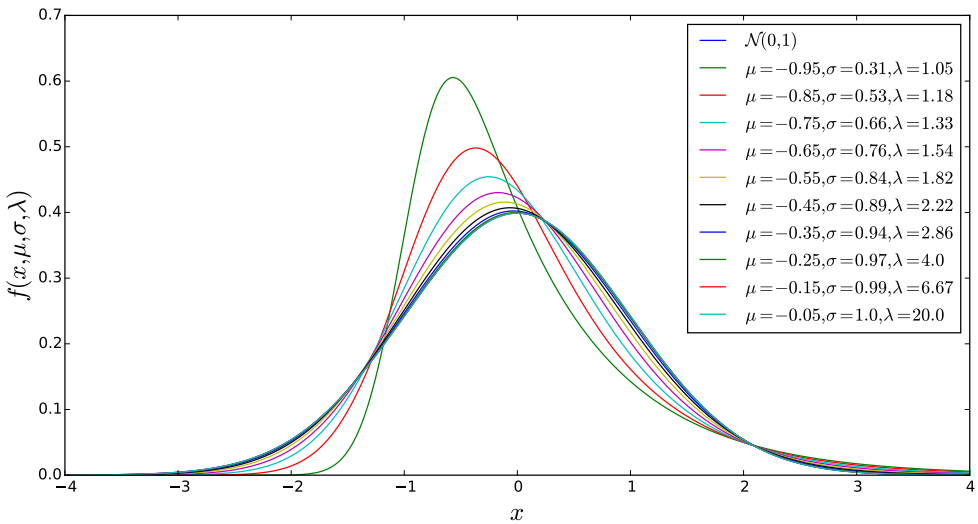


Figure 4.3: Centered and standardized EMG distribution with $\mu \in (-1, 0)$.

	one degree of freedom (μ)				
ρ	-0.95	-0.75	-0.5	-0.25	-0.05
0	-1.03643338838	-1.03643338985	-1.03643338951	-1.03643338949	-1.03643338948
0.15	-1.02969131119	-1.03343565972	-1.03570793529	-1.03637172805	-1.03643314098
0.25	-1.02047317147	-1.02924103881	-1.0346534512	-1.03628031651	-1.03643281286
0.35	-1.00841163577	-1.02373668747	-1.03324129491	-1.03615465463	-1.03643237844
0.5	-0.985397013377	-1.01335591776	-1.03054718408	-1.03590805474	-1.03643154432

Table 4.2.7: Threshold value when $Y \sim EMG(\mu)$ and $H \sim \mathcal{N}(0,1)$ for different correlations ρ and $p = 0.15$. Note $\alpha \to \Phi^{-1}(p)$ as $\mu \to 0$.

Let us consider a systematic factor Y following a EMG distribution and idiosyncratic factor ϵ normally distributed, hence $Y \sim EMG(\mu, \sqrt{1-\mu^2}, -1/\mu)$ and $H \sim \mathcal{N}(0,1)$.

$$f(\alpha;\rho,p,\mu) = \frac{\pi}{2} h \sum_{k=1}^{N} w_k \Phi\left(\frac{\alpha - \sqrt{\rho} x_k}{\sqrt{1-\rho}}\right) f_{EMG}(x_k;\mu,\sqrt{1-\mu^2},-1/\mu) - p \tag{4.2.42}$$

$$f'(\alpha;\rho,p,\mu) = \frac{\sqrt{\pi} h}{2\sqrt{2-2\rho}} \sum_{k=1}^{N} w_k e^{-\frac{(\alpha-\sqrt{\rho}x_k)^2}{2-2\rho}} f_{EMG}(x_k;\mu,\sqrt{1-\mu^2},-1/\mu) \tag{4.2.43}$$

Furthermore, we can consider a idiosyncratic factor also following a centered and standardized EMG distribution, $Y \sim EMG(\mu_1, \sqrt{1-\mu_1^2}, -1/\mu_1)$ and $H \sim EMG(\mu_2, \sqrt{1-\mu_2^2}, -1/\mu_2)$. We refer to this model as *EMG-factor model*.

$$f(\alpha;\rho,p,\mu_1,\mu_2) = \frac{\pi}{2} h \sum_{k=1}^{N} w_k F_{EMG}(t_k,\mu_2,\sqrt{1-\mu_2^2},-1/\mu_2) f_{EMG}(x_k;\mu_1,\sqrt{1-\mu_1^2},-1/\mu_1) - p \tag{4.2.44}$$

$$f'(\alpha;\rho,p,\mu_1,\mu_2) = \frac{\pi h}{2\sqrt{1-\rho}}\sum_{k=1}^{N} w_k f_{EMG}(t_k,\mu_2,\sqrt{1-\mu_2^2},-1/\mu_2) f_{EMG}(x_k;\mu_1,\sqrt{1-\mu_1^2},-1/\mu_1)$$
$$= \frac{\pi h}{8\sqrt{1-\rho}\mu_1\mu_2}\sum_{k=1}^{N} w_k e^{-\frac{1}{2}(6-\frac{1}{\mu_1^2}-\frac{1}{\mu_2^2}-2(\frac{x_k}{\mu_1}+\frac{t_k}{\mu_2}))}\mathrm{erfc}(z_1^k)\mathrm{erfc}(z_2^k) \tag{4.2.45}$$

where

$$z_1^k = \frac{2\mu_1 - 1/\mu_1 - x_k}{\sqrt{2-2\mu_1^2}}, \quad z_2^k = \frac{2\mu_2 - 1/\mu_2 - t_k}{\sqrt{2-2\mu_2^2}} \quad \text{and} \quad t_k = \frac{\alpha-\sqrt{\rho}x_k}{\sqrt{1-\rho}} \tag{4.2.46}$$

if $\mu_1 = \mu_2 = \mu$ then the previous expression can be simplified

$$f'(\alpha;\rho,p,\mu) = \frac{\pi h}{8\sqrt{1-\rho}\mu^2}\sum_{k=1}^{N} w_k e^{-\frac{1}{2}(6-\frac{1}{2\mu^2}-2(\frac{x_k+t_k}{\mu}))}\mathrm{erfc}(z_1^k)\mathrm{erfc}(z_2^k) \tag{4.2.47}$$

Normal Inverse Gaussian distribution (NIG)

We introduced the NIG distribution in Subsection 2.3 where we presented several methods for computing the cumulative distribution function and the quantile function.

$$Var[X] = \frac{\delta\alpha^2}{(\alpha^2-\beta^2)^{3/2}} = 1 \Leftrightarrow \delta = \frac{(\alpha^2-\beta^2)^{3/2}}{\alpha^2} \tag{4.2.48}$$

$$E[X] = \mu + \frac{\delta\beta}{\sqrt{\alpha^2-\beta^2}} = 0 \Leftrightarrow \mu = -\frac{\delta\beta}{\sqrt{\alpha^2-\beta^2}} = -\frac{\beta(\alpha^2-\beta^2)}{\alpha^2} \tag{4.2.49}$$

if $\beta = 0 \Rightarrow \delta = \alpha$ and $\mu = 0$.

$$f_{NIG}(x;\alpha) = \frac{\alpha^2}{\pi}e^{\alpha^2}\frac{K_1(\alpha\sqrt{\alpha^2+x^2})}{\alpha\sqrt{\alpha^2+x^2}}, \quad \alpha > 0 \tag{4.2.50}$$

Let us consider a systematic factor Y following a NIG distribution and idiosyncratic factor ϵ normally distributed, hence $Y \sim NIG(\alpha_Y,\beta_Y)$ and $H \sim \mathcal{N}(0,1)$.

$$f(\alpha;\rho,p,\alpha_Y,\beta_Y) = \frac{\pi}{2}\sum_{k=1}^{N} w_k \Phi\left(\frac{\alpha-\sqrt{\rho}x_k}{\sqrt{1-\rho}}\right) f_{NIG}(x_k;\mu_Y,\delta_Y,\alpha_Y,\beta_Y) - p \tag{4.2.51}$$

where $\mu_Y = -\beta_Y(\alpha_Y^2-\beta_Y^2)/\alpha_Y^2$ and $\delta_Y = (\alpha_Y^2-\beta_Y^2)^{3/2}/\alpha_Y^2$

Case $\beta_Y = 0$

$$f(\alpha;\rho,p,\alpha_Y) = \frac{\alpha^2 h}{2}e^{\alpha_Y^2}\sum_{k=1}^{N} w_k \Phi\left(\frac{\alpha-\sqrt{\rho}x_k}{\sqrt{1-\rho}}\right)\frac{K_1(\alpha_Y\sqrt{\alpha_Y^2+x_k^2})}{\sqrt{\alpha_Y^2+x_k^2}} - p \tag{4.2.52}$$

$$f'(\alpha;\rho,p,\alpha_Y) = \frac{h\alpha_Y^2 e^{\alpha_Y^2}}{2\sqrt{2\pi-2\pi\rho}}\sum_{k=1}^{N} w_k e^{-\frac{(\alpha-\sqrt{\rho}x_k)^2}{2-2\rho}}\frac{K_1(\alpha_Y\sqrt{\alpha_Y^2+x_k^2})}{\sqrt{\alpha_Y^2+x_k^2}} \tag{4.2.53}$$

Furthermore, we can consider a idiosyncratic factor also following a centered and standardized NIG distribution, $Y \sim NIG(\alpha_Y,\beta_Y)$ and $H \sim NIG(\alpha_H,\beta_H)$. We refer to this model as *NIG-factor model*.

$$f(\alpha;\rho,p,\alpha_Y,\beta_Y,\alpha_H,\beta_H) = \frac{\pi}{2}h\sum_{k=1}^{N} w_k F_{NIG}(t_k;\mu_H,\delta_H,\alpha_H,\beta_H) f_{NIG}(xk;\mu_Y,\delta_Y,\alpha_Y,\beta_Y) - p \tag{4.2.54}$$

Case $\beta_Y = 0$

$$f(\alpha;\rho,p,\alpha_Y,\alpha_H) = \frac{\alpha_Y^2 h e^{\alpha_Y^2}}{2}\sum_{k=1}^{N} w_k F_{NIG}(tk;\alpha_H)\frac{K_1(\alpha_Y\sqrt{\alpha_Y^2+x_k^2})}{\sqrt{\alpha_Y^2+x_k^2}} \tag{4.2.55}$$

$$f'(\alpha;\rho,p,\alpha_Y,\alpha_H) = \frac{h\alpha_Y^2\alpha_H^2 e^{\alpha_Y^2+\alpha_H^2}}{2\pi\sqrt{1-\rho}}\sum_{k=1}^{N}\frac{K_1(\alpha_Y\sqrt{\alpha_Y^2+x_k^2})}{\sqrt{\alpha_Y^2+x_k^2}}\frac{K_1(\alpha_H\sqrt{\alpha_H^2+t_k^2})}{\sqrt{\alpha_H^2+t_k^2}} \tag{4.2.56}$$

where $t_k = \frac{\alpha-\sqrt{\rho}x_k}{\sqrt{1-\rho}}$.

As explained in [23], we might consider the properties of the NIG distribution in order to reduce the computational time and difficulty of computing the threshold. From the one-factor model $V_n = \sqrt{\rho_n}Y + \sqrt{1-\rho_n}\,\epsilon_n$ where $Y, H(\epsilon_n)-$distributed, $i = 1,\ldots,N$ are independent NIG random variables

$$Y \sim NIG\left(\alpha_Y,\beta_Y,-\frac{\beta_Y(\alpha_Y^2-\beta_Y^2)}{\alpha_Y^2},\frac{(\alpha_Y^2-\beta_Y^2)^{3/2}}{\alpha_Y^2}\right) \tag{4.2.57}$$

$$H \sim NIG\left(\alpha_H,\beta_H,-\frac{\beta_H(\alpha_H^2-\beta_H^2)}{\alpha_H^2},\frac{(\alpha_H^2-\beta_H^2)^{3/2}}{\alpha_H^2}\right) \tag{4.2.58}$$

And given the scaling property of the NIG distribution

$$X \sim NIG(\alpha,\beta,\mu,\delta) \Rightarrow cX \sim NIG\left(\frac{\alpha}{c},\frac{\beta}{c},c\mu,c\delta\right) \tag{4.2.59}$$

$$\sqrt{\rho}Y \sim NIG\left(\frac{\alpha_Y}{\sqrt{\rho}},\frac{\beta_Y}{\sqrt{\rho}},-\frac{\sqrt{\rho}\beta_Y(\alpha_Y^2-\beta_Y^2)}{\alpha_Y^2},\frac{\sqrt{\rho}(\alpha_Y^2-\beta_Y^2)^{3/2}}{\alpha_Y^2}\right) \tag{4.2.60}$$

$$\sqrt{1-\rho}H \sim NIG\left(\frac{\alpha_H}{\sqrt{1-\rho}},\frac{\beta_H}{\sqrt{1-\rho}},-\frac{\sqrt{1-\rho}\beta_H(\alpha_H^2-\beta_H^2)}{\alpha_H^2},\frac{\sqrt{1-\rho}(\alpha_H^2-\beta_H^2)^{3/2}}{\alpha_H^2}\right) \tag{4.2.61}$$

Another relevant property of the NIG distribution is the closure under convolution for independent random variables X and Y

$$X \sim NIG(\alpha,\beta,\mu_1,\delta_1),\ Y \sim NIG(\alpha,\beta,\mu_2,\delta_2)$$

$$X+Y \sim NIG(\alpha,\beta,\mu_1+\mu_2,\delta_1+\delta_2) \tag{4.2.62}$$

we use the stability under convolution taking into account a simplification, $\frac{\alpha_Y}{\sqrt{\rho}} = \frac{\alpha_H}{\sqrt{1-\rho}}$ and $\frac{\beta_Y}{\sqrt{\rho}} = \frac{\beta_H}{\sqrt{1-\rho}}$, which reduce the flexibility to fitting data, therefore this approach is limited to specific models.

$$\frac{\alpha_Y}{\rho} = \frac{\alpha_H}{\sqrt{1-\rho}} \Rightarrow \alpha_H = \frac{\sqrt{1-\rho}}{\rho}\alpha_Y$$

Hence, for those cases V_n can be simplified as follows

$$V_n \sim NIG\left(\frac{\alpha}{\sqrt{\rho}},\frac{\beta}{\sqrt{\rho}},-\frac{1}{\sqrt{\rho}}\frac{\beta(\alpha^2-\beta^2)}{\alpha^2},\frac{1}{\sqrt{\rho}}\frac{(\alpha^2-\beta^2)^{3/2}}{\alpha^2}\right) \tag{4.2.63}$$

Thereby, the threshold α^* can be easily computed

$$\alpha^* = F_{NIG}^{-1}\left(p;\frac{\alpha}{\sqrt{\rho}},\frac{\beta}{\sqrt{\rho}},-\frac{1}{\sqrt{\rho}}\frac{\beta(\alpha^2-\beta^2)}{\alpha^2},\frac{1}{\sqrt{\rho}}\frac{(\alpha^2-\beta^2)^{3/2}}{\alpha^2}\right) \tag{4.2.64}$$

4.2.3 Generalized default probability function

For the moment we have been focused on the computation of the threshold α, but once we found an efficient algorithm, is time to tackle the computation of the integral (4.2.4),

$$P[X=n] = \int_{-\infty}^{\infty} \binom{N}{n} \left(H\left(\frac{\alpha - \sqrt{\rho}\, y}{\sqrt{1-\rho}} \right) \right)^n \left(1 - H\left(\frac{\alpha - \sqrt{\rho}\, y}{\sqrt{1-\rho}} \right) \right)^{N-n} g(y)\, dy$$

There are several numerical methods for solving this kind of integrals. We use the Gauss-Hermite quadrature in Subsection 4.1.2, taking advantage of the standard normal distribution. This method might be used for the Logistic and EMG distribution with some transformations, however, since other distributions can be chosen we prefer a more general method. Among all the possible methods, we chose the DE transformation, the Gauss-Legendre quadrature and Monte Carlo with Median Latin Hypercube sampling and we compare them. We believed that after the previous results of the computation of the threshold it was worth to test the DE transformation for more complicated integrals. As shown later, this guessing was positively reaffirmed.

Results

As previously stated, one of the methods applied to solve (4.2.4) is Monte Carlo + Median Latin Hypercube sampling (MC+MLHS). As shown in 3.2.1 the efficiency of MLHS is maximized when the bounds of the integral are finite, therefore, we apply the transformation (4.2.17), obtaining the following expression,

$$P[X=n] = 2 \int_{-\pi/2}^{\pi/2} \mathcal{B}\left(n, N, H\left(\frac{\alpha - \sqrt{\rho}\, \tan(x)}{\sqrt{1-\rho}} \right) \right) \frac{g(\tan(x))}{\cos(2x)+1}\, dx \tag{4.2.65}$$

where $\mathcal{B}(n, N, p)$ denotes the probability mass function of the binomial distribution. To apply the Gauss-Legendre quadrature we follow the steps presented in Subsection 2.3.1. Finally, we proceed to apply the DE transformation following the steps in Subsection 4.2.1. To perform our tests and do the comparison, we use 500 MC+MLHS trials, Gauss-Legendre quadrature with 60 nodes and DE transformation with $N = 300$ and $h = 12$.

In order to easily check the obtained results we consider a homogeneous portfolio of 100 assets with $p_n = p = 0.15$, $\rho_n = \rho = 0.1$ and all loans having the same size. We use these three measures for comparison:

1. Expected Loss (EL): $EL(L) = E[L]$ where L is the Portfolio Loss at time T.

$$E[L] = \langle l^s, p^s \rangle \tag{4.2.66}$$

   where l^s is the vector of losses for each scenario and p^s is the corresponding vector of probabilities. In the homogeneous portfolio the mean of the distribution is $EL = p$, therefore, this is a key measure for checking the quality of the method.

2. Portfolio standard deviation: $Stdev[L] = \sqrt{E[L^2] - E[L]^2}$

3. VaR_β: Value-at-Risk with confidence level $\beta = 99.9\%$. The β-quantile of the loss distribution $F(x)$

$$VaR_\beta(X) = inf\{x, F(x) \geq \beta\} = F^{-1}(\beta) \tag{4.2.67}$$

   VaR is a risk measure widely used to quantify the risk. This is the measure chosen in the Basel II Accord for the computation of capital requirements, meaning that a bank that manages its risks according to Basel II must reserve capital by an amount of $x_\beta = F^{-1}(\beta)$ to cover potential extreme losses.

Table 4.2.8 shows the results obtained with these three methods and the computational time [5]. The first result that may surprise us, is the computation time required by Monte Carlo + MLHS method. Although the Crude Monte Carlo method is very time consuming, we showed in Chapter 3 the variance reduction technique called Median Latin Hypercube sampling that reduces the number of required trials and thus the computation time. Therefore, it is possible to compute a portfolio accurately without functions for numerical quadrature.

Model	**Method**	**Mean**	**Stdev**	$\boldsymbol{VaR_{0.999}}$	**Time(s)**
$L-N$	MC+MLHS	0.1499999792	0.0840666153	0.592728	0.223
$L-N$	DE	0.1499999792	0.0840666153	0.592728	0.122
$L-N$	GL*	0.1499999614	0.0840665220	0.591216	0.018
$L-L$	MC+MLHS	0.1499999756	0.0852446935	0.629988	0.199
$L-L$	DE	0.1499999756	0.0852446935	0.629988	0.132
$L-L$	GL*	0.1499999610	0.0852446130	0.626293	0.020
$E-N$	MC+MLHS	0.1500000000	0.0690058883	0.363408	0.195
$E-N$	DE	0.1500000000	0.0690058883	0.363408	0.146
$E-N$	GL	0.1500000003	0.0690058895	0.363408	0.026
$E-E$	MC+MLHS	0.1500000000	0.1058402431	0.476118	0.255
$E-E$	DE	0.1500000000	0.1058402431	0.476118	0.191
$E-E$	GL	0.1500000005	0.1058402453	0.476121	0.032

Table 4.2.8: Metrics and computation time for generalized portfolios. $L = Logistic(0, \sqrt{3}/\pi)$. $N = \mathcal{N}(0,1)$. $E = EMG(\mu = -0.95)$. Methods (MC+MLHS(500), DE(12, 300) and GL(60)). GL* = GL(45) stopping after blow up singularity.

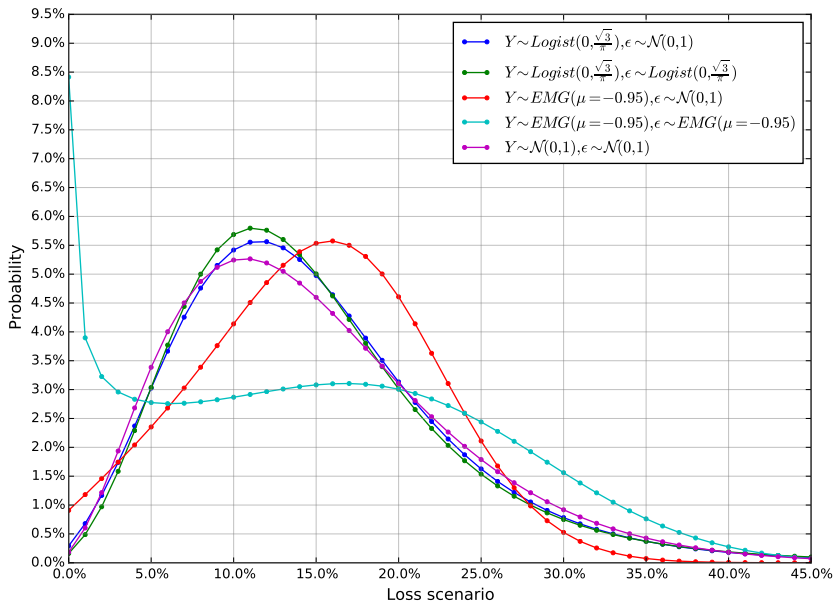


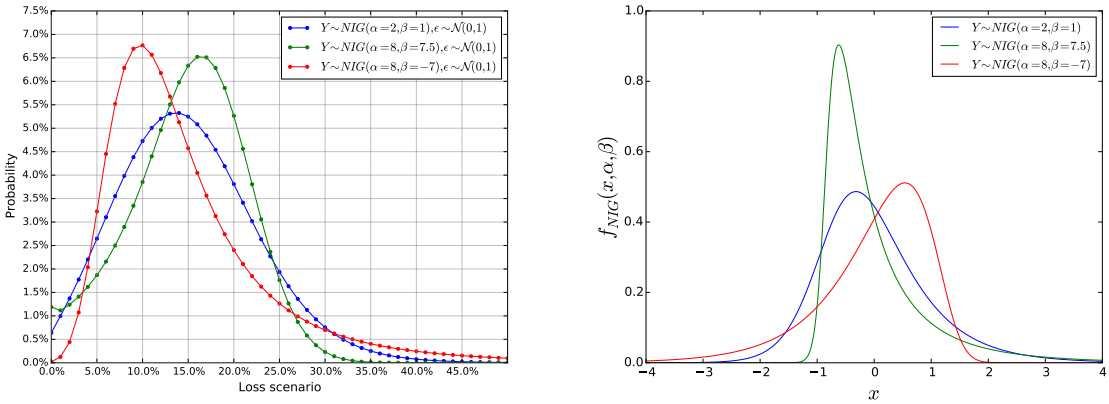
Figure 4.4: Probability distribution of the defaults of several special factor models.

The Gauss-Legendre quadrature stands out as the fastest method with a fairly good accuracy. However, in general, a higher number of quadrature nodes is needed to obtain the

[5] Intel CPU i5-3317U, 1.70GHz processor and 4GB RAM, and using Microsoft® C/C++ Optimizing Compiler Version 17.00.61030 for x86.

same level of precision of the DE transformation or MC+MLHS. The computation of a larger number of nodes tends to produce numerical instability and even computational issues with a straightforward implementation, so it requires the use of more involved methods like the one described in [18]. Thereby, the DE transformation would be the method of choice, since combines accurate results and a reasonable computation time.

The special factor model NIG-Normal requires a more detailed study due to its range of parameters. We tested three different models with different NIG distribution aiming to cover a representative spectrum of possibilities. As shown in Table 4.2.9, the accuracy diminishes with NIG distributions with negative skew, requiring a higher number of discretizations N.



(a) *Probability distribution.* (b) *Density of some NIG distribution.*

Figure 4.5: Probability distribution of the defaults of special factor model NIG-Normal.

Model	**Mean**	**Stdev**	$\boldsymbol{VaR_{0.999}}$	**Time(s)**
$\alpha = 2, \beta = 1$	0.1500000000	0.0752507418	0.452257	0.167
$\alpha = 8, \beta = 7.5$	0.1500000000	0.0627780245	0.329152	0.158
$\alpha = 8, \beta = -7$	0.1499951938	0.0958756491	0.765785	0.175

Table 4.2.9: Metrics and computation time for factor model: $Y \sim NIG(\alpha, \beta)$ and $H \sim \mathcal{N}(0, 1)$. Method DE(12, 300).

The previous results show different available methods that allow the computation of small-medium portfolios in a very reasonable time. Larger portfolios can also be computed using these methods, but as shown on the next Section, some closed-form expressions can be developed to approximate the distribution of the portfolios losses when considering homogeneous portfolios. Furthermore, it is important to mention the possibility to apply parallel computing, which can be accomplished easily for Monte Carlo methods or DE transformation, but might require a more involved method for the computation of Gauss-Legendre nodes and weights, see [5].

4.3 The large portfolio approximation

In this Section we introduce the so-called Vasicek Single Factor Model for a large portfolio of homogeneous assets. This model was initially provided by Vasicek in [39] and generalized by Schönbucher in [35]. The Single Factor Model increases the tractability of the one-factor model by assuming a portfolio of homogeneous assets ($p_n = p$ and $\rho_n = \rho$) composed by a number of obligors tending to infinity, given an explicit formula for the default distribution which is directly derived from the Law of Large Numbers. Our contribution is the introduction of four explicit formulas for the *Logistic-Normal factor model*, *Logistic-Logistic factor model*, *EMG-Normal factor model* and *EMG-EMG factor model*, respectively, which were explained in detail in Subsection 4.2.2.

4.3.1 Properties of the Vasicek distribution

Several assumptions are considered in the Single Factor Model:

Assumption 4.1 (The number of obligors $N \to \infty$) *Due to individual defaults are independent when conditioned to the realization of the common factor Y, by the Law of Large Numbers, the average of N fractions of defaulted obligors in the portfolio sharing the same probability X_n converges towards the individual default probability of each individual obligor $p(Y)$.*

For a given realization of the systematic factor $Y = y$, the individual default probability is given by

$$E\left[\sum_{n=1}^{N} \frac{X_n}{N} \mid Y = y\right] = p(y) = \Phi\left(\frac{\alpha - \sqrt{\rho} y}{\sqrt{1-\rho}}\right) \tag{4.3.1}$$

Assumption 4.2 (The loss given default is deterministic and homogeneous) *The loss of each obligor due to default is expressed as a percentage of its size and is common to all obligors in the porfolio.*

Assumption 4.3 (The contributions of each obligor are similar) *All the obligors in the portfolio have the same relative size $s_n = 1/N$, the same correlation $\rho_n = \rho$ and the same default threshold $\alpha_n = \alpha$. The convergence of the portfolio loss distribution actually holds for unequal relative sizes if the portfolio contains a sufficiently large number of obligors without it being dominated by a few obligors much larger than the rest, consequently*

$$\sum_{n=1}^{N} s_n^2 \to 0 \tag{4.3.2}$$

As a consequence of these three assumption the cumulative distribution function of portfolio losses can be constructed as follows

$$\begin{aligned}
P[p(y) \le x] &= P\left(\Phi\left(\frac{\Phi^{-1}(p) - \sqrt{\rho}\, y}{\sqrt{1-\rho}}\right) \le x\right) \\
&= P\left(\frac{\Phi^{-1}(p) - \sqrt{\rho}\, y}{\sqrt{1-\rho}} \le \Phi^{-1}(x)\right) \\
&= P\left(-y \le \frac{\Phi^{-1}(x)\sqrt{1-\rho} - \Phi^{-1}(p)}{\sqrt{\rho}}\right) \\
&= \Phi\left(\frac{\sqrt{1-\rho}\,\Phi^{-1}(x) - \Phi^{-1}(p)}{\sqrt{\rho}}\right)
\end{aligned}$$

The portfolio loss distribution given by the cumulative distribution function is a continuous distribution concentrated on the interval $x \in [0, 1]$.

$$F(x; p, \rho) = P[p(y) \leq x] = \Phi\left(\frac{\sqrt{1-\rho}\,\Phi^{-1}(x) - \Phi^{-1}(p)}{\sqrt{\rho}}\right) \tag{4.3.3}$$

which possesses a symmetry property

$$F(x; p, \rho) = 1 - F(1 - x; 1 - p, 1 - \rho) \tag{4.3.4}$$

The portfolio loss density function or Vasicek distribution is defined as a two-parametric ($0 < p < 1$ and $0 < \rho < 1$) continuous distribution given by the derivative of the portfolio loss distribution

$$\begin{aligned} f(x; p, \rho) &= \frac{\partial F(x; p, \rho)}{\partial x} \\ &= \sqrt{\frac{1-\rho}{\rho}} \frac{1}{\sqrt{2\pi}} e^{-\frac{(\sqrt{1-\rho}\,\Phi^{-1}(x) - \Phi^{-1}(p))^2}{2\rho}} \frac{\partial}{\partial x}\Phi^{-1}(x) \\ &= \sqrt{\frac{1-\rho}{\rho}} \frac{1}{\sqrt{2\pi}} e^{-\frac{(\sqrt{1-\rho}\,\Phi^{-1}(x) - \Phi^{-1}(p))^2}{2\rho}} \sqrt{2\pi} e^{\frac{(\Phi^{-1}(x))^2}{2}} \\ &= \sqrt{\frac{1-\rho}{\rho}} e^{\left(-\frac{(\sqrt{1-\rho}\,\Phi^{-1}(x) - \Phi^{-1}(p))^2}{2\rho} + \frac{(\Phi^{-1}(x))^2}{2}\right)} \end{aligned} \tag{4.3.5}$$

where,

$$\frac{\partial}{\partial x}\Phi^{-1}(x) = \frac{\partial}{\partial x}\sqrt{2}\,\mathrm{erf}^{-1}(2x-1) = \sqrt{2\pi} e^{(\mathrm{erf}^{-1}(2x-1))^2} = \sqrt{2\pi} e^{\frac{(\Phi^{-1}(x))^2}{2}}$$

Two main statistics of the distribution are the mean and variance which can be obtained from the moments of the Vasicek distribution as shown in [37]

Moments of the Vasicek distribution

Proposition 4.4 *Let n be a positive integer and $\xi_1, \ldots, \xi_n$ i.i.d. standard normal. If X is Vasicek-distributed with parameters $p = \Phi(\alpha)$ and ρ, then*

$$\begin{aligned} E[X^n] &= E\left[\Phi\left(\frac{\alpha - \sqrt{\rho}y}{\sqrt{1-\rho}}\right)\right] = E\left[\prod_{i=1}^{n} P[\sqrt{\rho}Y + \sqrt{1-\rho}\xi_i \leq \alpha | Y]\right] \\ &= P[Y_1 \leq \alpha, \ldots, Y_n \leq \alpha] \end{aligned}$$

where $(Y_1, \ldots, Y_n)$ is a multi-variate normal vector with $E[Y_i] = 0$, $Var[Y_i] = 1$ and $corr[Y_i, Y_j] = \rho$, $i \neq j$.

Thus mean and variance expressions are:

$$E[X] = P[Y \leq \alpha] = p \tag{4.3.6}$$

$$var[X] = E[X^2] - E[X]^2 = P[Y_1 \leq \alpha, Y_2 \leq \alpha] - p^2 = \Phi_2(\alpha, \alpha, \rho) - p^2 \tag{4.3.7}$$

where (Y_1, Y_2) is a bivariate normal vector with $E[Y_i] = 0$, $Var[Y_i] = 1$ and $corr[Y_1, Y_2] = \rho$, $i = 1, 2$ and $\Phi_2(\cdot, \cdot, \rho)$ is the bivariate cumulative normal distribution function with zero mean and variance equal to the correlation ρ.

Finally, the quantile function or inverse cumulative distribution function can be easily obtained from Equation (4.3.3) as follows

$$\Phi\left(\frac{\sqrt{1-\rho}\,\Phi^{-1}(x)-\Phi^{-1}(p)}{\sqrt{\rho}}\right)=\beta \Longrightarrow \sqrt{1-\rho}\Phi^{-1}(x)-\Phi^{-1}(p)=\sqrt{\rho}\Phi^{-1}(\beta)$$

$$F^{-1}(\beta;p,\rho)=\Phi\left(\frac{\Phi^{-1}(p)+\sqrt{\rho}\Phi^{-1}(\beta)}{\sqrt{1-\rho}}\right) \tag{4.3.8}$$

which can be used for the calculation of Value-at-Risk.

4.3.2 General loss distribution

The Vasicek distribution implies a strong assumption; the systematic and idiosyncratic factors are normally and standardized distributed. This fact is far from common and in general we encounter factors with different distributions, as it was stated when we introduced the Generalized Vasicek model in Section 4.2. Schönbucher provided a general portfolio cumulative distribution function in [35] given by

$$F(x;\alpha,\rho)=P[X\leq x]=1-G\left(\frac{\alpha}{\sqrt{\rho}}-\sqrt{\frac{1-\rho}{\rho}}H^{-1}(x)\right) \tag{4.3.9}$$

We can derive Equation (4.3.9) to obtain the general portfolio loss density function

$$f(x;\alpha,\rho)=\frac{\partial F(x;\alpha,\rho)}{\partial x}=\sqrt{\frac{1-\rho}{\rho}}G'\left(\frac{\alpha}{\sqrt{\rho}}-\sqrt{\frac{1-\rho}{\rho}}H^{-1}(x)\right)\frac{\partial}{\partial x}H^{-1}(x) \tag{4.3.10}$$

and the quantile function or inverse cumulative distribution function can be easily obtained from Equation (4.3.9)

$$F^{-1}(\beta;\alpha,\rho)=H\left(\frac{\alpha-\sqrt{\rho}G^{-1}(1-\beta)}{\sqrt{1-\rho}}\right) \tag{4.3.11}$$

Considering again the particular case where both systematic and idiosyncratic factors are normally distributed:

$$\alpha=\Phi^{-1}(p),\quad \Phi^{-1}(1-\beta)=-\Phi^{-1}(\beta) \tag{4.3.12}$$

and substituting in (4.3.11) we obtain Equation (4.3.8), the quantile function for the Vasicek distribution.

Now we proceed to generate explicit formulas for some of the special factors models studied in Section 4.2, we provide formulas for those special factors involving the Logistic and Exponentially modified Gaussian distribution.

Logistic distribution

The first special factor model is *Logistic-Normal* defined as $Y\sim Logistic(0,\frac{\sqrt{3}}{\pi})$ and $H\sim\mathcal{N}(0,1)$ $(L-N)$. The $L-N$ distribution function is

$$\begin{aligned}F(x;\alpha,\rho)&=1-\frac{1}{2}\left(1+\tanh\left(\frac{\pi}{2\sqrt{3\rho}}(\alpha-\sqrt{1-\rho}\Phi^{-1}(x))\right)\right)\\&=\frac{1}{2}\left(1-\tanh\left(\frac{\pi}{2\sqrt{3\rho}}(\alpha-\sqrt{1-\rho}\Phi^{-1}(x))\right)\right)\end{aligned} \tag{4.3.13}$$

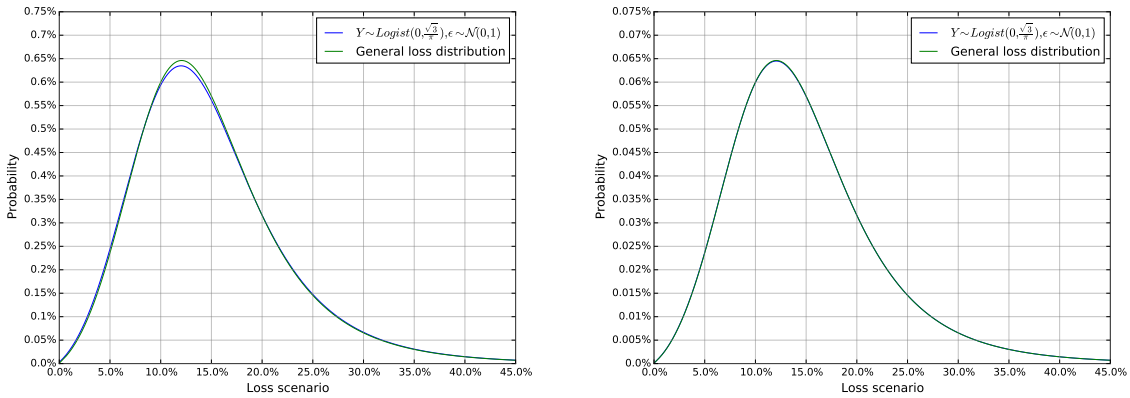
The $L-N$ density function is computed and simplified as follows

$$f(x;\alpha,\rho) = \sqrt{\frac{1-\rho}{\rho}}\frac{\pi}{4\sqrt{3}}\text{sech}^2\left(\frac{\pi}{2\sqrt{3}}\left(\frac{\alpha}{\sqrt{\rho}} - \sqrt{\frac{1-\rho}{\rho}}\Phi^{-1}(x)\right)\right)\sqrt{2\pi}e^{\frac{(\Phi^{-1}(x))^2}{2}}$$
$$= \sqrt{\frac{(1-\rho)\pi^3}{24\rho}}\text{sech}^2\left(\frac{\pi}{2\sqrt{3\rho}}(\alpha - \sqrt{1-\rho}\Phi^{-1}(x))\right)e^{\frac{(\Phi^{-1}(x))^2}{2}} \tag{4.3.14}$$

Finally, the $L-N$ quantile function

$$F^{-1}(\beta;\alpha,\rho) = \Phi\left(\frac{\alpha - \sqrt{\rho}\ln\left(\frac{1-\beta}{\beta}\right)\frac{\sqrt{3}}{\pi}}{\sqrt{1-\rho}}\right) \tag{4.3.15}$$

To compute the threshold α for this model and the subsequent ones, we can use one of the methods introduced in Subsection 4.2.1. To compare results we compute the same portfolio by means of the Generalized Vasicek model for large homogeneous portfolios of 1000, 5000 and 10000 assets respectively, Figure 4.6 shows how both methods produce almost imperceptible differences as the number of assets increases.



(a) *Assets=1000.* (b) *Assets=10000.*

Figure 4.6: General Loss Distribution vs. Generalized Vasicek Model: $Y \sim Logistic(0, \sqrt{3}/\pi)$ and $H \sim \mathcal{N}(0,1)$

Model	**N**	**Mean**	**Stdev**	$\boldsymbol{VaR_{0.999}}$
GLD	1000	0.1499999994	0.0764899844	0.570561
GVM(DE)	1000	0.1499999998	0.0772810889	0.572771
GLD	5000	0.1500000000	0.0764899834	0.570561
GVM(DE)	5000	0.1500000000	0.0766488578	0.571003
GLD	10000	0.1500000000	0.0764899834	0.570561
GVM(DE)	10000	0.1500000000	0.0765694618	0.570782

Table 4.3.1: Main metrics. General Loss Distribution vs. Generalized Vasicek Model: $Y \sim Logistic(0, \sqrt{3}/\pi)$ and $H \sim \mathcal{N}(0,1)$.

The following special factor model is the *Logistic-Logistic* defined as $Y \sim Logistic(0, \frac{\sqrt{3}}{\pi})$ and $H \sim Logistic(0, \frac{\sqrt{3}}{\pi})$ $(L-N)$. For this model we need to compute the derivative of the quantile function of Logistic distribution

$$F(x;\mu,s) = \frac{1}{1+e^{-\frac{x-\mu}{s}}} \Longrightarrow F^{-1}(p;\mu,s) = -\ln\left(\frac{1}{p}-1\right)s - \mu = \ln\left(\frac{p}{1-p}\right)s + \mu \tag{4.3.16}$$

$$\frac{\partial}{\partial p}F^{-1}(p;s) = \frac{s}{p(1-p)} \tag{4.3.17}$$

The $L-L$ distribution function is given by

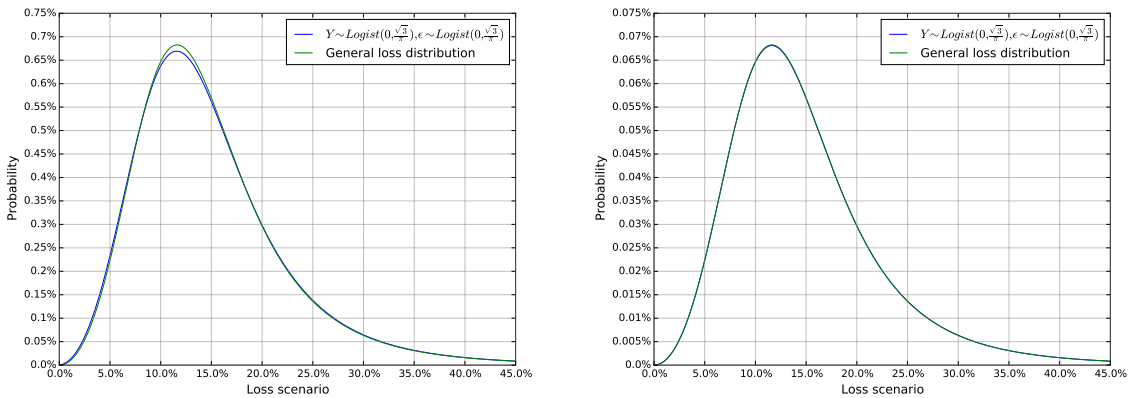
$$\begin{aligned} F(x;\alpha,\rho) &= 1 - \frac{1}{2}\left(1 + \tanh\left(\frac{\pi}{2\sqrt{3\rho}}(\alpha - \sqrt{1-\rho}\ln\left(\frac{x}{1-x}\right)\frac{\sqrt{3}}{\pi})\right)\right) \\ &= \frac{1}{2}\left(1 - \tanh\left(\frac{\alpha\pi}{2\sqrt{3\rho}} - \frac{\sqrt{1-\rho}\ln\left(\frac{x}{1-x}\right)}{2\sqrt{\rho}}\right)\right) \end{aligned} \tag{4.3.18}$$

and its derivative leads to the $L-L$ density function

$$\begin{aligned} f(x;\alpha) &= \sqrt{\frac{1-\rho}{\rho}}\frac{\pi}{4\sqrt{3}}\text{sech}^2\left(\frac{\pi}{2\sqrt{3}}\left(\frac{\alpha}{\sqrt{\rho}} - \sqrt{\frac{1-\rho}{\rho}}\ln\left(\frac{x}{1-x}\right)\frac{\sqrt{3}}{\pi}\right)\right)\frac{\sqrt{3}}{\pi(1-x)x} \\ &= \sqrt{\frac{(1-\rho)}{16\rho}}\text{sech}^2\left(\frac{\alpha\pi}{2\sqrt{3\rho}} - \frac{\sqrt{1-\rho}\ln\left(\frac{x}{1-x}\right)}{2\sqrt{\rho}}\right)\frac{1}{(1-x)x} \end{aligned} \tag{4.3.19}$$

Finally, we applied the derivative of the quantile function of Logistic distribution to obtain the $L-L$ quantile function

$$F^{-1}(\beta;\alpha,\rho) = \frac{1}{2}\left(1 + \tanh\left(\frac{\pi}{2\sqrt{3}}\frac{\alpha - \sqrt{\rho}\ln\left(\frac{1-\beta}{\beta}\right)\frac{\sqrt{3}}{\pi}}{\sqrt{1-\rho}}\right)\right) \tag{4.3.20}$$



(a) *Assets=1000.* (b) *Assets=10000.*

Figure 4.7: General Loss Distribution vs. Generalized Vasicek Model: $Y \sim Logistic(0, \sqrt{3}/\pi)$ and $H \sim Logistic(0, \sqrt{3}/\pi)$.

Model	**N**	**Mean**	**Stdev**	$\boldsymbol{VaR_{0.999}}$
GLD	1000	0.1500000000	0.0777958665	0.610054
GVM(DE)	1000	0.1500000000	0.0785725438	0.612051
GLD	5000	0.1500000000	0.0777958665	0.610054
GVM(DE)	5000	0.1500000000	0.0785725438	0.612051
GLD	10000	0.1500000000	0.0777958665	0.610054
GVM(DE)	10000	0.1500000000	0.0778738828	0.610253

Table 4.3.2: Main metrics. General Loss Distribution vs. Generalized Vasicek Model: $Y \sim Logistic(0, \sqrt{3}/\pi)$ and $H \sim Logistic(0, \sqrt{3}/\pi)$.

Exponentially Modified Gaussian (EMG) distribution

Similarly, we can generate explicit formulas for the special factors including the EMG distribution. However, these formulas might be apparently more involved and they may require some extra care in order to compute them accurately. Let us start with the *EMG-Normal* factor model defined as $Y \sim EMG(\mu = -0.95)$ and $H \sim \mathcal{N}(0,1)$. The $E-N$ distribution function is

$$F(x;\alpha,\rho,\mu) = 1 - \Phi\left(1 - \frac{t}{\mu}, 0, -\frac{\sqrt{1-\mu^2}}{\mu}\right) + e^{\frac{t}{\mu}-1+\frac{1-\mu^2}{2\mu^2}+\ln\left(\Phi\left(1-\frac{t}{\mu},\frac{1-\mu^2}{\mu^2},-\frac{\sqrt{1-\mu^2}}{\mu}\right)\right)} \tag{4.3.21}$$

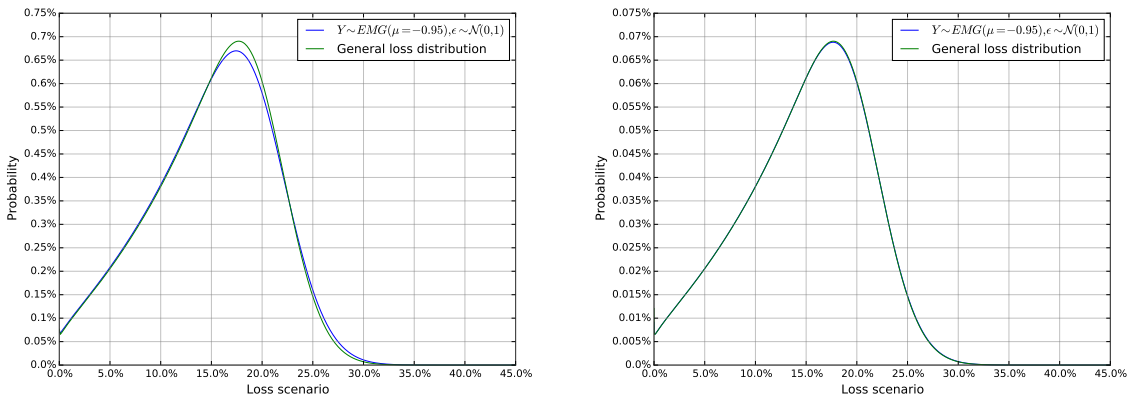
and its derivative leads to the $E-N$ density function

$$\begin{aligned} f(x;\alpha,\rho,\mu) &= \sqrt{\frac{1-\rho}{\rho}} f_{EMG}\left(t;\mu,\sqrt{1-\mu^2},-\frac{1}{\mu}\right)\sqrt{2\pi}e^{\frac{(\Phi^{-1}(x))^2}{2}} \\ &= -\sqrt{\frac{2\pi(1-\rho)}{\rho}}\frac{e}{2\mu}^{-\frac{1}{2\mu}\left(3\mu-\frac{1}{\mu}-2t+\frac{(\Phi^{-1}(x))^2}{2}\right)}\operatorname{erfc}\left(\frac{2\mu-\frac{1}{\mu}-t}{\sqrt{2(1-\mu^2)}}\right) \end{aligned} \tag{4.3.22}$$

where $t = \frac{\alpha}{\sqrt{\rho}} - \sqrt{\frac{1-\rho}{\rho}}\Phi^{-1}(x)$. Last the $E-N$ quantile function is given by

$$F^{-1}(\beta;\alpha,\rho) = \Phi\left(\frac{\alpha - \sqrt{\rho}F^{-1}_{EMG}(1-\beta;\mu,\sigma,\lambda)}{\sqrt{1-\rho}}\right) \tag{4.3.23}$$

where $\sigma = \sqrt{1-\mu^2}$ and $\lambda = -1/\mu$ and $F^{-1}_{emg}(\cdot)$ is the quantile function or inverse cumulative distribution of the EMG distribution. Section 2.2 analyses different methods for obtaining the β-quantile numerically.



(a) *Assets=1000.* (b) *Assets=10000.*

Figure 4.8: Large Homogeneous Portfolio.

Model	N	Mean	Stdev	$VaR_{0.999}$
GLD	1000	0.1499999408	0.0593468678	0.298122
GVM(DE)	1000	0.1500000000	0.0603822069	0.305975
GLD	5000	0.1499999969	0.0593467259	0.298122
GVM(DE)	5000	0.1500000000	0.0595552562	0.299734
GLD	10000	0.1499999991	0.0593467203	0.298122
GVM(DE)	10000	0.1500000000	0.0594510785	0.298930

Table 4.3.3: General Loss Distribution vs. Generalized Vasicek Model: $Y \sim EMG(\mu = -0.95)$ and $H \sim \mathcal{N}(0,1)$.

The fourth and last special factor presented is the *EMG-EMG* factor model defined as $Y \sim EMG(\mu_1 = -0.95)$ and $H \sim EMG(\mu_2 = -0.95)$ $(E-E)$. The $E-E$ distribution function is

$$F(x;\alpha,\rho,\mu_1,\mu_2) = 1 - F_{EMG}\left(\frac{\alpha}{\sqrt{\rho}} - \sqrt{\frac{1-\rho}{\rho}}F^{-1}_{EMG}\Big(x;\mu_2,\sigma_2,\lambda_2\Big);\mu_1,\sigma_1,\lambda_1\right) \tag{4.3.24}$$

the $E-E$ density function is given by

$$f(x;\alpha,\rho,\mu_1,\mu_2) = \sqrt{\frac{1-\rho}{\rho}}f_{EMG}\left(\frac{\alpha}{\sqrt{\rho}} - \sqrt{\frac{1-\rho}{\rho}}F^{-1}_{EMG}\Big(x;\mu_2,\sigma_2,\lambda_2\Big);\mu_1,\sigma_1,\lambda_1\right) \times \frac{\partial}{\partial x}F^{-1}_{EMG}(x) \tag{4.3.25}$$

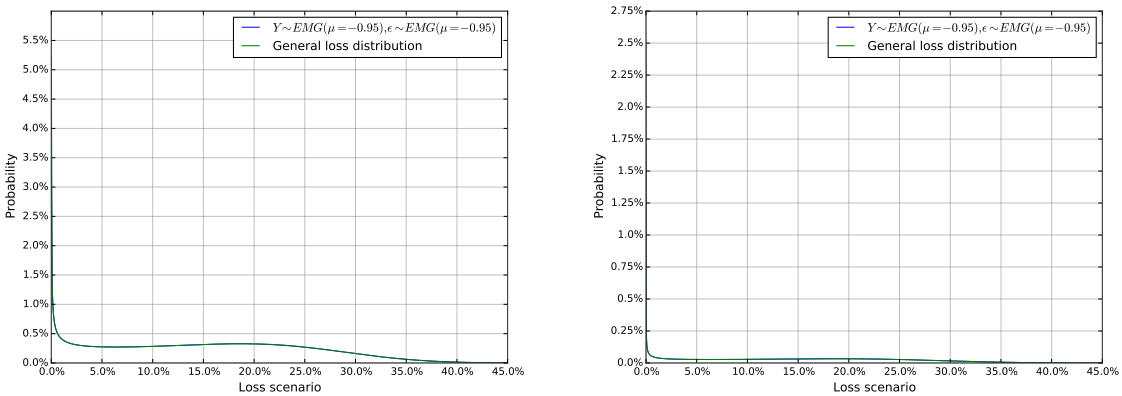
Given the following identity [6]

$$\frac{\partial}{\partial x}F^{-1}(x) = \frac{1}{F'(F^{-1}(x))} \tag{4.3.26}$$

we can avoid numerical differentiation methods, so that $\frac{\partial}{\partial x}F^{-1}_{EMG}(x) = \frac{1}{f_{EMG}(F^{-1}_{EMG}(x))}$

$$F^{-1}(\beta;\alpha,\rho,\mu_1,\mu_2) = F_{EMG}\left(\frac{\alpha - \sqrt{\rho}F^{-1}_{EMG}(1-\beta;\mu_1,\sigma_1,\lambda_1)}{\sqrt{1-\rho}};\mu_2,\sigma_2,\lambda_2\right) \tag{4.3.27}$$

where $\sigma = \sqrt{1-\mu^2}$ and $\lambda = -1/\mu$



(a) *Assets=1000.* (b) *Assets=10000.*

Figure 4.9: General Loss Distribution vs. Generalized Vasicek Model: $Y \sim EMG(\mu = -0.95)$ and $H \sim EMG(\mu = -0.95)$.

Model	**N**	**Mean**	**Stdev**	$\boldsymbol{VaR_{0.999}}$
GLD	1000	0.1499968461	0.1001417851	0.425138
GVM(DE)	1000	0.1500000000	0.1007219139	0.430791
GLD	5000	0.1499996361	0.1001376081	0.425138
GVM(DE)	5000	0.1500000000	0.1002543061	0.426282
GLD	10000	0.1499998536	0.1001372823	0.425138
GVM(DE)	10000	0.1500000000	0.1001957017	0.425711

Table 4.3.4: Main metrics. General Loss Distribution vs. Generalized Vasicek Model: $Y \sim EMG(\mu = -0.95)$ and $H \sim EMG(\mu = -0.95)$.

[6] $y = F^{-1}(x) \Leftrightarrow F(y) = x$, so $F'(y) = \frac{\partial x}{\partial y} \Leftrightarrow \frac{\partial y}{\partial x} = \frac{1}{F'(y)} = \frac{1}{F'(F^{-1}(x))}$

4.4 The Fourier Transform method

The next methodology is fundamentally different from those that have been studied in previous Sections. The Fourier Transform Method (FTM) is a methodology developed by Moody's [11] based on the numerical transform inversion of the portfolio, accelerated by means of Fast Fourier Transform (FTT) which aims to improve the speed, accuracy and adaptability of the Monte Carlo simulations. The FTM is particularly useful to analyse portfolios with asset heterogeneity in terms of rating/credit risk, size or maturity and also for portfolios with a limited number of assets. A particular feature of interest is its adaptability to incorporate stochastic loss given default in the model, some examples are shown in Subsection 4.4.2, where we provide an improved algorithm to compute the characteristic function of the Beta distribution, which employs the enhancements described in Subsection 2.4.1 for computing the confluent hypergeometric function.

In order to facilitate the comprehension of the Fourier transform method, let us first remind some of its basic properties.

Fourier transform basics

The characteristic function of a continuous random variable X, with density function $f(x)$, is a complex-valued function defined as

$$\varphi_X(t) := E[e^{itX}] = \int_{-\infty}^{\infty} e^{itx} f(x)\, dx = \int_{-\infty}^{\infty} cos(tx) f(x)\, dx + i \int_{-\infty}^{\infty} sin(tx) f(x)\, dx \tag{4.4.1}$$

Now, supposing $|\varphi_X(t)|$ is integrable, the density function of the random variable X, called the Inverse Fourier transform is given by

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-itx} \varphi_X(t)\, dt \tag{4.4.2}$$

The Fourier Transform method uses the following two properties of Fourier transforms

1. The Fourier transform of the sum of independent random variables is the product of the transforms, therefore given two independent random variables X and Y:

$$\varphi_{X+Y}(t) = \varphi_X(t) \cdot \varphi_Y(t) \tag{4.4.3}$$

2. Conjugation: If $h(x) = \overline{f(x)}$, then $\hat{h}(t) = \overline{\hat{f}(-t)}$, where $\hat{h}(t)$ and $\hat{f}(t)$ represent the Fourier transform of $h(x)$ and $f(x)$, respectively. In particular, if $f(x)$ is real, then the *reality condition* applies and $\hat{f}(-t) = \overline{\hat{f}(t)}$, hence $\hat{f}(t)$ is a Hermitian function [7].

4.4.1 The model framework

Consider a credit portfolio consisting of N obligors, where any obligor $n = 1, \ldots, N$ can be characterized by three parameters: the exposure at default (EAD) denoted by E_n, loss given default (LGD) denoted by Λ_n and the probability of default PD_n. Obligor n is subject to default after a fixed time horizon, usually 1 year, and the default can be modelled as a Bernoulli random variable D_n, such that

$$D_n = \begin{cases} 1 & \text{if obligor } n \text{ is in default with probability } PD_n \\ 0 & \text{if obligor } n \text{ is not in default with probability } 1 - PD_n \end{cases} \tag{4.4.4}$$

[7] A **Hermitian function** is a complex function with the property that its complex conjugate is equal to the original function with the variable changed in sign.

The loss incurred due to default of obligor n is given by

$$L_n = E_n \cdot \Lambda_n \cdot D_n = w_n \cdot D_n \tag{4.4.5}$$

where $w_n = E_n \cdot \Lambda_n$ is the effective exposure of obligor n. Then the portfolio loss is defined as

$$L = \sum_{n=1}^{N} L_n \tag{4.4.6}$$

Let us a consider a deterministic loss given default of 100% so $\Lambda_n = 1$ for $n = 1, \ldots, N$. For a given state of the economy $Y = y$, the conditional Fourier transform of the portfolio default distribution $(f_L(x), 0 < x < 1)$ is defined as

$$\hat{f}_{L|Y=y}(t) = E[e^{-itL} \,|\, Y = y] = E[e^{-it\sum_{n=1}^{N} w_n D_n} \,|\, Y = y] \tag{4.4.7}$$

Using the conditional independence given Y of the default indicators D_n, we can apply

$$\hat{f}_{L|Y=y}(t) = \prod_{n=1}^{N} E[e^{-itw_n D_n} \,|\, Y = y] \tag{4.4.8}$$

Given that $P[D_n = 1 \,|\, Y = y] = p_n(y)$ and $P[D_n = 0 \,|\, Y = y] = 1 - p_n(y)$, the conditional Fourier transform of the portfolio default distribution is given by

$$\hat{f}_{L|Y=y}(t) = \prod_{n=1}^{N} [1 + p_n(y) + p_n(y)e^{-itw_n}] \tag{4.4.9}$$

and the unconditional Fourier transform is obtained by considering the density function of the factor Y, $\phi(y)$

$$\hat{f}_L(t) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 + p_n(y) + p_n(y)e^{-itw_n}]\phi(y)\, dy \tag{4.4.10}$$

where

$$p_n(y) = \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}y}{\sqrt{1-\rho_n}}\right) \quad \text{and} \quad \phi(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$$

In spite of we used different statistical distributions to represent the systematic and idiosyncratic factor throughout this Chapter, we restrict ourselves to the normal case for this methodology. The reason is based on the fact that we are interested in demonstrating the capabilities of the method rather than providing a bunch of different models. However, this methodology can be easily extended using some of the approaches presented in previous Sections of this Chapter.

The portfolio's aggregate default/loss distribution or portfolio Fourier transform $\hat{f}_{L|Y=y}(t)$ can be computed numerically for any given value of t. A detailed explanation about our implementation in C++ is presented in Subsection 4.4.3.

4.4.2 Stochastic Loss Given Default

So far we have been considering a portfolio with deterministic loss given default. We now extend the explored method in order to consider stochastic loss given default. Assuming D_n and Λ_n are conditionally independent given $Y = y$.

$$\hat{f}_{L|Y=y}(t) = \prod_{n=1}^{N} E[e^{-itE_n D_n \Lambda_n} \,|\, Y = y] \tag{4.4.11}$$

$$E[e^{-itE_n D_n \Lambda_n} \,|\, Y = y] = 1 - p_n(y) + p_n(y)\hat{f}_{\Lambda_n}(t, En; Y) \tag{4.4.12}$$

where

$$\hat{f}_{\Lambda_n}(t, En; Y) = E[e^{-itE_n \cdot 1 \cdot \Lambda_n} \,|\, Y = y] \tag{4.4.13}$$

Hence, the conditional portfolio Fourier Transform is given by

$$\hat{f}_{L|Y=y}(t) = \prod_{n=1}^{N} [1 - p_n(y) + p_n(y)\hat{f}_{\Lambda_n}(t, En; Y)] \tag{4.4.14}$$

and therefore the unconditional portfolio Fourier transform is expressed as

$$\hat{f}_L(t) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y)\hat{f}_{\Lambda_n}(t, En; Y)]\phi(y)\, dy \tag{4.4.15}$$

Now we introduce two of the possible modelling assumptions when considering LGDs: The deterministic case and the stochastic case, from which we show four distribution assumptions. As shown below the stochastic LGD can lead to expressions of $\hat{f}_{\Lambda_n}(t, En; Y)$ that might be difficult to evaluate.

Deterministic LGD

If LGDs are considered deterministic, values of Λ_n for $n = 1, \ldots, N$ are directly provided to the model, therefore the portfolio Fourier transform is given by

$$\hat{f}_L(t) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y)e^{-itE_n \Lambda_n}]\phi(y)\, dy \tag{4.4.16}$$

since,

$$\hat{f}_{\Lambda_n}(t, E_n; Y) = E[e^{-itEn\Lambda_n}] = e^{-itE_n \Lambda_n} \tag{4.4.17}$$

Stochastic LGD

For the following four cases we consider that Λ_n is a random variable such that $\Lambda_n \in (0, 1)$ with mean μ_n and standard deviation σ_n. The LGDs are independent from the systematic factor Y and from L, therefore $\hat{f}_{\Lambda_n}(t, E_n; Y) = \hat{f}_{\Lambda_n}(t, E_n)$.

Case 1: Normal distribution Applying the Fourier transform definition of the Normal distribution:

$$\hat{f}_{\Lambda_n}(t, E_n) = e^{-itE_n \mu_n} e^{\frac{\sigma_n^2}{2} t^2 E_n^2} \tag{4.4.18}$$

the portfolio Fourier transform is given by

$$\hat{f}_L(t) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y)e^{-itE_n \mu_n} e^{\frac{\sigma_n^2}{2} t^2 E_n^2}]\phi(y)\, dy \tag{4.4.19}$$

Case 2: Triangular distribution

$$\hat{f}_{\Lambda_n}(t, E_n) = e^{-it\mu_n} \operatorname{sinc}\left(\frac{\sqrt{6}\sigma_n t E_n}{2}\right)^2 \tag{4.4.20}$$

where $\operatorname{sinc}(x)$ is the unnormalized sinc function defined as

$$\operatorname{sinc}(x) = \frac{\sin(x)}{x}, \quad \text{for } x \neq 0 \text{ and } \operatorname{sc}(0) = 1$$

Case 3: Gamma distribution

$$\hat{f}_{\Lambda_n}(t, E_n) = (1 + i\beta_n t E_n)^{-\alpha_n} \tag{4.4.21}$$

Due to the mean and variance of the Gamma distribution are defined as

$$E[X] = \mu = \alpha\beta \quad \text{and} \quad Var[X] = \sigma^2 = \alpha\beta^2 \tag{4.4.22}$$

we can parametrize α_n and β_n as follows

$$\alpha_n = \frac{\mu_n^2}{\sigma_n^2} \quad \text{and} \quad \beta_n = \frac{\sigma_n^2}{\mu_n}$$

Case 4: Beta distribution It has been empirically evidenced that the uncertainty regarding the LGD rates of defaulted obligors can be modelled as beta random variable independent from each obligor, therefore a beta distribution for LGD distribution seems justified. Moody's in [11] uses an approximation based on the Fourier transform of the Beta discretized distribution.

$$\hat{f}_{\Lambda_n}(t) = \frac{1}{B(n,p)M} \sum_{k=0}^{M-1} \left(\frac{k}{M}\right)^{n-1} \left(1 - \frac{k}{M}\right)^{p-1} e^{-itE_n \frac{k}{M}} \tag{4.4.23}$$

where M is the number of discretization points of the LGD distribution and

$$B(n,p) = \frac{\Gamma(n)\Gamma(p)}{\Gamma(n+p)}, \quad n = \mu\left(\frac{(1-\mu)\mu}{\sigma^2} - 1\right) \quad \text{and} \quad p = (1-\mu)\frac{n}{\mu} \tag{4.4.24}$$

This approximation is very expensive to compute. Instead we use the fact that the characteristic function of a beta distributed random variable with parameters α and β can be expressed as follows

$$\varphi_X(t) = \sum_{k=0}^{\infty} \frac{(it)^k B(\alpha+k, \beta)}{k! B(\alpha, \beta)} = 1 + \sum_{k=1}^{\infty} \frac{(it)^k}{k!} \prod_{n=0}^{k-1} \frac{\alpha+n}{\alpha+\beta+n} = {}_1F_1(\alpha; \alpha+\beta; it) \tag{4.4.25}$$

The last function ${}_1F_1(a; b; z)$ is the confluent hypergeometric function of the first kind, which is described in detail in Section 2.4. Given the LGD parameters μ and β we can parametrize α and β using mean and variance of the beta distribution,

$$E[X] = \frac{\alpha}{\alpha+\beta} \quad \text{and} \quad Var[X] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{4.4.26}$$

$$u = \frac{1-\mu}{\mu}, \quad \alpha = \frac{u}{(u+1)^3\sigma^2} - \frac{1}{1+u}, \quad \text{and} \quad \beta = \alpha u \tag{4.4.27}$$

Thus, the portfolio Fourier transform with beta distributed LGD is given by

$$\hat{f}_L(t) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y) {}_1F_1(\alpha_n; \alpha_n + \beta_n; -itE_n)] \phi(y)\, dy \tag{4.4.28}$$

In order to compute the characteristic function of the beta distribution we have designed the following algorithm using the methods introduced in Section 2.4.1 for computing the confluent hypergeometric function.

Algorithm 4 HypBeta: Characteristic function of the Beta distribution

```
function HypBeta(a, b, z, nodes = 64, tol = 1e−15, maxiter = 1000)
    a, b ∈ ℝ and x, z ∈ ℂ
    if a, b ≲ |z| then
        if a, b < 50 then
            x = HypTaylor(a, b, z, tol, maxiter)          ▷ Taylor series method
        else
            x = HypGJ(a, b, z, nodes)                     ▷ Gauss-Jacobi method
        end if
    else
        x = HypSD(a, b, z, nodes)                         ▷ Numerical Steepest Descent method
    end if
    return x
end function
```

A more sophisticated algorithm computes the integral representation of the confluent hypergeometric function for several nodes and returns an estimate of the relative error for two different quadratures. This algorithm is applied in the Gauss-Jacobi method and the numerical steepest descent method.

Algorithm 5 HypIntegral: method for integral representation ${}_1F_1(a; b; z)$

```
function HypIntegral(a, b, z, nodes = [...], relerr = 1e−15)
    a, b ∈ ℝ and x, z ∈ ℂ
    nodes: array of nodes, dim(nodes)> 1                  ▷ i.e. [20, 32, 64, ...]
    s_0 = Hyp1f1(a, b, z, 10)                             ▷ for a, b not too large 10 nodes might suffice
    for j := 1 to dim(nodes) do
        s_1 = Hyp1f1(a, b, z, nodes[j])
        relerr_s = |s_1 − s_0|/s_0
        if relerr_s < relerr then
            break
        else
            s_0 = s_1
        end if
    end for
    return s_1, relerr_s
end function
```

4.4.3 Implementation

In this Subsection we present an implementation of the FTM with beta distributed LGD. This implementation has two steps: **generation** of portfolio Fourier transform and **inversion** to obtain the portfolio loss distribution. The first step contains two main routines: computes the characteristic function and computes the portfolio Fourier transform via complex Gauss-Hermite quadrature.

The inversion step of the portfolio is performed by using Fast Fourier Transform (FFT). We implemented a FFT algorithm that requires two arrays with the real and imaginary part.

The inversion algorithm is not straightforward because of the number of settings and tunings. A detailed explanation can be found in [11] Appendix 1. The basic idea is listed below

1. computation of $\hat{f}(t)$ generates a vector of complex numbers.
2. given a Fourier resolution M the inversion is performed using the Discrete Inverse Fourier Transform (DIFT)

$$[f_0, f_1, \ldots, f_{M-1}] = \text{DIFT}([\hat{f}(t_0), \ldots, \hat{f}(t_{M/2}), \overline{\hat{f}(t_{M(2-1)})}, \ldots, \overline{\hat{f}(t_1)}]) \tag{4.4.29}$$

where

$$\text{DIFT}([\hat{f}(t_0), \ldots, \hat{f}(t_{M/2})]) = \frac{1}{M} \sum_{k=0}^{M/2} \hat{f}(t_k) e^{2\pi i k m/M},\ m = 0, \ldots, M/2 \tag{4.4.30}$$

Figure 4.10 illustrates the computation steps required for the Fast Fourier Transform.

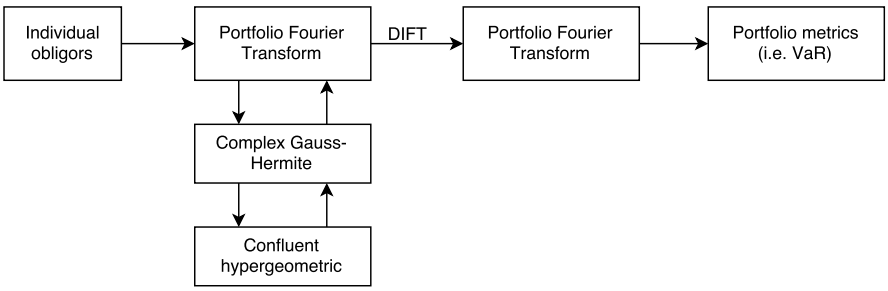


Figure 4.10: Steps of the FTM computation with Beta distributed LGDs.

As stated, this method has been implemented in C++, to do so we use a `functor` for the characteristic function of the beta distribution and the conditional portfolio Fourier transform. A `functor` looks like a `class` which defines the `operator()`, but has good properties like the ease to pass a `functor` as argument to other functions so it can be used as a function pointer (`functors` are potentially more efficient) and it can hold state. Three of the main routines are shown below, the complete code can be found in the repository in Chapter 1.

Listing 4.2: CF Beta LGD - C++ code

```
struct CFBetaLGD
{
  // Characteristic function \hat{f}_{LGD}(t, E) with Beta Loss Given Default
      (alpha, beta)

  const double t, aa, bb;
  const int N;
  const std::vector<double> E, pd, rho;

  CFBetaLGD(const double tt, const std::vector<double> default_probability,
      const std::vector<double> correlation,
    const std::vector<double> &exposures, const double a, const double b,
        const int assets):
  t(tt), pd(default_probability), rho(correlation), N(assets), E(exposures),
      aa(a), bb(b){};

  std::complex<double> operator () (const double y)
  {
    std::complex<double> prod=1;
    for (int n = 0; n < N; n++) {
```

```
      ObligorDP_GH obligor(rho[n], pd[n]);
      double dp = obligor.operator()(y);
      std::complex<double> beta_c = hyper_1F1(aa, aa + bb, std::complex<double
          >(0, -t*E[n]));
      prod *=  1 - dp + dp*beta_c;
    }
    return prod;
  }
};
```

The `functor CFBetaLGD` is initialized and passed as argument to `GaussHermiteComplex`, which implements the Gauss-Hermite quadrature allowing complex numbers.

Listing 4.3: Portfolio Complex integration quadrature - C++ code

```
inline std::complex<double> portfolioBetaLGD (double t, const std::vector<
    double> p_n,
  const std::vector<double> rho, const std::vector<double> &exposures, const
      double alpha, const double beta)
{
  // Compute Unconditional CF using 40-nodes Gauss-Hermite complex quadrature.
  const int N = exposures.size();
  CFBetaLGD cf (t, p_n, rho, exposures, alpha, beta, N);
  const double sq_pi = 1/sqrt(pi);

  std::complex<double> u = sq_pi * GaussHermiteComplex(cf, 40);
  return u;
}
```

Finally, the following `inline` function is the one that should be called from a `main()` function to compute the portfolio loss distribution. The inputs required are: array of default probabilities, correlations and exposures. For the sake of simplicity, we consider a common α and β shared for all obligors. `Vmax` and `delta_x` are specific setting of the algorithm, see [11] Appendix 1.

Listing 4.4: Inline function to compute the portfolio loss distribution - Code C++

```
inline std::vector<double> fourier_default_portfolio_BetaLGD(std::vector<
    double> & prob,
  std::vector<double> &corr, std::vector<double> & sizes, double alpha, double
      beta, double Vmax,
  double delta_x, double total_amount=NULL)
{
  const int N = prob.size();
  double total_portfolio;

  if (total_amount == NULL)
  {
    total_portfolio= 0.0;
    for (int i = 0; i < N; i++){total_portfolio += sizes[i];}
  }
  else
    total_portfolio = total_amount;

  // size - exposure (%)
  std::transform(sizes.begin(), sizes.end(), sizes.begin(), [&] (double p) {
      return p/total_portfolio;});

  const double T = 2*pi/delta_x;
  const int Nmax = floor(Vmax/delta_x)+1;
  const int pmin = floor(log(Nmax-1)/log(2))+1;
  const int ResF = pow(2, pmin);
```

```
    std::vector<double>f_real(ResF);
    std::vector<double>f_imag(ResF);

    double f_infinity = portfolioLGD_infinity(prob, corr, sizes);
    for (int i = 0; i <ResF ; i++) {
      std::complex<double> ft = portfolioBetaLGD(i*T/ResF,prob, corr, sizes,
          alpha, beta);
      f_real[i] = real(ft);
      f_imag[i] = imag(ft);
    }

    // perform the numerical transform inversion using FFT
    short result = FFT(-1, pmin , f_real, f_imag);
    f_real[0] += f_infinity;

    std::transform(f_real.begin(), f_real.end(), f_real.begin(), [&] (double p)
        {return p/ResF;});
    return f_real;
  }
```

4.4.4 Numerical examples

Several portfolios has been considered to test the Fourier Transform method with respect to other methods already seen throughout this thesis. The portfolio computation by Gauss-Hermite integration serve us as a benchmark for those portfolios without concentrations. For other cases a crude Monte Carlo is used.

Portfolio 1. This portfolio has $N = 100$ obligors with $p_n = 0.15$, $\rho_n = 0.2$ and $E_n = 1$ for $n = 1, \ldots, N$.

Portfolio 2. This portfolio has $N = 100$ obligors with $p_n = 0.05$, $\rho_n = 0.1$ and $E_n = 1$ for $n = 1, \ldots, N-2$ and $p_n = 0.15$ and $\rho_n = 0.05$ and $E_n = 20$ for $n = N-1, N$.

Portfolio 3. This portfolio has $N = 1001$ obligors with $p_n = 0.0033$, $\rho_n = 0.2$ and $E_n = 1$ for $n = 1, \ldots, N-1$ and $E_{1001} = 100$ as in [25].

Portfolio 4. This portfolio has $N = 100$ obligors with $p_n = 0.03$, $\rho_n = 0.04$ and $E_n = 1$ for $n = 1, \ldots, N$. LGD: $\mu = 0.55$, $\sigma = [0.1, 0.2, 0.3, 0.4]$ as in [11].

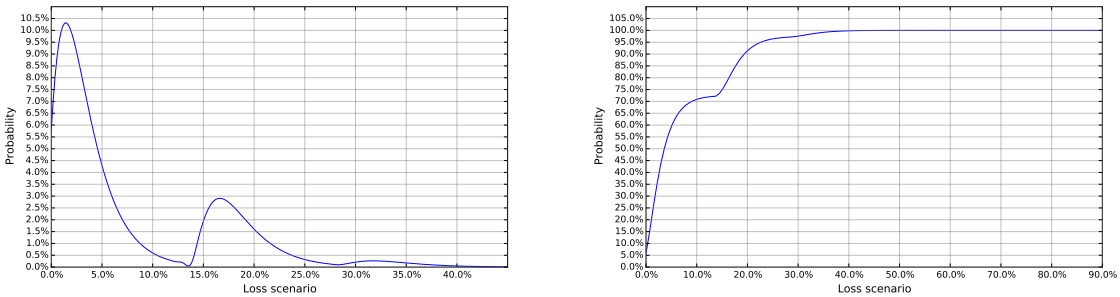
Portfolio 5. This portfolio has $N = 100$ obligors with $p_n = 0.03$, $\rho_n = 0.04$ and $E_n = 1$ for $n = 1, \ldots, N$. LGD: $\mu = 0.55$, $\sigma = 0.25$ as in [11].

For all portfolios E_n is normalized such that $\sum_{n=1}^{N} E_n = 1$. Portfolio 1 is an homogeneous portfolio used as a first benchmark due to we have various available methods for comparison. As shown, the FTM does not present an improvement in terms of accuracy and computation time, therefore this indicates that is not the right method for this type of portfolios.

Method	**Stdev**	**Stdev/mean**	$\boldsymbol{VaR_{0.999}}$	**Time(s)**
Gauss-Hermite 48	0.11493807949	0.76625386514	0.67448200839	0.041
Fourier Transform Method	0.11493808061	0.76625387072	0.67449802301	4
MC+MLHS 100 + tan() *transform*	0.11493807935	0.76625386441	0.67450167148	0.066
MC+MLHS 10000	0.11493228368	0.76622021808	0.67450339261	7
MC 10000	0.11477569151	0.76418386150	0.62348652726	26

Table 4.4.1: Main metrics of portfolio 1. Homogeneous portfolio.

Portfolio 2 is a well diversified portfolio where two big exposures represent the 29% of the total portfolio exposure. One might have noticed that the analytic approximation of the Vasicek model has not been included in the benchmarking. This is due to the analytic approximation of the Vasicek model can significantly underestimate risks in the presence of exposure concentrations, as when the portfolio is dominated by a few obligors as in this example.



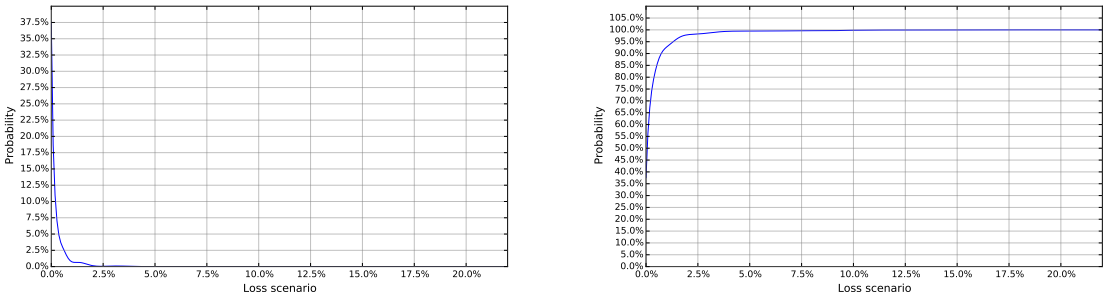
(a) *Portfolio loss distribution.* (b) *Cumulative portfolio loss distribution.*

Figure 4.11: Example of well diversified portfolio. Portfolio 2.

Method	**Mean**	**Stdev**	$\boldsymbol{VaR_{0.999}}$	**Time(s)**
Fourier Transform Method	0.07898550725	0.08672431283	0.44057496509	2

Table 4.4.2: Main metrics of portfolio 2.

Portfolio 3 is also a well diversified portfolio with a big exposure representing 9.09% of the total portfolio exposure. Portfolio 2 and portfolio 3 are examples of portfolios with exposure concentrations. As shown in Table 4.4.3, the computation time required increases significantly as we consider portfolios with more obligors. This is a problem, since portfolios of thousand or millions of obligors exist in the financial industry.



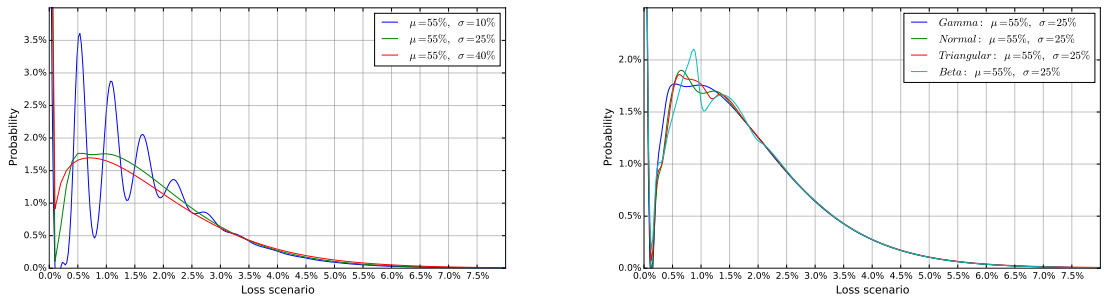
(a) *Portfolio loss distribution.* (b) *Cumulative portfolio loss distribution.*

Figure 4.12: Portfolio with exposure concentrations. Portfolio 3.

Method	**Mean**	**Stdev**	**VaR**	**Time(s)**
FTM (CL = 0.999)	0.00329855889	0.00836879739	0.1074 **(0.28%)**	59
FTM (CL = 0.9999)	0.00329855889	0.00836879739	0.1529 **(0.22%)**	59

Table 4.4.3: Main metrics of portfolio 3. Errors relative to Monte Carlo with 5 millions scenarios for the VaR are shown in parenthesis. CL := confidence level.

Finally, Figure 4.13 shows the portfolio loss distribution for Portfolio 4 and 5, which consider stochastic LGD. In portfolio 4 the LGDs follow a Gamma distribution with varying standard deviation to notice the effect of the parameter on the shape of the portfolio loss distribution. Portfolio 5 compares the portfolio loss distribution for different distributions of LGD given $\mu = 0.55$ and $\sigma = 0.25$. Porfolio 5 shows a similar shape for the Normal, Triangular and Gamma distribution whereas a prominent peak is appreciated for the Beta distribution.



(a) *Portfolio 4.* (b) *Portfolio 5.*

Figure 4.13: Portfolio loss distribution with stochastic LGD.

As previously stated, an important issue with the Fourier Transform method is the computational time required as the number of obligors increases. This issue becomes relevant when stochastic LGD is considered, for example the computation of the above portfolios took 25 minutes approximately. Therefore, some research has to be carried out in order to develop a methodology able to compute the portfolio loss distribution accurately in a reasonable computation time. A first straightforward approach could be the parallelization of the generation step along with the use of parallel FFT algorithms. Furthermore, some parts of the code can be precomputed avoiding repeated computations. Finally, note that the code presented was not optimized due to time constraints and therefore a considerable improvement can be accomplished with a more accurate implementation.

Chapter 5

The Haar Wavelet Approximation to computing VaR

Banks and financial institutions quantify credit risk at portfolio level by means of risk measures such as VaR or Expected Shortfall. Historically, the most widely used risk measure has been the VaR_α when $\alpha \to 1$, i.e. when considering high loss levels. As it is well known, VaR is not a coherent risk measure due to its undesirable properties such as lack of sub-additivity and the fact that is difficult to optimize when calculated using scenarios because of its non-convexity and discontinuities. Therefore, other risk measures such Expected Shortfall (ES) or conditional Value-at-Risk (CVaR) are used, since they satisfies the properties of a coherent risk measure. However, in this Chapter we restrict ourselves to the VaR to compare results with respect to other methods previously studied.

The Haar wavelet-based approximation (WA) is a new method for computing VaR by numerically inverting the Laplace transform of the CDF of the loss distribution, once it is approximated by a finite sum of Haar wavelets basis functions. The WA method unlike the FTM, does not compute the entire loss distribution, therefore it is less affected in terms of computational time by the increase of the number of obligors. On the other hand, both methods are capable of handling portfolios with exposure concentrations.

This Chapter follows the detailed description of the Haar wavelet-based approximation described in (J.J. Masdemont, L.Ortiz-Gracia [24, 25]). We provide details which were omitted in [24, 25] to facilitate the complete comprehension of all the steps required by this method. Furthermore, we introduce a more general method to compute the coefficients in the wavelet expansion approximation for non-normal density distribution functions. We compare this method with respect to a method using a complex Gauss-Hermite quadrature. Finally, we show how this methodology can be extended in order to consider stochastic LGD.

5.1 The model framework

The underlying model framework does not differ from the model framework described in Subsection 4.4.1. Let us briefly recall the definition of the portfolio loss

$$L = \sum_{n=1}^{N} E_n \cdot D_n \cdot \Lambda_n$$

As in the FTM, we use the Vasicek one-factor model with systematic and idiosyncratic factor following a standard normal distribution

$$p_n(y) = P[V_n < \alpha_n \,|\, Y = y] = \Phi\left(\frac{\alpha_n - \sqrt{\rho_n}y}{\sqrt{1-\rho_n}}\right) \quad \text{and} \quad \phi(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}$$

5.1.1 Haar wavelets approximation

Let F be the cumulative distribution function of L. Without loss of generality, we can assume $\sum_{n=1}^{N} E_n = 1$ and consider

$$F(x) = \begin{cases} \overline{F}(x) & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{5.1.1}$$

for a certain $\overline{F}$ defined in $[0,1]$. $\overline{F}$ can be approximated by a finite summation of scaling functions, Haar wavelets basis functions

$$\overline{F}(x) \approx \overline{F}_m(x) = \sum_{i=1}^{2^m-1} c_{m,k}\phi_{m,k}(x) \tag{5.1.2}$$

where $\phi_{m,k}(x) = 2^{m/2}\phi(2^m x - k)$ and

$$\phi(x) = \begin{cases} 1 & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.1.3}$$

Parameters m and k are the scale of the approximation and the translation parameter, respectively. As $m \to \infty$, the approximation converges in $L^2([0,1])$ by the theory of Multi-resolution analysis (MRA)

$$\lim_{m\to\infty} \overline{F}_m(x) = \overline{F}(x) \tag{5.1.4}$$

Now we need to compute the unconditional portfolio loss distribution. The procedure is similar to the one described in the FTM, but we consider the moment generating function (MGF) instead of the characteristic function (CF). The required steps follow below:

The MGF conditional to Y is given by

$$\begin{aligned} M_L(s;Y) &= E[e^{-s\cdot L} \,|\, Y = y] = E[e^{-s\sum_{n=1}^{N} E_n D_n \Lambda_n} \,|\, Y = y] \\ &= \prod_{n=1}^{N} E[e^{-sE_n D_n \Lambda_n} \,|\, Y = y] \end{aligned} \tag{5.1.5}$$

Taking into account the fact that $P[D_n = 1 \,|\, Y = y] = p_n(y)$ and $P[D_n = 0 \,|\, Y = y] = 1 - p_n(y)$,

$$\begin{aligned} E[e^{-sE_n D_n \Lambda_n} \,|\, Y = y] &= (1 - p_n(y))e^{-sE_n \cdot 0 \cdot \Lambda_n} + p_n(y)e^{-sE_n\Lambda_n} \\ &= 1 - p_n(y) + p_n(y)e^{-sE_n\Lambda_n} \end{aligned} \tag{5.1.6}$$

Thus the conditional MGF expression is simplified as follows

$$M_L(s;Y) = \prod_{n=1}^{N} [1 - p_n(y) + p_n(y)e^{-sE_n\Lambda_n}] \tag{5.1.7}$$

Assuming non-stochastic LGD, the unconditional MGF obtained by taking the expectation value of the conditional MGF is given by

$$M_L(s) = E[e^{-sL}] = E[M_L(s;Y)] = E\Big[\prod_{n=1}^{N}[1 - p_n(y) + p_n(y)e^{-sE_n\Lambda_n}]\Big]$$
$$= \int_{-\infty}^{\infty}\prod_{n=1}^{N}[1 - p_n(y) + p_n(y)e^{-sE_n\Lambda_n}]f(y)\,dy \tag{5.1.8}$$

If f is the probability density function of the loss function, then the unconditional MGF can be rewritten in terms of the Laplace transform of f

$$M_L(s) = E[e^{-sL}] = \mathcal{L}\{F'(x)\} = \int_0^{\infty} e^{-sx}F'(x)\,dx \tag{5.1.9}$$

and we can split the integral and integrate by parts as follows

$$M_L(s) = \int_0^{\infty} e^{-sx}F'(x)\,dx = \underbrace{\int_0^1 e^{-sx}\overline{F}'(x)\,dx}_{(a)} + \underbrace{\int_1^{\infty} e^{-sx}F'(x)\,dx}_{(b)} \tag{5.1.10}$$

Note in (b) that $F(x)$ for $x \geq 1$ has a constant value 1, then $F'(x) = 0$, therefore the second integral is cancelled. Integrating by parts (a)

$$M_L(s) = \int_0^1 e^{-sx}\overline{F}'(x)\,dx = e^{-sx}\overline{F}(x)\Big|_0^1 - \int_0^1 -se^{-sx}\overline{F}(x)\,dx = e^{-s} + s\int_0^1 e^{-sx}\overline{F}(x)\,dx$$
$$\approx e^{-s} + \int_0^1 e^{-sx}\sum_{i=1}^{2^m-1} c_{m,k}\phi_{m,k}(x)\,dx = e^{-s} + 2^{m/2}s\sum_{i=1}^{2^m-1} c_{m,k}\tilde{\phi}_{m,k}(x) \tag{5.1.11}$$

where

$$\tilde{\phi}_{m,k}(s) = \frac{1}{s}e^{-s(k/2^m)}(1 - e^{-s/2^m}) \tag{5.1.12}$$

is the Laplace transform of the basis function $\phi_{m,k}(x)$. One can notice that $\tilde{\phi}_{m,k}(s) = \tilde{\phi}_{m,0}(s)e^{-s/2^m}$ and making the change of variable $z = e^{-s/2^m}$

$$Q(z) \equiv \sum_{k=0}^{2^m-1} c_{m,k}z^k \approx \frac{M_L(-2^m\ln(z)) - z^{2^m}}{2^{m/2}(1-z)} \tag{5.1.13}$$

Since $Q(z)$ is not defined at $z = 0$ we approximate the value by taking the limit as $z \to 0$

$$\lim_{z\to 0} Q(z) = c_{m,0} = \frac{\int_{-\infty}^{\infty}\prod_{n=1}^{N}[1 - p_n(y)]f(y)\,dy}{2^{m/2}} \tag{5.1.14}$$

therefore,

$$\overline{Q}(x) = \begin{cases} Q(z) & \text{if } z \neq 0 \\ \lim_{z\to 0} Q(z) & \text{if } z = 0 \end{cases} \tag{5.1.15}$$

$\overline{Q}(z)$ is analytic inside the open unit disc of the complex plane. This avoids the singularity at $z = 0$. The coefficients $c_{m,k}$ can be obtained by means of Cauchy's integral formula.

$$c_{m,k} = \frac{1}{2\pi i}\oint_{\gamma}\frac{Q(z)}{z^{k+1}}\,dz, \quad z = 1,\ldots,2^m - 1 \quad (z \neq 0) \tag{5.1.16}$$

where γ denotes a circle of radius r, $0 < r < 1$ about the origin. Finally, we consider the change of variable $z = re^{iu}$ which yields

$$\begin{aligned} c_{m,k} &= \frac{1}{2\pi i}\int_0^{2\pi} \frac{Q(re^{iu})\,d(re^{iu})}{r^k re^{iu(k+1)}} = \frac{1}{2\pi i}\int_0^{2\pi} \frac{Q(re^{iu})re^{iu}i\,du}{r^k re^{iu(k+1)}} = \frac{1}{2\pi r^k}\int_0^{2\pi} \frac{Q(re^{iu})\,du}{e^{iku}} \\ &= \frac{1}{2\pi r^k}\int_0^{2\pi} [\Re(Q(re^{iu}))\cos(ku) + \Im(Q(re^{iu}))\sin(ku)]\,du \\ &= \frac{2}{\pi r^k}\int_0^{\pi} \Re(Q(re^{iu}))\cos(ku)\,du \end{aligned} \tag{5.1.17}$$

Proof: Taking $Q(z) \equiv \sum_{t=0}^{2^m-1} c_{m,k} z^t$ and applying the change of variable and given that $e^{iut} = \cos(tu) + i\sin(tu)$

$$Q(re^{iu}) = \sum_{t=0}^{2^m-1} c_{m,k} r^t e^{itu} = \sum_{t=0}^{2^m-1} c_{m,k} r^t [\cos(tu) + i\sin(tu)] \tag{5.1.18}$$

$$\Re(Q(re^{iu})) = \sum_t c_{m,k} r^t \cos(tu) \quad \text{and} \quad \Im(Q(re^{iu})) = \sum_t c_{m,k} r^t \sin(tu) \tag{5.1.19}$$

Substituting Equation (5.1.18) in the Cauchy's integral and using Euler's formula

$$e^{-iku}e^{itu} = \cos(ku)\cos(tu) + i\cos(ku)\sin(tu) - i\sin(ku)\cos(tu) + \sin(ku)\sin(tu) \tag{5.1.20}$$

$$\begin{aligned} c_{m,k} &= \frac{1}{2\pi r^k}\int_0^{2\pi} Q(re^{iu})e^{-iku}\,du \\ &= \frac{1}{2\pi r^k}\int_0^{2\pi} \sum_t c_{m,k} r^t \cos(tu)\cos(ku) + \sum_t c_{m,k} r^t \sin(tu)\sin(ku)\,du \\ &+ \frac{i}{2\pi r^k}\int_0^{2\pi} \sum_t c_{m,k} r^t \cos(ku)\sin(tu) - \sum_t c_{m,k} r^t \sin(ku)\cos(tu)\,du \end{aligned} \tag{5.1.21}$$

From the second integral (b)

$$\begin{aligned} (b) &= \frac{i\sum_t c_{m,k} r^t}{2\pi r^k}\int_0^{2\pi} \cos(ku)\sin tu - \sin(ku)\cos(tu)\,du \\ &= \frac{i\sum_t c_{m,k} r^t}{2\pi r^k}\int_0^{2\pi} \sin(k-t)u\,du = \frac{i\sum_t c_{m,k} r^t}{2\pi r^k} 2\pi^2 \sin(k-t) = 0 \quad \text{if } k = t \end{aligned} \tag{5.1.22}$$

Therefore the second integral is cancelled. Following a similar procedure with the first integral (a)

$$(a) = \frac{\sum_t c_{m,k} r^t}{2\pi r^k}\left[\int_0^{2\pi} \cos(tu)\cos(ku)\,du + \int_0^{2\pi} \sin(tu)\sin(ku)\,du\right] \tag{5.1.23}$$

and given that $\{\cos(nu)\}_{n=0}^{\infty}$ and $\{\sin(nu)\}_{n=1}^{\infty}$ are mutually orthogonal on $0 \le u \le 2\pi$

$$\int_0^{2\pi} \cos(nu)\cos(mu)\,du = \int_0^{2\pi} \sin(nu)\sin(mu)\,du = \begin{cases} \pi & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \tag{5.1.24}$$

This proves that

$$\int_0^{2\pi} \Re(Q(re^{iu}))\cos(ku)\,du = \int_0^{2\pi} \Im(Q(re^{iu}))\sin(ku)\,du \tag{5.1.25}$$

$$c_{m,k} = \frac{1}{2\pi r^k} \int_0^{2\pi} 2\Re(Q(re^{iu})) \cos(ku)\, du \tag{5.1.26}$$

and since $cos(\cdot)$ is an even function, we obtain

$$c_{m,k} = \frac{2}{\pi r^k} \int_0^{\pi} \Re(Q(re^{iu})) \cos(ku)\, du \tag{5.1.27}$$

□

5.2 Implementation

This Section aims to show the essential algorithms in order to obtain an efficient implementation of the WA method. The challenging part of this method is the computation of the coefficients $c_{m,k}$. Firstly, $\Re(Q(re^{iu}))$ is computed and subsequently a numerical integration of an oscillatory integral is required. As shown later, the first steps can be replaced by performing a complex Gaussian quadrature, which reduces the computation time substantially.

5.2.1 Computation of $\Re(Q(re^{iu}))$

First we explain in detail the computation of the coefficients in the wavelet expansion approximation method. We present all the steps considered in [25], and we generalize the procedure when considering non-normal density distribution functions by using the double-exponential transformation.

$$Q(re^{iu}) = \frac{M_L(-2^m \ln(re^{iu}) - (re^{iu})^{2^m}}{2^{m/2}(1 - re^{iu})} \tag{5.2.1}$$

$$\Re(Q(re^{iu})) = \frac{\Re(z_1)\Re(z_2) + \Im(z_1)\Im(z_2)}{(\Re(z_2))^2 + (\Im(z_2))^2} \tag{5.2.2}$$

where

$$\Re(z_1) = \Re(M_L(-2^m \ln(re^{iu})) - r^{2^m} \cos(2^m u), \quad \Re(z_2) = 2^{m/2}(1 - r\cos(u)) \tag{5.2.3}$$

and

$$\Im(z_1) = \Im(M_L(-2^m \ln(re^{iu})) - r^{2^m} \sin(2^m u), \quad \Re(z_2) = -2^{m/2} r \sin(u) \tag{5.2.4}$$

$$M_L(-2^m \ln(re^{iu})) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y) r^{2^m E_n \Lambda_n} e^{i2^m E_n \Lambda_n u}] f(y)\, dy \tag{5.2.5}$$

As shown in Section 4.2 we can approximate this integral for general distribution functions using the Double Exponential quadrature

$$M_L(-2^m \ln(re^{iu})) \approx \frac{\pi}{2} h \sum_{j=1}^{N} w_j \overline{M_L}(-2^m \ln(re^{iu}); x_j) \tag{5.2.6}$$

and

$$\overline{M_L}(-2^m \ln(re^{iu}); x) = \prod_{n=1}^{N} [1 - p_n(x) + p_n(x) r^{2^m E_n \Lambda_n} e^{i2^m E_n \Lambda_n u}] \tag{5.2.7}$$

Using polar coordinates the previous cast into

$$\overline{M_L}(-2^m \ln(re^{iu}); x) = \prod_{n=1}^{N} (R_n)_{\theta_n} = \left(\prod_{n=1}^{N} R_n \right) e^{\sum_{n=1}^{N} \theta_n} \tag{5.2.8}$$

where $R_n = |z_n|$ and $\theta_n = \arctan(\Im(z_n)/\Re(z_n))$

$$z_n = 1 - p_n(x) + p_n(x) r^{2^m E_n \Lambda_n} (\cos(2^m E_n \Lambda_n u) + i \sin(2^m E_n \Lambda_n u))$$

Finally, above expressions can be easily expanded,

$$R_n = \sqrt{(1 - p_n(x))^2 + r^{2^{m+1} E_n \Lambda_n} p_n^2(x) + 2p_n(x)(1 - p_n(x)) r^{2^m E_n \Lambda_n} \cos(2^m E_n \Lambda_n u)}$$

$$\theta_n = \arctan\left(\frac{r^{2^m E_n \Lambda_n} p_n(x) \sin(2^m E_n \Lambda_n u)}{1 - p_n(x) + p_n(x) r^{2^m E_n \Lambda_n} \cos(2^m E_n \Lambda u)} \right)$$

therefore,

$$\Re(\overline{M_L}(-2^m \ln(re^{iu}); x)) = \left(\prod_{n=1}^{N} R_n \right) \cos\left(\sum_{n=1}^{N} \theta_n \right) \tag{5.2.9}$$

$$\Im(\overline{M_L}(-2^m \ln(re^{iu}); x)) = \left(\prod_{n=1}^{N} R_n \right) \sin\left(\sum_{n=1}^{N} \theta_n \right) \tag{5.2.10}$$

and

$$\Re(M_L(-2^m \ln(re^{iu}))) \approx \frac{\pi}{2} h \sum_{j=1}^{N} w_j \Re(\overline{M_L}(-2^m \ln(re^{iu}); x_j)) \tag{5.2.11}$$

$$\Im(M_L(-2^m \ln(re^{iu}))) \approx \frac{\pi}{2} h \sum_{j=1}^{N} w_j \Im(\overline{M_L}(-2^m \ln(re^{iu}); x_j)) \tag{5.2.12}$$

With these two equations we are able to compute expression (5.2.2). One can easily see that this approach avoids the computations on the complex plane taking advantage of the polar coordinates. However, this approach implicates an increase in terms of the number of computations, therefore we compare the described procedure in [25] with our procedure based on computing $\Re(Q(re^{iu}))$ on the complex plane. For a better comparison we evaluate both approaches considering a normal density function, which permits the use of Gauss-Hermite formula. The main difficulty is to rewrite the code to accept complex numbers, which can be carry out very efficiently in programming languages like C++. To do so, we use `std::complex<double>`, see below a possible implementation in C++.

Listing 5.1: MGF on the complex plane - code C++

```
struct MGFComplex {
  // Compute Moment generating function M_L(s, Y)
  const double pd, rho;
  const int N;
  const std::complex<double> s,E;

  MGFComplex(const std::complex<double> ss, const double default_probability,
      const double correlation, const int assets, const std::vector<double> &
      exposures): s(ss), pd(default_probability), rho(correlation), N(assets),
       E(exposures) {};

  std::complex<double> operator () (const double y)
  {
    ObligorDP_GH obligor(rho, pd);
```

```
    double dp = obligor.operator()(y);
    std::complex<double> prod=1;
    for (int n = 0; n < N; n++) {prod *= 1 - dp + dp*exp(-s*E[n]);}
    return prod;
  }
};
```

The routine for computing the unconditional MGF is practically equal to the routine employed in FTM. So a library should include only one routine and modify it using `template` and passing the conditional MGF/CF.

Listing 5.2: Unconditional MGF - code C++

```
inline std::complex<double> UnconditionalMGFComplex(const std::complex<double>
     s, const double p_n,const double rho, const std::vector<double> &
    exposures) {
  // Compute Unconditional MGF (M_L(s)) using 20-nodes Gauss-Hermite.
  const int N = exposures.size();
  MGFComplex mgf (s, p_n, rho, N, exposures);
  const double sq_pi = 1/sqrt(pi);

  std::complex<double> u = sq_pi * GaussHermiteComplex(mgf, 20);
  return u;
}
```

Finally, the integrand of Equation (5.1.27) is computed as follows

Listing 5.3: Compute Q - code C++

```
struct Qcomplex {
  const double p, rh, mm, rr;
  const int l, kk;
  const std::vector<double> EE;

  Qcomplex (const double p_n, const double rho, const int m, const int k,
      const std::vector<double> E, const double r = 0.9995,
  const int nodes=20): p(p_n), rh(rho), mm(m), EE(E), rr(r), l(nodes), kk(k)
      {};

  double operator() (const double u)
  {
    std::complex<double> re = rr*exp(std::complex<double>(0, u));
    std::complex<double> s = -pow(2, mm)* log(re);
    std::complex<double> M_L = UnconditionalMGFComplex(s, p, rh, EE);
    std::complex<double> z2 = pow(re, pow(2, mm));
    std::complex<double> z3 = pow(2, mm/2)*(std::complex<double>(1,0)-re);

    std::complex<double> result = (M_L - z2)/z3;

    return real(result)*cos(kk*u);
  }
};
```

We have tested these two options efficiently implemented, obtaining in average a 25% of computation time reduction with our approach, requiring approximately 4 milliseconds of computation time, which is a reasonable figure given the difficulty of the integrand.

5.2.2 Computation of the coefficients $c_{m,k}$

The authors in [24, 25] consider the trapezoidal rule for solving integral (5.1.27), taking 2^m subintervals, whose error in terms of m is given by

$$E(m) = f''(\xi)\frac{\pi^3}{12 \cdot 2^{2m}}, \quad 0 \leq \xi \leq \pi \tag{5.2.13}$$

the error to Trapezoidal rule is proportional to f'' and converges quadratically [1] with rate of convergence 2^{-2m}. A more interesting method is the extended trapezoidal rule [33], which first computes the crudest estimate of $\int_a^b f(x)\,dx$ and subsequently improves the accuracy by adding 2^{n-2} for $n = 2, 3, \ldots$ additional interior points until a given degree of accuracy has been achieved. Therefore this algorithm involve adding successive stages of refinement, evaluating the integrand only at those new points necessary to refine the grid.

In order to figure out if there is a more suitable method for computing $c_{m,k}$, we can analyse and represent the integrand $\Re(Q(re^{iu}))\cos(ku)$ over the interval $[0, \pi]$. This will show how the integrand behaves as the frequency parameter k increases. Figure 5.1 shows a highly oscillatory integrand with very fast decay. This fact has some implications that justify the usage of more simple algorithms as stated below.

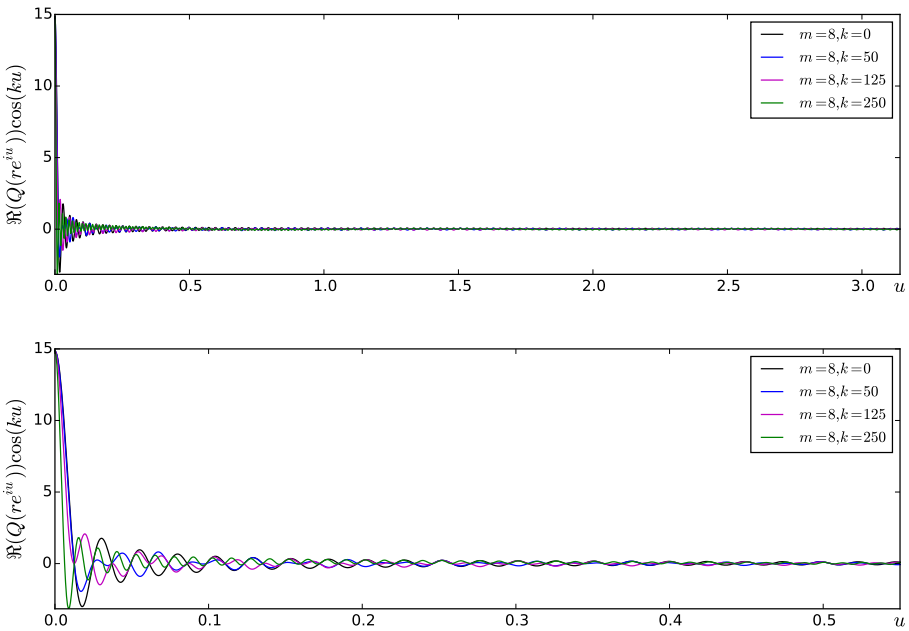


Figure 5.1: Plot function $\Re(Q(re^{iu}))\cos(ku)$ in $u = [0, \pi]$.

There are others method to compute highly oscillatory integrals. We showed in Subsection 2.4.1 the QAWO algorithm and the numerical steepest descent method. In this case the latter cannot be applied due to the function is not analytic but it is computed via numerical integration. Therefore, we are restricted to adaptive quadrature methods (QAG). Table 5.2.1 show the accuracy, number of iterations and compares the four methods by the ratio of best method's runtime to the rest.

[1] $\exists M \in \mathbb{R} : \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} \leq M, \quad \forall k$ large enough

Method	iterations	result	Rel.Error	Ratio
QAWO (GSL)	108	0.052649594467716	-	8
QAG GK41(GSL)	16	0.052649594467716	-	3.5
Trapezoidal	256	0.052649557000781	7.12×10^{-7}	2
Ext. Trapezoidal(10^{-4})	9	0.052649589486008	9.46×10^{-8}	1

Table 5.2.1: Methods for computing $c_{m,k}$.

The Extended trapezoidal rule stands as the fastest method with remarkable accuracy for this kind of integrals. We suggest the implementation of this algorithm when developing a package for the WA method.

5.2.3 VaR computation

The computation of the VaR at confidence level α in the WA method considers an approximation in a level of resolution m. The approximation is given by

$$l^W_{\alpha,m} = \frac{2\bar{k}+1}{2^{m+1}} \tag{5.2.14}$$

for a certain $\overline{k} \in 0, 1, \ldots, 2^m - 1$, where it holds that

$$\overline{F}_m(l^W_{\alpha,m}) \approx \alpha \tag{5.2.15}$$

The procedure to obtain $l^W_{\alpha,m}$ is based on a bisection with m maximum steps, which avoids root-finding algorithms that returns the closest value to α in a m resolution approximation, see pseudocode below.

Algorithm 6 Algorithm to compute VaR in WA method

```
function VaR_WA(m, α)
    k = 2^(m−1)
    k̄ = k
    k_next = 2^(m−2)
    c = c_{m,k}                                  ▷ compute coefficient c_{m,k}
    diff_1 = |c · 2^(m/2) − α|
    for j := 1 to m do
        if c · 2^(m/2) > α then
            k = k − k_next
        else
            k = k + k_next
        end if
        k_next = k_next/2
        c = c_{m,k}
        diff_2 = |c · 2^(m/2) − α|
        if diff_2 < diff_1 then
            k̄ = k
        end if
    end for
    return k̄
end function
```

5.3 Extension: Stochastic Loss Given Default

Similarly to the FTM, we have been considering a portfolio with deterministic loss given default. We now extend the WA method in order to consider stochastic loss given default. The main difference with respect to the FTM lies on the usage of the moment generating function instead of the characteristic function, therefore we briefly provide the expressions for the moment generating functions for the same distributions previously studied.

Case 1: Normal distribution

$$M_{\Lambda_n}(s, E_n) = e^{-sE_n\mu_n} e^{\frac{\sigma_n^2}{2} s^2 E_n^2} \tag{5.3.1}$$

$$M_L(s) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y) e^{-sE_n\mu_n} e^{\frac{\sigma_n^2}{2} s^2 E_n^2}] \phi(y)\, dy \tag{5.3.2}$$

Case 2: Triangular distribution

$$M_{\Lambda_n}(s, E_n) = e^{-s\mu_n} \mathrm{sinc}\left(\frac{\sqrt{6}\sigma_n s E_n}{2}\right)^2 \tag{5.3.3}$$

Case 3: Gamma distribution

$$M_{\Lambda_n}(s, E_n) = (1 + i\beta_n s E_n)^{-\alpha_n} \tag{5.3.4}$$

$$M_L(s) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y) e^{-sE_n\mu_n} e^{\frac{\sigma_n^2}{2} s^2 E_n^2}] \phi(y)\, dy \tag{5.3.5}$$

Case 4: Beta distribution (α, β)

$$MGF(s) = \sum_{k=0}^{\infty} \frac{s^k}{k!} \frac{B((\alpha + k, \beta)}{B(\alpha, \beta)} = 1 + \sum_{k=1}^{\infty} \frac{s^k}{k!} \prod_{n=0}^{k-1} \frac{\alpha + n}{\alpha + \beta + n} = {}_1F_1(\alpha; \alpha + \beta; s) \tag{5.3.6}$$

$$M_L(s) = \int_{-\infty}^{\infty} \prod_{n=1}^{N} [1 - p_n(y) + p_n(y) {}_1F_1(\alpha_n; \alpha_n + \beta_n; -sE_n)] \phi(y)\, dy \tag{5.3.7}$$

In this case ${}_1F_1$ can be computed using most of the packages listed in Section 2.4, since $Im(z) = 0$. If $z \to \infty$ there exist asymptotic series for real z

$${}_1F_1(a; b; z) \sim \frac{\Gamma(b) e^z z^{a-b}}{\Gamma(a)} \sum_{j=0}^{\infty} \frac{b - a_j 1 - a_j z^{-j}}{j!} = \frac{\Gamma(b) e^z z^{a-b}}{\Gamma(a)} {}_2F_0(b - a; 1 - a; -; z^{-1}) \tag{5.3.8}$$

where the series representation of ${}_2F_0$ is a divergent hypergeometric function. Due to time constraints, this extension has not been implemented, although is part of our ongoing work. In our future implementation we will integrate the GSL library for computing the confluent hypergeometric function for real arguments, since it has shown to be accurate for this regime of values.

5.4 Numerical examples

In this Section we test the WA method for different portfolios. We are primarily interested in the VaR computation, whose value will be compared with respect to the results obtained by crude Monte Carlo with 5 million simulations and the FTM. Additionally, portfolios 4 and 5 aim to test the efficiency of the WA method for totally diversified homogeneous portfolios of small and medium size.

Portfolio 1. This portfolio has $N = 10000$ obligors with $p_n = 0.01$, $\rho_n = 0.15$ and $E_n = 1/n$ for $n = 1, \ldots, N$ as in [24].

Portfolio 2. This portfolio has $N = 1001$ obligors with $p_n = 0.0033$, $\rho_n = 0.2$ and $E_n = 1$ for $n = 1, \ldots, N-1$ and $E_{1001} = 100$ as in [25].

Portfolio 3. This portfolio has $N = 1000$ obligors with $p_n = 0.003$, $\rho_n = 0.15$ and $E_n = 1/n$ for $n = 1, \ldots, N$ as in [24].

Portfolio 4. This portfolio has $N = 100$ obligors with $p_n = [0.001, 0.01, 0.1]$, $\rho_n = [0.02, 0.1, 0.2, 0.5]$ and $E_n = 1$ for $n = 1, \ldots, N$.

Portfolio 5. This portfolio has $N = 1000$ obligors with $p_n = [0.001, 0.01, 0.1, 0.2]$, $\rho_n = [0.02, 0.1, 0.2, 0.5]$ and $E_n = 1$ for $n = 1, \ldots, N$.

For these examples we use WA with 64 nodes of Gauss-Hermite quadrature and the coefficients are computed using 2^m intervals for the trapezoidal rule and $r = 0.9995$ as suggested in [25].

Table 5.4.1 shows the VaR obtained for the portfolio 1 for different scale m. Given the size of this first portfolio, the computation time is considerably small. For example, the same portfolio computed using FTM exceeded our computation time threshold of 30 minutes.

Scale	$\boldsymbol{VaR_{0.999}}$	**Rel.Error**	**Time(s)**
$m = 8$	0.162109	0.25%	82.76
$m = 9$	0.163086	0.86%	164.13
$m = 10$	0.161621	0.05%	388.36

Table 5.4.1: VaR of portfolio 1 for $m \in \{8, 9, 10\}$. Relative error with respect to Monte Carlo with 5 millions scenarios.

Portfolio 2 serves us to compare the accuracy of the WA in medium portfolios with high exposure concentrations. The same portfolio was tested for the FTM. The results show a lower relative error for very high loss levels using the WA method, whereas the FTM seems to perform better for lower confidence levels, devoting a similar computation time.

Scale	$\boldsymbol{VaR_{0.999}}$	**Rel.Error**	**Time(s)**	$\boldsymbol{VaR_{0.9999}}$	**Rel.Error**	**Time(s)**
$m = 10$	0.106934	0.71%	27.61	0.153809	0.40%	29.36
$m = 11$	0.108154	0.42%	56.76	0.153076	0.08%	64.86
FTM	0.107400	0.28%	58.50	0.152861	0.22%	58.50

Table 5.4.2: VaR of portfolio 2 for $m \in \{10, 11\}$. Relative error with respect to Monte Carlo with 5 millions scenarios.

In portfolio 3 we have increased the scale m until the relative error compared to MC is significantly low. The obtained results highlight a convergence as m increases. This same

behaviour is experimented in portfolio 1. Our experiments show a superior performance of the WA method when the portfolios are heavily concentrated, whereas the accuracy decreases substantially when considering well-diversified portfolios with large number of assets or assets with small exposures (the latter situation often leads to simplifications considering a granular portfolio), which is evidenced in portfolios 4 and 5.

Scale	$\boldsymbol{VaR_{0.999}}$	**Rel.Error**	**Time(s)**
$m = 8$	0.138672	1.30%	6.77
$m = 9$	0.139648	0.61%	15.00
$m = 10$	0.140137	0.26%	30.93
$m = 11$	0.140381	0.08%	53.91
$m = 12$	0.140503	0.00%	110.23

Table 5.4.3: VaR of portfolio 3 for $m \in \{8, 9, 10, 11, 12\}$. Relative error with respect to Monte Carlo with 5 millions scenarios.

$\boldsymbol{p} \setminus \boldsymbol{\rho}$	0.02	0.1	0.2	0.5
0.001	0.021973 (**23.27%**)	0.029785 (**12.14%**)	0.040527 (**11.70%**)	0.108887 (**2.68%**)
0.01	0.060059 (**8.81%**)	0.101074 (**2.64%**)	0.159668 (**4.16%**)	0.419434 (**2.06%**)
0.1	0.249512 (**1.45%**)	0.399902 (**1.70%**)	0.560059 (**1.55%**)	0.909668 (**0.30%**)

Table 5.4.4: VaR value at 99.9% for portfolio 4. Relative error with respect to Gauss-Hermite quadrature with 48 nodes.

$\boldsymbol{p} \setminus \boldsymbol{\rho}$	0.02	0.1	0.2	0.5
0.001	0.007324 (**4.13%**)	0.015137 (**0.15%**)	0.028809 (**3.53%**)	0.087402 (**17.07%**)
0.01	0.032715 (**0.57%**)	0.079590 (**0.81%**)	0.142090 (**3.94%**)	0.386230 (**13.05%**)
0.1	0.202637 (**0.31%**)	0.374512 (**0.85%**)	0.536621 (**1.99%**)	0.884277 (**2.11%**)
0.2	0.347168 (**0.34%**)	0.556152 (**0.76%**)	0.720215 (**1.30%**)	0.968262 (**0.42%**)

Table 5.4.5: VaR value at 99.9% for portfolio 5. Relative error with respect to Monte Carlo using Median Latin Hypercube Sampling and applying tan transformation.

$\boldsymbol{p} \setminus \boldsymbol{\rho}$	0.02	0.1	0.2	0.5
0.001	0.009277 (**3.23%**)	0.024902 (**1.60%**)	0.058105 (**2.23%**)	0.269043 (**9.01%**)
0.01	0.040527 (**0.53%**)	0.116699 (**0.99%**)	0.235840 (**3.13%**)	0.679199 (**4.01%**)
0.1	0.229980 (**0.22%**)	0.462402 (**0.94%**)	0.673340 (**1.42%**)	0.974121 (**0.26%**)
0.2	0.382324 (**0.24%**)	0.644043 (**0.59%**)	0.825684 (**0.57%**)	0.995605 (**0.01%**)

Table 5.4.6: VaR value at 99.99% for portfolio 5. Relative error with respect to Monte Carlo using Median Latin Hypercube Sampling and applying tan transformation.

These numerical experiments allow us to create a classification of methods used according to the characteristics of the portfolio. For well-diversified homogeneous portfolios, MC + MLHS applying the tan transformation can be used either for small or big portfolio with an appreciable balance between accuracy and computation time. Additionally, one can use the numerical integration techniques for small and medium portfolios, typically less than 1000-2000 assets or FTM, although the latter is not competitive in terms of computation time. For heavily concentrated portfolios of small and medium size, we can choose between the FTM (especially if we are interested in the whole loss distribution) or the WA method if we are mainly interested in risk measures. For bigger portfolios, the WA method tends to perform better with a sufficient

accuracy, whereas MC provides the most accurate results increasing the computation time significantly, unworkable in portfolios of millions of assets. Summarizing, there is no ideal method for all situations rather different methods need to be applied for different situations. From our point of view, the WA method can be used as a tool to compute risk measures, but financial institutions are also interested in other metrics such as *the standard deviation-over-mean* or *diversity score*, which are commonly used for ABS and MBS transactions. These measures cannot be computed without the whole loss distribution, which implies that research on this field is needed in order to find a method able to embrace all the good properties of the methods studied throughout this thesis.

Chapter 6

Future work

Throughout this thesis several methods based on factor models have been explored. As has been demonstrated, there is still ongoing research in order to obtain a robust method able to compute the portfolio loss distribution quickly and accurately, especially for large portfolios. Our future research aims to find alternative methods for the computation of the portfolio loss distribution based on numerical transform inversion techniques.

For much of this thesis we focused on one-factor models. A recent study of methods for computing credit loss distribution for multi-factor models can be found in [15], where they conclude that the plain Monte Carlo method is the best method in terms of speed and accuracy, followed by the saddlepoint approximation. Furthermore, they found that numerical transform inversion methods, which numerically inverts the Bromwich integral, become unstable due to the highly oscillatory behaviour of the integral. From our perspective, these inconveniences can be handled using some of the techniques presented in Chapter 2. Additionally, we are interested in the application of Monte Carlo with importance sampling in order to speed up the calculations.

Ideally, a new method for multi-factor models should be adaptable in order to allow generalizations. Therefore, it might be worthwhile trying to extend the numerical methods presented in Chapter 4 for the general case. Furthermore, we are interested in considering other statistical distribution for the factor models, such as Generalized extreme value distribution or Hyperbolic distribution. Finally, we would like to explore if nonlinear least square - curve fitting algorithms can be used to obtain a factor model distribution given a series of data points.

An ultimate goal is to develop a numerical library/software for credit risk modelling with several configurable methods and analytic tools for quantifying portfolio credit risk.

Regarding numerical methods, we are particularly focused on the computation of confluent hypergeometric function when parameters and argument are complex. Several methods exist based on asymptotic expansions, which tend to return a significant error term if the variables or argument are not sufficiently large. The method presented in Chapter 2 might be extended to deal with parameters without real part and extreme combinations of values can be handled by using recurrence relations. Finally, a method able to reliably computing the Tricomi confluent hypergeometric function, which might be less complicated, can be used on the computation of several hypergeometric function if the parameter b is not an integer.

Chapter 7

Conclusions

We began this thesis by investigating different numerical methods in Chapter 2. The aim of Chapter 2 was to study and improve the numerical methods applied on the computation of several statistical distributions and special functions. We demonstrated that different approaches can speed up most of the methods currently implemented in well known R packages. Regarding special functions, we focused on the confluent hypergeometric function needed for the computation of the characteristic function of the beta distribution. We studied several state-of-the-art methods and we created a new algorithm for this special case.

In Chapter 3, we introduced some of Monte Carlo methods, where we focused on Latin hypercube sampling techniques and their application on the computation of definite and indefinite integrals. We tested three variants of Latin hypercube sampling techniques and concluded that Median Latin hypercube sampling provides the most satisfactory results in terms of number of samples and accuracy.

In Chapter 4, we described some of the most widely used models for the calculation of default probabilities in portfolio credit risk. We introduced the Vasicek one-factor model and its generalization for factors following non-normal distributions, where we provided a general algorithm. Similarly, we presented the large portfolio approximation method and we generated closed-form expressions for the so-called general loss distribution. Furthermore, we investigated the Fourier transform method applied for medium portfolio with exposure concentrations and loss given default. We applied the new algorithm for the computation of the characteristic function of the beta distribution, described in Chapter 2, for the case when the beta distribution is used for modelling loss given default.

Finally, in Chapter 5, we introduced the Haar wavelet approximation to computing VaR. We described in detail the theoretical framework behind the method and we provided a modification in order to increase the flexibility of the method. We investigated a computational enhancement to speed up the method, which considers computations on the complex plane and a better method for the computation of highly oscillatory integrals with fast decay. Furthermore, we compared this method with respect to those previously studied for several types of portfolios, concluding that the wavelet approximation is applicable for large portfolios with high exposure concentrations where other methods require an unaffordable computation time. Moreover, as stated, the method can be extended by including stochastic loss given default by means of several moment-generating functions.

Bibliography

[1] M. Abramowitz, I. Stegun.*Handbook of Mathematical Functions*, Dover Publications, New York (1972).

[2] David H. Bailey. *Tanh-Sinh High-Precision Quadrature*, Lawrence Berkeley National Laboratory, (2006).

[3] O. Barndorff-Nielsen. *Normal Inverse Gaussian Distributions and Stochastic Volatility Modelling*, Scandinavian Journal of Statistics, (1997).

[4] Necdet Batir. *Very accurate approximations for the factorial function*, Journal of Mathematical Inequalities, (2010).

[5] I. Bogaert, B. Michiels, and J. Fostier, *O(1) computation of Legendre polynomials and Gauss-Legendre nodes and weights for parallel computing*, SIAM J. Sci. Comput., 34, pp. C83-C101, (2012).

[6] Jonathan M. Borwein, David H. Bailey and Roland Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*, A K Peters, pages 312-313, (2003).

[7] R. P. Brent. *Algorithms for Minimization Without Derivatives.* Englewood Cliffs, NJ: Prentice-Hall, ch. 3-4, (1973).

[8] R. H. Byrd, P. Lu and J. Nocedal. *A Limited Memory Algorithm for Bound Constrained Optimization*, SIAM Journal on Scientific and Statistical Computing , 16, 5, pp. 1190-1208, (1995).

[9] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Comput. Sci. Appl. Math., Academic Press, New York, (1984).

[10] Deaño, A. and Huybrechs, D. *Complex Gaussian quadrature of oscillatory integrals*, Technical Report 2008/NA04, University of Cambridge, (2008).

[11] A. Debuysscher, M. Szegö. *The Fourier Transform Method - Technical Document*, Moody's Investors Service (2003).

[12] William Feller. *An Introduction to Probability Theory and its Applications*, Wiley, New York, 3d edition, (1968).

[13] R. Fourer, D. Gay, B. Kernighan. *AMPL: A Modelling Language for Mathematical Programming*, Cengage Learning, (2002).

[14] T. Gerstner and M. Griebel. *Numerical integration using sparse grids*, Numerical Algorithms 18(3-4), pp. 209-232, (1998).

[15] P. Glasserman, J. Ruiz-Mata. *Computing the credit loss distribution in the Gaussian copula model: a comparison of methods*, Journal of credit risk 2(4), 33-26, (2006).

[16] P. Glasserman. *Monte Carlo methods in financial engineering*, Springer, New York (2004).

[17] G. H. Golub and J. H. Welsch. *Calculation of Gauss quadrature rules*, Math. Comp., 23, pp. 221230, (1969).

[18] N. Hale, and A. Townsend. *Fast and Accurate Computation of GaussLegendre and Gauss Jacobi Quadrature Nodes and Weights*, SIAM Journal on Scientific Computing, 35, pp. A652A674, (2013).

[19] X. Huang, C.W. Oosterlee and M. Mesters. *Computation of VaR and VaR Contributions in the Vasicek portfolio credit loss model: a comparative study*, Delf University of Technology, Report 07-06, (2007).

[20] X. Huang, C.W. Oosterlee. *Generalized beta regression models for random loss given default*, The Journal of Credit Risk (1-26), Volume7/Number 4, (2011).

[21] Huybrechs, D. and Vandewalle, S. *On the evaluation of highly oscillatory integrals by analytic continuation*, SIAM Journal of Numerical Analysis, 76(360), pp.1955-1980, (2006).

[22] Steven G. Johnson. *Notes on the convergence of trapezoidal-rule quadrature*, Massachusetts Institute of Technology, (2010).

[23] A.Kalemanova, B.Schmid, R.Werner. *The Normal inverse Gaussian distribution for synthetic CDO pricing*, The Journal of Derivatives, Spring 2007, Vol. 14, No. 3: pp. 80-94.

[24] J.J. Masdemont, L.Ortiz-Gracia. *Credit risk contributions under the Vasicek one-factor model: a fast wavelet expansion approximation*, Journal of Computational Finance, (2011).

[25] J.J. Masdemont, L.Ortiz-Gracia. *Haar wavelets-based approach for quantifying credit portfolio losses*, Quantitative Finance, (2014).

[26] M. Mori, M. Sugihara. *The double-exponential transformation in numerical analysis*, Journal of Computational and Applied Mathematics, (2001).

[27] J.Nocedal, S.J.Wright. *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, (2006).

[28] T.N.L. Patterson. *On High Precision Methods for the Evaluation of Fourier Integrals with Finite and Infinite Limits*, Numerische Mathematik, 27, pp.41-52, (1976).

[29] John W. Pearson *Computation of Hypergeometric Functions*. MSc in Mathematical Modelling and Scientific Computing Dissertation, University of Oxford, (2009).

[30] John W. Pearson, Sheehan Olver, Mason A. Porter. *Numerical Methods for the Computation of the Confluent and Gauss Hypergeometric Functions*, arXiv:1407.7786, (2015).

[31] Piessens R and Branders M. *Algorithm 002: computation of oscillatory integrals*, Journal of Computational and Applied Mathematics 1, pp.153-164, (1975).

[32] Piessens R, de Doncker Kapenga E, Ueberhuber C and Kahaner D. *QUADPACK, A Subroutine Package for Automatic Integration*, Springer Verlag, (1983).

[33] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brain P. Flannery. *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press; 3 edition, (2007).

[34] Philipp J. Schönbucher. *Credit derivatives pricing models*, Wiley Finance, (2003).

[35] Philipp J. Schönbucher. *Factor Models for Portfolio Credit Risk*, (2000).

[36] H. Takahasi, M. Mori. *Double Exponential Formulas for Numerical Integration*, Publ. RIMS Kyoto Univ. 9, (1974).

[37] Dirk Tasche. *The Vasicek Distribution*, Lloyds TSB Bank - Corporate Markets Rating Systems, Bristol/London, (2008).

[38] Christoph W. Ueberhuber. *Numerical Computation 2: Methods, Software and Analysis*, Springer, New York (2013).

[39] O. Vasicek. *The Distribution of Loan Portfolio Value*, Risk, (2002).

[40] P. Wynn. *The epsilon algorithm and operational formulas of numerical analysis*, Math. Comp. 15, pp. 151-158, (1961).