Numerical Methods for the Valuation of Synthetic Collateralized Debt Obligations

by

Xiaofang Ma

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Numerical Methods for the Valuation of Synthetic Collateralized Debt Obligations

Xiaofang Ma

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2007

A Collateralized Debt Obligation (CDO) is a credit derivative that creates fixed income securities, which are known as tranches. A CDO is called a synthetic CDO if the risky assets in the underlying pool are credit default swaps. An essential part of the valuation of a synthetic CDO tranche is how to estimate accurately and efficiently the expected value of the tranche loss function. It is known that the expected value of a function of one random variable is completely determined by the distribution of the random variable and the function itself. A standard approach to estimate the expected value of a function of one random variable is to estimate the distribution of the underlying random variable, the pool loss in our case, and then to evaluate the expected value of the given function, the tranche loss function for our problem. Following this approach, we introduce three methods for estimating the distribution of the pool loss: a stable recursive method for computing the distribution of the pool loss exactly, an improved compound Poisson approximation method and a normal power approximation method for approximating the distribution of the pool loss.

We also develop a new method that focuses on the tranche loss function directly. The tranche loss function is expressed simply in terms of two bases functions. Each of the two bases functions is a transformation of the *hockey stick* function $h(x)$, where $h(x) = 1 - x$ if $0 \leq x < 1$ and 0 if $x \geq 1$. By approximating the hockey stick function by a sum of exponentials, the tranche loss function is approximated by a sum of exponentials.

The main advantage of this method is that the distribution of the pool loss need not be estimated. A crucial part of this new method is the determination of the coefficients of an exponential approximation to the hockey stick function. We discuss both the numerical method for computing the exponential approximation to the hockey stick function as well as the theoretical properties of the approximation.

Performance comparisons of the four new methods developed in this thesis and other standard methods for synthetic CDO valuation are presented.

Acknowledgements

Although many people have helped in the successful completion of this thesis, none has done so more directly than my supervisors, Professor Ken Jackson and Professor Alex Kreinin. Special thanks to you for your encouragement, support and patient guidance! I am grateful for having had the opportunity to benefit from your knowledge and intuition in facing challenging numerical problems. I have gained from you not only a better and deeper understanding of computational finance, but, most importantly, an eagerness to continue to study. It has been an honor to work with you!

Thanks to the members of my thesis committee, Professors Wayne Enright, Sheldon Lin, and Marcel Rindisbacher, for reading my thesis proposal and draft thesis, and providing insightful comments and suggestions for improvements. I have benefited greatly from your advice. Thanks also to Professor Christina Christara for very helpful discussions on theoretical properties of the exponential approximation to the hockey stick function. My collaborations with Dr. Ian Iscoe have been very rewarding and thoroughly enjoyable. Thanks Ian for your invaluable help.

Special thanks to Professor Tom Hurd for agreeing to be the external examiner for my final oral examination and Professor Tom Fairgrieve for volunteering to be a member of my final oral committee.

Thanks to Professor Chengxian Xu for his encouragement and help since 1991; to Dr. Alexander Tchernitser for his encouragement and help over the past six years; to Dr. Kit-Sun Ng for providing generous help in many ways since I arrived at the University of Toronto; to Jingrui Zhang, Wanhe Zhang and Xuping Zhang for fruitful discussions on academic research.

I would like to acknowledge the Natural Science and Engineering Research Council (NSERC) of Canada, the Ontario Graduate Scholarship Program (OGS), the School of Graduate Studies and the Department of Computer Science at the University of Toronto for their generous financial assistance. Without their support this research would not have been possible. I am grateful to Professor Ken Jackson and the Department of Computer Science at the University of Toronto for providing me the opportunity to pursue graduate studies in Canada.

My parents have given me their unquestioning support, encouragement and love. Their confidence in my ability to succeed has been a tremendous source of strength. Thanks Mom and Dad! I would like to thank my daughter Zhongshu and my son Zhongru. They are another source of strength. Special thanks to my wife, Bin, for her endless support and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A Collateralized Debt Obligation (CDO) is a credit derivative[1] that creates fixed income securities with widely different risk characteristics from a pool of risky assets. The coupon and principal payments of these securities are linked to the performance of the underlying pool. These fixed income securities are known as tranches and divided into senior, mezzanine and subordinated/equity tranches. Each of these tranches has a different level of seniority relative to the others in the sense that a senior tranche has coupon and principal payment priority over a mezzanine tranche, while a mezzanine tranche has coupon and principal payment priority over an equity tranche. It is important to note that a CDO only redistributes the total risk associated with the underlying pool of assets to the priority ordered tranches. It neither reduces nor increases the total risk associated with the pool.

A CDO is called a synthetic CDO if the risky assets in the underlying pool are credit default swaps (CDS')[2]. In this thesis, we focus on numerical methods for the valuation

[1]A derivative is a financial instrument whose price depends on, or is derived from, the price of another asset, for example, an option on a stock. A credit derivative is a derivative whose payoff depends on the creditworthiness of one or more entities, for example, a credit default swap.

[2]A credit default swap is a bilateral financial contract in which the CDS buyer, alternatively called the protection buyer, pays a periodic fee, also known as premium, which is expressed in basis points per annum on the notional amount and is called the *CDS spread* or the *credit spread*, in return for a contingent payment by the CDS seller, alternatively called the protection seller, upon a credit event (such as a default or restructuring) happening to the reference entity (such as a corporate bond). For

of synthetic CDO tranches.

1.1 Background

In this section, we give a very brief review of CDO markets. Besides synthetic CDOs, there are other types of CDOs. Depending on the nature of the risky assets in the underlying pool, a CDO may be called a *collateralized loan obligation* or a *collateralized bond obligation* if it holds only loans or bonds, respectively. CDOs can also be categorized based on the motivation of the issuer of a CDO. If the motivation of the issuer is to earn the difference between the average yield of the collateral assets and the payments made to the various tranches of the CDO, then the transaction is called an *arbitrage CDO.* If the motivation of the issuer is to remove debt instruments from its balance sheet, then the transaction is called a *balance sheet CDO.*

A CDO can be structured as either a cash flow or a synthetic transaction, or a hybrid of both. In a cash flow transaction, the CDO is backed by a pool of cash assets that are truly owned by the CDO sponsor. A synthetic CDO makes use of CDS' to transfer the credit risk of a pool of reference entities to tranche investors. The main difference between a cash flow CDO and a synthetic CDO is that in the latter no transfer of securities takes place. Cash flow CDOs dominated the CDO market in the early days, while synthetic CDOs account for a large portion of the overall CDO market now, partially due to the high liquidity of the CDS market and the appearance of the standard credit indexes, such as the Dow Jones CDX for North America and emerging markets and the Dow Jones iTraxx for Europe and Asia.

CDOs, which first appeared in the late 1980s, are considered the most important innovation in the structured finance[3] market in the past two decades. According to

more details concerning CDS, see, for example, [29].

[3]Structured finance is a type of financing in which the quality of the debt is assumed to be based on a direct guarantee from a creditworthy entity or on the credit quality of the debtor's assets, with or without credit enhancement, rather than on the financial strength of the debtor itself [51].

a report published by Celent [46], a research and consulting firm, the CDO market has experienced an average annual growth rate of 150% since 1998. Data released by the Bond Market Association shows that the gross global CDO issuance totalled US$171 billion in the first two quarters of 2006 [54]. It was estimated by Celent that the overall size of the CDO market in terms of outstanding contracts on a notional amount basis would grow to nearly US$2 trillion by the end of 2006. The synthetic CDO market has grown rapidly since the appearance of JP Morgan's Bistro deal [43], the first synthetic CDO, in December 1997. A survey conducted by Fitch Ratings revealed that the synthetic CDO market grew by 234% in 2004 to US$1.1 trillion in terms of outstanding contracts on a notional amount basis [18].

A good introduction to CDOs is *The ABC of CDO* by Moore [43]. More detailed discussion of CDOs can be found in the books by Banks [4], Culp [12], Deacon [15], Fabozzi and Choudhry [17], Goodmand and Fabozzi [23], and Tavakoli [52], the product guides by the Bank of America [6], JP Morgan [41], Lehman Brothers [44], Merrill Lynch [19], and Wachovia Securities [10], and research papers available at `http://www.defaultrisk.com`.

1.2 A synthetic CDO example

The structure of a typical synthetic CDO can be explained through the following example, which is illustrated in Figure 1.1. This synthetic CDO is based on a pool of 125 CDS'. The notional value of each CDS is US$8 million. The total notional value of the underlying pool is US$1 billion. The CDO has four priority ordered tranches: (i) an equity tranche with an initial notional value of US$30 million and a credit spread of 2000 basis points (bps) per annum; (ii) a mezzanine tranche with an initial notional value of US$50 million and a credit spread of 800 bps per annum; (iii) a senior tranche with an initial notional value of US$120 million and a credit spread of 100 bps per annum; and (iv) a super senior tranche with an initial notional value of US$800 million and a credit spread of 20

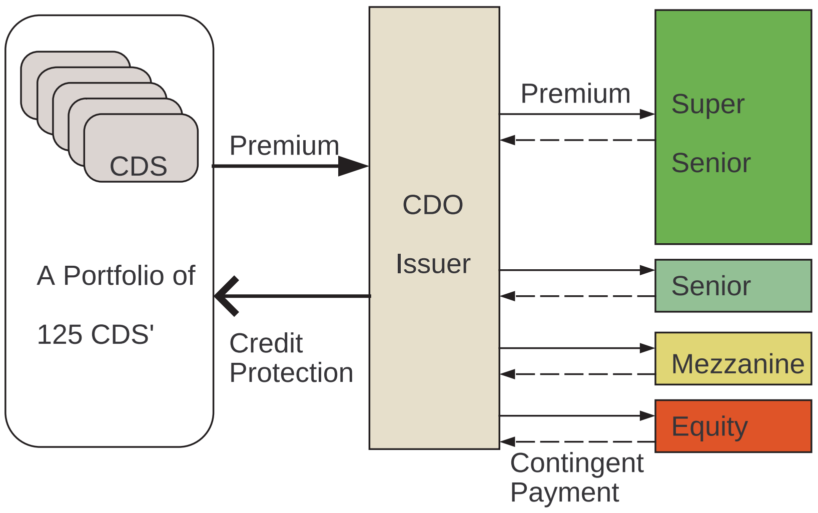

Figure 1.1: A synthetic CDO example

bps per annum. The equity tranche is the riskiest one; the mezzanine tranche bears a medium risk; the senior tranche is less risky and the super senior tranche has the lowest risk. The maturity of the CDO, thus the maturity of each tranche, is five years from now and the premiums are paid quarterly.

During the life of the CDO, each tranche may receive premiums quarterly from the CDO issuer. The premium that a tranche investor receives for a specified premium payment period is proportional to the remaining notional value of the specified tranche at the end of the premium payment period. Let us consider one scenario. Suppose there is no loss in the first quarter of the CDO life. Then the equity tranche investor receives $30 * 20\% * 0.25$ =US\$1.5 million from the CDO issuer; the mezzanine tranche investor receives $50 * 8\% * 0.25$ =US\$1 million, and so on. Suppose further that, in the second quarter, the underlying pool suffers a US\$6 million loss. Then the equity tranche investor absorbs this US\$6 million loss. The premium that the equity tranche investor receives for this period is $(30 - 6) * 20\% * 0.25$ =US\$1.2 million. In total, the equity tranche investor pays the CDO issuer US\$(6-1.2)=US\$4.8 million. The notional value of the equity tranche for the next premium payment period is reduced to US\$(30-6)=US\$24 million. The mezzanine tranche investor still receives $50 * 8\% * 0.25 =$US\$1 million from the CDO issuer, and so on. If the equity tranche's notional value has been reduced to zero before the maturity of the CDO, the mezzanine tranche investor would then be the next one to suffer losses. The pool would need to suffer a loss of US\$200 million before the super senior tranche investor would suffer any loss.

1.3 Valuation of a synthetic CDO tranche

1.3.1 Valuation equation

We consider the valuation of a specified synthetic CDO tranche. Let PV_D be the present value of the expected loss of the tranche and PV_P the present value of the expected premium that the tranche investor may receive over the life of the contract. Mathematically, they are defined as

$$PV_D = \mathbb{E}\left[\int_0^T D(t)\mathrm{d}L_t\right], \tag{1.1}$$

$$PV_P = \mathbb{E}\left[\int_0^T D(t)\mathrm{d}P_t\right], \tag{1.2}$$

where $D(t)$ is the risk-free discount factor[4], L_t and P_t are the cumulative tranche loss and cumulative tranche premium at time t, respectively, $\mathbb{E}$ denotes the expected value calculated under a risk-neutral measure[5], and T is the maturity of the contract. We make the standard assumption of independence between $D(t)$ and L_t.

A fair credit spread for a tranche is a constant credit spread that makes $PV_D = PV_P$. If a credit spread of a tranche is known, then the value of the tranche to the tranche investor, protection seller, is $PV_P - PV_D$.

[4]A discount factor, $D(t)$, is a number by which a future cash flow to be received at time t must be multiplied by in order to obtain the present value. For an annualized continuously compounded discount rate r, $D(t) = \exp(-rt)$.

[5]In mathematical finance, a risk-neutral measure is a probability measure in which today's fair, *i.e.*, arbitrage-free, price of a derivative is equal to the expected value (under the measure) of the future payoff of the derivative discounted at the risk-free rate.

Let $0 = t_0 < t_1 < t_2 < \cdots < t_n = T$ be the set of premium dates, where T is the maturity of the tranche. Then PV_D and PV_P can be approximated by

$$PV_D \approx \mathbb{E}\left[\sum_{i=1}^{n}(L_i - L_{i-1})D(t_i)\right], \tag{1.3}$$

$$PV_P \approx \mathbb{E}\left[\sum_{i=1}^{n} s(S - L_i)(t_i - t_{i-1})D(t_i)\right], \tag{1.4}$$

respectively, where L_i is the cumulative tranche loss at time t_i, s is the constant fair credit spread of the tranche, S is the initial notional value (also known as size) of the tranche. These discrete formulas are widely used in practice and give very good approximations to the continuous ones (1.1) and (1.2), respectively.

Noting that $D(t_i)$ and L_i are independent, it follows from (1.3) and (1.4) that the valuation of a tranche is reduced to the estimation of $\mathbb{E}[L_i], i = 1, 2, \ldots, n$. In the sequel, we use d_i to denote the expected value of the risk-free discount factor $D(t_i)$ in a risk-neutral measure. Then the fair credit spread s can be estimated by

$$s \approx \frac{\sum_{i=1}^{n} \mathbb{E}[L_i - L_{i-1}]\, d_i}{\sum_{i=1}^{n} \mathbb{E}[S - L_i]\,(t_i - t_{i-1})d_i}, \tag{1.5}$$

where $L_0 = 0$, due to the natural assumption that there is no default at t_0. The value of the tranche to the tranche investor is approximately

$$s\sum_{i=1}^{n} \mathbb{E}[S - L_i]\,(t_i - t_{i-1})d_i - \sum_{i=1}^{n} \mathbb{E}[L_i - L_{i-1}]\, d_i. \tag{1.6}$$

1.3.2 One-factor model

In this section we describe the one-factor model—the model used to estimate $\mathbb{E}[L_i]$.

Let K be the number of CDS' in the underlying pool. Accordingly, there are K reference entities associated with the K CDS'. Let N_k and R_k be the notional value and the recovery rate[6] of the notional value of the reference entity k, respectively. Define

[6]The recovery rate of a risky asset, for example, a corporate bond, is the fraction of the exposure, for example, the face value of the corporate bond, that may be recovered through bankruptcy proceedings or some other form of settlement in the event the issuer defaults.

the loss-given-default (LGD) of reference entity k as $LGD_k = N_k(1 - R_k)$. Let τ_k be the default time and $\pi_k(t) = \mathbb{P}(\tau_k \le t)$ the risk-neutral default probability[7] of reference entity k, respectively, where τ_k and t take discrete values from $\{t_1, t_2, \ldots, t_n\}$.

Let ℓ and u be the attachment and the detachment points of the specified tranche, respectively. The attachment point of a tranche is the threshold that determines whether some of the pool losses shall be absorbed by this tranche: if the pool loss is less than the attachment point of the tranche, then the tranche does not absorb any loss; otherwise it absorbs some of the losses. The detachment point is the threshold that determines when the tranche will be wiped out: if the pool loss is larger than the detachment point of the tranche, then it is wiped out; otherwise, its remaining notional value is larger than zero. The size of a tranche, $S = u - \ell$, determines the maximum loss that the tranche can absorb. In the previous synthetic CDO example, the attachment point, the detachment point and the size of the mezzanine tranche are US\$30 million, US\$80 million, and US\$50 million, respectively. As percentages of the total notional value of the underlying pool, they are 3%, 8%, and 5%, respectively. In the remainder of this thesis, the attachment point, the detachment point and the size of a tranche are quoted as percentages of the total notional value of the underlying pool.

Let $\mathscr{L}_i^P$ be the cumulative loss of the underlying pool at time t_i:

$$\mathscr{L}_i^P = \sum_{k=1}^{K} LGD_k \mathbf{1}_{\{\tau_k \le t_i\}},$$

where $\mathbf{1}_{\{\tau_k \le t_i\}} = 1$ if the k-th reference entity defaults on or before t_i, $\mathbf{1}_{\{\tau_k \le t_i\}} = 0$, otherwise. Then the cumulative tranche loss at time t_i is

$$L_i = f(\mathscr{L}_i^P; \ell, u) = \min\left(S, \max\left(\mathscr{L}_i^P - \ell, 0\right)\right),$$

where $S = u - \ell$. In this thesis, the function f is called the tranche loss function. This function is also associated with a special insurance policy with ordinary deductible ℓ and

[7]These probabilities are taken as input to a synthetic CDO tranche valuation model. They can be estimated, for example, by bootstrapping [56].

policy limit S in actuarial science [5], [38].

Let Y_k be the creditworthiness index of reference entity k and be defined by

$$Y_k = \beta_k X + \sigma_k \varepsilon_k, \tag{1.7}$$

where X is interpreted as a common risk factor, ε_k as an idiosyncratic risk factor, β_k and σ_k are constants satisfying $\beta_k^2 + \sigma_k^2 = 1$. The common risk factor X and the idiosyncratic risk factors ε_k are assumed to be mutually independent. The risk-neutral default probability and the creditworthiness index are related by

$$\pi_k(t) = \mathbb{P}\left(\tau_k \leq t\right) = \mathbb{P}\left(Y_k \leq H_k(t)\right), \tag{1.8}$$

where $H_k(t)$ is a threshold value determining whether reference entity k is in default or not at time t: reference entity k is in default if $Y_k \leq H_k(t)$; not in default, otherwise. The correlation structure of default events is captured by the common risk factor X. If we further assume, as we do in this thesis, that X and ε_k follow independent standard normal distributions, then Y_k also follows the standard normal distribution and from (1.8) we have $H_k(t) = \Phi^{-1}\left(\pi_k(t)\right)$, where Φ is the cumulative distribution function of the standard normal distribution. Furthermore, the correlation between two different creditworthiness indexes Y_i and Y_j is $\beta_i \beta_j$.

The conditional risk-neutral default probability of reference entity k is

$$\pi_k(t; x) = \mathbb{P}\left(Y_k \leq H_k(t) | X = x\right). \tag{1.9}$$

Thus from (1.7) and (1.9) we have

$$\pi_k(t; x) = \Phi\left(\frac{H_k(t) - \beta_k x}{\sigma_k}\right). \tag{1.10}$$

In this conditional independence framework, we have

$$\mathbb{E}\left[L_i\right] = \int_{-\infty}^{\infty} \mathbb{E}_x\left[L_i\right] \mathrm{d}\Phi(x), \tag{1.11}$$

where $\mathbb{E}_x[L_i] = \mathbb{E}_x\left[f(\mathscr{L}_i^P;\ell,u)\right]$ is the expected value of L_i conditional on $X = x$, $\mathscr{L}_i^P = \sum_{k=1}^K LGD_k \mathbf{1}_{\{Y_k \le H_k(t_i)\}}$ noting that $\mathbf{1}_{\{\tau_k \le t_i\}} = \mathbf{1}_{\{Y_k \le H_k(t_i)\}}$, and the indicator functions $\mathbf{1}_{\{Y_k \le H_k(t_i)\}}$ are mutually independent conditional on $X = x$.

The model just described is known as the one-factor model[8]. It was first introduced by Vasicek [57] to estimate the loan loss distribution and then generalized to portfolio derivative valuation by Li [40], Gordy and Jones [24], Hull and White [30], Iscoe, Kreinin and Rosen [34], Laurent and Gregory [39], and Schönbucher [50], to name just a few. In this thesis we use this model to illustrate our numerical methods. It is important to emphasize that all four numerical methods developed in this thesis are directly applicable to other more general models provided that the model belongs to the conditional independence framework, such as the double t–copula model proposed by Hull and White [30]. For a comparative analysis of different models, see the paper by Burtschell, Gregory and Laurent [9].

Generally, the integral in (1.11) needs to be evaluated numerically using an appropriate quadrature rule[9]:

$$\mathbb{E}[L_i] \approx \sum_{m=1}^{M} \mathrm{w}_m \mathbb{E}_{x_m}[L_i], \tag{1.12}$$

where x_m and w_m are the abscissas and weights of the chosen quadrature rule. One possible choice is Gaussian quadrature. The abscissas and weights of Gaussian quadrature rules for small values of M can be found in Chapter 25 of [1], while parameters for large values of M can be generated using well developed routines, such as those in [47].

Noting that $L_i = f(\mathscr{L}_i^P;\ell,u)$, we see from (1.12) that the fundamental numerical problem in synthetic CDO tranche valuation is how to evaluate $\mathbb{E}_{x_m}\left[f(\mathscr{L}_i^P;\ell,u)\right]$ for a fixed abscissa x_m at a fixed time t_i for the fixed attachment and detachment points ℓ and u. In the remainder of this thesis, we focus on computing this expected value,

[8]This one-factor model is also known as a Gaussian copula model [30] [39].

[9]The improper integral in (1.11) can be efficiently approximated by partitioning the infinite interval into a small number of subintervals and then applying special quadrature rules to each of the subintervals. In this thesis, we do not discuss those details. Instead, we use a special 30–point quadrature rule mentioned in [14].

which is denoted as $\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right]$, dropping the abscissa x_m and the time index i and writing $\mathscr{L}^P = \sum_{k=1}^K LGD_k \mathbf{1}_{\{k\}}$, where $\mathbf{1}_{\{k\}} = \mathbf{1}_{\{Y_k \le H_k(t_i)\}}$. Let Q_k be the probability that $\mathbf{1}_{\{k\}} = 1$. Then $S_k = 1 - Q_k$ is the probability that $\mathbf{1}_{\{k\}} = 0$. We emphasize again that the indicator functions $\mathbf{1}_{\{k\}}$ are mutually independent conditional on X.

1.4 Brief review of known methods

The Monte Carlo method plays an important role in computational finance. Though it is also used in synthetic CDO tranche valuation, it is generally used only as a benchmark due to its inefficiency, despite its flexibility [33].

As was emphasized in Section 1.3.2, the fundamental numerical problem in synthetic CDO tranche valuation is how to evaluate $\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right]$, where $\mathscr{L}^P$ is a sum of independent nonnegative random variables. The expected value of a function of a random variable is completely determined by two components: the distribution of the underlying random variable and the function itself. A standard approach to estimate the expected value of a function of a random variable is to estimate the distribution of the underlying random variable, $\mathscr{L}^P$ in our case, and then to evaluate the expected value of the given function, the piecewise linear tranche loss function $f(\mathscr{L}^P;\ell,u)$ for our problem, possibly using its special properties.

Almost all methods for synthetic CDO tranche valuation have focused on the first component: the distribution of $\mathscr{L}^P$, the pool's loss distribution. These methods can be divided into two classes. The first class computes a pool's loss distribution exactly. To do this we assume that a positive loss unit is given such that each LGD_k is an integer multiple of the given unit. Without loss of generality, we still use LGD_k to denote the loss-given-default of reference entity k measured in the given loss unit. In this case, we say the LGDs of all reference entities sit on a common lattice. If all LGDs of the reference entities in underlying pool are the same, then we call this pool a homogeneous pool; otherwise we call it an inhomogeneous pool. Among these exact methods are the ASB method, a recursive method, proposed by Andersen, Sidenius and Basu [2], the LG method, a fast Fourier transformation (FFT) based convolution method, by Laurent and Gregory [39], and the HW method, another recursive method, by Hull and White [30]. Note that both the ASB and the LG methods are directly applicable to inhomogeneous pools. Although the HW method is directly applicable to homogeneous pools only, it can be applied indirectly to inhomogeneous pools by noting that in practice an inhomogeneous pool can usually be partitioned into a few homogeneous subpools to which the HW method can be applied. Then the results for the homogeneous subpools can be combined to price the full inhomogeneous pool. Experiments show that the ASB method is generally faster than the LG method, while the HW method is faster than the ASB method for homogeneous or low-dispersion inhomogeneous pools (*i.e.*, pools having a few homogeneous subpools only). However, a naive implementation of the HW method can suffer from numerical stability problems due to overflow and cancellation in floating-point operations.

The second class of methods evaluates a pool's loss distribution approximately. De Prisco, Iscoe and Kreinin's compound Poisson approximation method [14], which also requires that the LGDs sit on a common lattice, is an example of a method of this class. It is shown by the authors that this method usually gives reasonably accurate results. However, the error in the approximated loss distribution may result in an error in the spread of as much as 20 basis points for an equity tranche. Therefore the accuracy of this approximation is not always satisfactory.

Besides the exponential approximation based method developed in Chapter 3 of this thesis, the only method that focuses on $\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right]$ directly is the so-called saddle-point approximation (SPA) method proposed by Antonov, Mechkov and Misirpashaev [3] and Yang, Hurd and Zhang [59] independently. Note that

$$f(\mathscr{L}^P;\ell,u) = \min\left(S, \max\left(\mathscr{L}^P - \ell, 0\right)\right) = \max\left(\mathscr{L}^P - \ell, 0\right) - \max\left(\mathscr{L}^P - u, 0\right),$$

and

$$\mathbb{E}\left[\max\left(\mathscr{L}^P - \Delta, 0\right)\right] = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \frac{\exp\left(\Gamma(\zeta) - \zeta\Delta\right)}{\zeta^2} \mathrm{d}\zeta, \tag{1.13}$$

where $c > 0$ and $\Gamma(\zeta) = \sum_{k=1}^{K} \ln\left(1 - Q_k + Q_k \exp(LGD_k\zeta)\right)$. In a SPA method, the integrand $\frac{\exp(\Gamma(\zeta)-\zeta\Delta)}{\zeta^2}$ is expanded at a fixed point, then the series is truncated and finally the truncated series is integrated to give an analytic approximation to the integral (1.13). Antonov, Mechkov and Misirpashaev expand the integrand at ζ^*_{AMM}, the solution of

$$[\Gamma(\zeta) - \zeta\Delta]' = 0;$$

while Yang, Hurd and Zhang expand the integrand at ζ^*_{YHZ}, the solution of

$$[\Gamma(\zeta) - \zeta\Delta - 2\ln\zeta]' = 0.$$

A more complete discussion of these two methods can be found in [3] and [59].

1.5 Contributions of this thesis

For pricing and hedging of synthetic CDO tranches, accuracy is generally more important than efficiency. While for risk management of synthetic CDO tranches, efficiency is equally important, because a synthetic CDO tranche must be priced thousands of times to generate a reasonable risk assessment. In this thesis we propose four efficient and accurate numerical methods for estimating $\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right]$. The one-factor Gaussian copula model is used throughout this thesis to illustrate these numerical methods. However, we reemphasize that all four numerical methods developed in this thesis are directly applicable to other more general models, such as the double t–copula model proposed by Hull and White [30], provided that the model belongs to the conditional independence framework.

In Chapter 2, which is based largely on the results in [36], we first propose a stable recursive method for computing the pool's loss distribution exactly. Then we introduce an improved compound Poisson approximation to approximate the pool's loss distribution. Both methods are based on the assumption that the LGDs sit on a common lattice. Finally, we introduce a normal power approximation method that has been used in actuarial science. Numerical results based on these three and some previously known methods are given.

In Chapter 3, which is based largely on the results in [32], we express the tranche loss function f as a simple expression involving two bases functions. Each of the two bases functions is a transformation of the *hockey stick* function $h(x)$, where $h(x) = 1 - x$ if $0 \leq x < 1$ and 0 if $x \geq 1$. By approximating the hockey stick function $h(x)$ by a sum of exponentials, the tranche loss function is approximated by a sum of exponentials. In this way, the estimation of the expected value of the tranche loss function is reduced to the estimation of a series of expected values of the individual reference entities in the underlying pool. The main advantage of this method is that the distribution of the pool loss need not be estimated. Numerical results based on our new method are reported.

A crucial part of this new method is the determination of the coefficients of an exponential approximation to the hockey stick function. In Chapter 4, which is based largely on the results in [31], we develop a numerical method to compute the coefficients of an exponential approximation to the hockey stick function. The theoretical properties of the exponential approximation to the hockey stick function are studied.

The thesis ends in Chapter 5 with conclusions and discussions.

Chapter 2

Loss distribution evaluation

In this chapter, which is based largely on the results in [36], we consider numerical methods for evaluating the distribution of $\mathscr{L}^P$ conditional on a given abscissa x_m at a fixed time t_i. In Section 2.1 we present an example to show that the HW method is not stable. In Sections 2.2 and 2.3 we propose three numerical methods for evaluating the pool's loss distribution. We first propose a stable recursive method for computing the exact pool loss distribution. Then introduce an improved compound Poisson approximation to approximate the pool loss distribution. Finally, we introduce a normal power approximation method that has been used in actuarial science to approximate the pool loss distribution. Numerical results based on these three and some known methods for synthetic CDO tranche valuation are given in Section 2.4.

2.1 Introduction

In this section we present an example to show that the HW method is not stable, thus is not reliable for synthetic CDO tranche valuation. Recall that $\mathscr{L}^P = \sum_{k=1}^{K} LGD_k \mathbf{1}_{\{k\}}$, and $\mathbf{1}_{\{k\}}$ are mutually independent conditional on a given value x_m of X. Let $w_k = \frac{Q_k}{S_k}$. For a homogeneous pool, the HW method computes the distribution of $\mathscr{L}^P$ in the following way.

The probability of exactly l defaults in the underlying pool is

$$p_{K,l} = p_{K,0} \sum w_{\imath(1)} w_{\imath(2)} \cdots w_{\imath(l)},$$

where $p_{K,0} = \prod_{k=1}^{K} S_k$ is the probability of no default, $\{\imath(1), \imath(2), \ldots, \imath(l)\}$ is a subset of l different integers chosen from $\{1, 2, \ldots, K\}$ and the summation is taken over the $\frac{K!}{l!(K-l)!}$ different ways in which the subset can be chosen. The summation is calculated using the so-called Newton-Girard recursive formulas, which can be found, for example, at `http://mathworld.wolfram.com/Newton-GirardFormulas.html`. However, this approach may cause numerical problems. More specifically, $p_{K,0} = \prod_{k=1}^{K} S_k$ may be too small and $\prod_{k=1}^{K} w_k$ may be too large to be represented correctly in a floating-point number system. For example, for $K = 100$ and $S_k = 1.0e\text{-}5$ for all k, we have $\prod_{k=1}^{K} S_k = 1.0e\text{-}500$ and $\prod_{k=1}^{K} w_k \approx 1.0e500$. In a floating-point number system, the first number usually underflows to zero and the second one overflows. Thus numerical problems arise.

2.2 Recursive method for loss distribution evaluation

In this section, we propose a stable recursive method for computing the exact pool loss distribution. The proposed method is stable from the numerical point of view and is at least as fast as the HW method. Note that, in practice, the loss-given-defaults of the referred entities are not necessarily the same; thus the underlying pool is not necessarily a homogeneous pool. However, the reference entities in the pool can usually be divided into a small number of sub-pools for which all reference entities in a sub-pool have the same loss-given-default. Therefore, an important basic problem is how to evaluate the loss distribution for a homogeneous pool.

2.2.1 Loss distribution for a homogeneous pool

For a homogeneous pool the problem reduces to computing the distribution of the random variable $\mathbf{1}_{\{\mathscr{L}^P\}} = \sum_{k=1}^{K} \mathbf{1}_{\{k\}}$, noting that without loss of generality we can assume that the common $LGD_k = 1$. Assume that the loss distribution of a homogeneous sub-pool of k names, $1 \le k < K$, is already known. Let $\mathbf{p}_k = (p_{k,k}, p_{k,k-1}, \ldots, p_{k,0})^T$, where $p_{k,j} = \mathbb{P}\left(\mathbf{1}_{\{\mathscr{L}^P\}}(k) = j\right)$, and $\mathbf{1}_{\{\mathscr{L}^P\}}(k) = \sum_{i=1}^{k} \mathbf{1}_{\{i\}}$. Then the loss distribution of the pool consisting of the first k names plus the $(k+1)$-st name with default probability Q_{k+1} can be calculated using the recursive formula

$$\mathbf{p}_{k+1} = \begin{pmatrix} p_{k+1,k+1} \\ p_{k+1,k} \\ \vdots \\ p_{k+1,1} \\ p_{k+1,0} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_k & 0 \\ 0 & \mathbf{p}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ S_{k+1} \end{pmatrix}. \tag{2.1}$$

In this way, $\mathbf{p}_K$ can be computed after $K-1$ iterations, starting from the initial value $\mathbf{p}_1 = (p_{1,1}, p_{1,0})^T = (Q_1, S_1)^T$.

We claim that the method based on formula (2.1) is numerically stable. Let ε be the *machine epsilon* (see Golub and Van Loan [22] or Heath [25] for the definition) for a floating-point system. Assume that the input probabilities Q_k are exactly representable in the floating-point number system[1]. Then the floating-point approximation to S_k is $\hat{S}_k = S_k + \delta_k S_k$, where $|\delta_k| \le \varepsilon$. Let $\epsilon_k = \mathbf{p}_k - \hat{\mathbf{p}}_k$, $k = 1, 2, \ldots, K$, where $\hat{\mathbf{p}}_k$ is the loss distribution evaluated using formula (2.1) in a floating-point number system:

$$\hat{\mathbf{p}}_{k+1} = fl\left(\begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} \right), \quad \hat{\mathbf{p}}_1 = \begin{pmatrix} Q_1 \\ \hat{S}_1 \end{pmatrix}. \tag{2.2}$$

Proposition 1 *The error vector ϵ_k satisfies*

$$\|\epsilon_k\|_\infty \le \left(1.001^{k-1} c - 3001\right) \varepsilon, \text{ for } k = 1, 2, \ldots, K, \tag{2.3}$$

[1]Note that a claim similar to Proposition 1, but somewhat more complicated, holds without this assumption.

where $\|\cdot\|_\infty$ *is the max norm of a vector and* $c = 3002$*, provided that* $4\varepsilon \le 0.001$*.*

Proof Note that for any nonnegative l-vector $\mathbf{a} = (a_1, a_2, \ldots, a_l)^T$ and nonnegative constants c and d, which are all exactly representable in a floating-point number system, we have

$$\begin{aligned}
&\left\| \begin{pmatrix} \mathbf{a} & 0 \\ 0 & \mathbf{a} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} - fl\left(\begin{pmatrix} \mathbf{a} & 0 \\ 0 & \mathbf{a} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} \right) \right\|_\infty \\
\le\ & (2\varepsilon + \varepsilon^2) \left\| \begin{pmatrix} \mathbf{a} & 0 \\ 0 & \mathbf{a} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} \right\|_\infty \\
\le\ & (2\varepsilon + \varepsilon^2)(c+d)\|\mathbf{a}\|_\infty . \qquad (2.4)
\end{aligned}$$

Applying (2.4) to (2.2) results in

$$\begin{aligned}
&\left\| \begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} - \hat{\mathbf{p}}_{k+1} \right\|_\infty \\
\le\ & (2\varepsilon + \varepsilon^2)\left(Q_{k+1} + \hat{S}_{k+1}\right)\|\hat{\mathbf{p}}_k\|_\infty \\
\le\ & (2\varepsilon + \varepsilon^2)(1+\varepsilon)\|\hat{\mathbf{p}}_k\|_\infty . \qquad (2.5)
\end{aligned}$$

Noting that when $4\varepsilon \le 0.001$, we have

$$(2\varepsilon + \varepsilon^2)(1+\varepsilon) \le 2.001\varepsilon.$$

Thus (2.5) can be written as

$$\left\| \begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} - \hat{\mathbf{p}}_{k+1} \right\|_\infty \le 2.001\varepsilon \|\hat{\mathbf{p}}_k\|_\infty . \qquad (2.6)$$

On the other hand, using the triangle inequality we have

$$\begin{aligned}
&\left\| \begin{pmatrix} \mathbf{p}_k & 0 \\ 0 & \mathbf{p}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ S_{k+1} \end{pmatrix} - \begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} \right\|_\infty \\
\le\ & \left\| \begin{pmatrix} \mathbf{p}_k & 0 \\ 0 & \mathbf{p}_k \end{pmatrix} \right\|_\infty \left\| \begin{pmatrix} 0 \\ \varepsilon S_{k+1} \end{pmatrix} \right\|_\infty + \left\| \begin{pmatrix} \mathbf{p}_k - \hat{\mathbf{p}}_k & 0 \\ 0 & \mathbf{p}_k - \hat{\mathbf{p}}_k \end{pmatrix} \right\|_\infty \left\| \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} \right\|_\infty \\
\le\ & \varepsilon\|\mathbf{p}_k\|_\infty + (1+\varepsilon)\|\epsilon_k\|_\infty . \qquad (2.7)
\end{aligned}$$

Noting that

$$\|\hat{\mathbf{p}}_k\|_\infty \leq \|\mathbf{p}_k\|_\infty + \|\epsilon_k\|_\infty \leq 1 + \|\epsilon_k\|_\infty,$$

where we used $\|\mathbf{p}_k\|_\infty \leq 1$, from (2.6) and (2.7) we have

$$\begin{aligned} \|\epsilon_{k+1}\|_\infty &\leq \varepsilon\|\mathbf{p}_k\|_\infty + (1+\varepsilon)\|\epsilon_k\|_\infty + 2.001\varepsilon\,\|\hat{\mathbf{p}}_k\|_\infty \\ &\leq \varepsilon\|\mathbf{p}_k\|_\infty + (1+\varepsilon)\|\epsilon_k\|_\infty + 2.001\varepsilon + 2.001\varepsilon\|\epsilon_k\|_\infty \\ &\leq (1+3.001\varepsilon)\|\epsilon_k\|_\infty + 3.001\varepsilon \\ &\leq 1.001\|\epsilon_k\|_\infty + 3.001\varepsilon. \end{aligned}$$

Thus we have

$$\|\epsilon_{k+1}\|_\infty \leq 1.001\|\epsilon_k\|_\infty + 3.001\varepsilon, \text{ for } k = 1, 2, \ldots, K-1. \tag{2.8}$$

Noting that $\|\epsilon_1\|_\infty \leq \varepsilon$, (2.8) implies that

$$\|\epsilon_k\|_\infty \leq \left(1.001^{k-1}c - 3001\right)\varepsilon, \quad \text{for } k = 1, 2, \ldots, K,$$

where $c = 3002$. This upper bound is obtained by using the result that the solution to the linear recurrence equation $x_{n+1} = ax_n + b$, where $a \neq 1$, is $x_n = \frac{b}{1-a} + a^n c$ for $n \geq 1$, where $c = \frac{x_1(a-1)+b}{a(a-1)}$.

The proof is completed.

If it is assumed that both Q_k and S_k are exactly representable in the floating-point number system, then $\|\epsilon_1\|_\infty = 0$ and the error bound (2.8) for $\|\epsilon_{k+1}\|$ reduces to

$$\|\epsilon_{k+1}\|_\infty \leq 1.001\|\epsilon_k\|_\infty + 2.001\varepsilon, \text{ for } k = 1, 2, \ldots, K-1. \tag{2.9}$$

In this case, the relation (2.3) reduces to

$$\|\epsilon_k\|_\infty \leq \left(1.001^k c' - 2001\right)\varepsilon, \quad \text{for } k = 1, 2, \ldots, K, \tag{2.10}$$

where $c' = 2001/1.001$.

Inequalities (2.3) and (2.10) ensure that the recursive method based on formula (2.1) is numerically stable.

2.2.2 Loss distribution for an inhomogeneous pool

In this section we discuss how to compute the loss distribution of an inhomogeneous pool. Suppose that an inhomogeneous pool consists of I homogeneous sub-pools with loss-given-defaults $LGD_1, LGD_2, \ldots, LGD_I$ for the sub-pools sitting on a common lattice[2] and the loss distribution for the i-th sub-pool being $(p_{i,0}, \ldots, p_{i,d_i})$, $i = 1, 2, \ldots, I$, where d_i is the upper bound on the number of defaults in sub-pool i, which implies that the largest possible loss for this sub-pool is $d_i LGD_i$ units. To compute the loss distribution of this inhomogeneous pool, we use the following method. Suppose that the loss distribution of a pool consisting of the first i sub-pools has been determined, denoted by $(p_0^{(i)}, p_1^{(i)}, \ldots, p_{A_i}^{(i)})$, where $p_a^{(i)}$ is the probability of a units of losses in the pool that consists of the first i sub-pools, $a = 0, 1, \ldots, A_i$, $A_i = \sum_{j=1}^{i} d_j LGD_j$. Then the loss distribution of the bigger pool consisting of the first i sub-pools plus the $(i+1)$-st sub-pool is

$$p_a^{(i+1)} = \sum_{\substack{l \in \{0, \ldots, A_i\} \\ (a-l)/LGD_{i+1} \in \{0, \ldots, d_{i+1}\}}} p_l^{(i)} \cdot p_{i+1,(a-l)/LGD_{i+1}}$$

for $a = 0, 1, \ldots, A_{i+1} = A_i + d_{i+1} LGD_{i+1}$. To start the iteration, the loss distribution $(p_{1,0}, \ldots, p_{1,d_1})$ of the first sub-pool must be mapped to $(p_0^{(1)}, p_1^{(1)}, \ldots, p_{d_1 LGD_1}^{(1)})$:

$$p_a^{(1)} = \begin{cases} p_{1,a/LGD_1} & \text{if } a/LGD_1 \text{ is an integer;} \\ 0 & \text{otherwise,} \end{cases}$$

where $a = 0, 1, \ldots, A_1 = d_1 LGD_1$. After $I-1$ iterations, the loss distribution of the pool is computed. We call the method based on the one described in subsection 2.2.1 and the one outlined in this subsection, JKM.

It can be shown that JKM is equivalent to ASB [2] when the underlying pool is either

[2]By this we mean that $LGD_1, LGD_2, \ldots, LGD_I$ are integer multiples of some properly chosen monetary unit.

homogeneous or completely inhomogeneous[3]. Since JKM exploits the property that the pool can usually be divided into a small number of homogeneous sub-pools, we expect it to be faster than the ASB method. Performance comparisons shown in Section 2.4 support this conjecture.

2.2.3 Generalization to multiple states

In the previous section we proposed a recursive method for computing the distribution of a sum of mutually independent random variables that follow Bernoulli distributions[4]. This method can be generalized to computing the distribution of a sum of multi-value random variables in the following way. Suppose that, for each name k, there is a random variable $\mathbf{1}_{\{k;M\}}$ that takes an integer value from $\{0, 1, \ldots, M-1\}$ with probability $Q_{k,m}$, where $\sum_{m=0}^{M-1} Q_{k,m} = 1$, and that $\mathbf{1}_{\{k;M\}}$ are mutually independent. In the context of pool loss distributions, we interpret $Q_{k,m}$ as the probability that entity k has a loss of m units. Then the probability distribution of the random variable $\mathbf{1}_{\{\mathscr{L}^P;M\}} = \sum_{k=1}^{K} \mathbf{1}_{\{k;M\}}$ can be computed using the recursive formula

$$\mathbf{p}_{k+1} = \begin{pmatrix} p_{k+1,(M-1)(k+1)} \\ \vdots \\ p_{k+1,1} \\ p_{k+1,0} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_k & 0 & \cdots & 0 \\ 0 & \mathbf{p}_k & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \mathbf{p}_k \end{pmatrix} \begin{pmatrix} Q_{k+1,M-1} \\ \vdots \\ Q_{k+1,1} \\ Q_{k+1,0} \end{pmatrix}, \tag{2.11}$$

where $p_{k,j}$ is the probability of the pool consisting of the first k names having a loss of j units, $\mathbf{p}_k = (p_{k,(M-1)k}, \ldots, p_{k,1}, p_{k,0})^T$ and $\mathbf{p}_1 = (Q_{1,M-1}, \ldots, Q_{1,1}, Q_{1,0})^T$. Assume that $Q_{k,m}$ are exactly representable in the computer system. Let $\epsilon_k = \mathbf{p}_k - \hat{\mathbf{p}}_k$, $k = 1, 2, \ldots, K$, where $\hat{\mathbf{p}}_k$ is the distribution evaluated using formula (2.11) in a floating-point number system. A result similar to Proposition 1 holds:

[3]By completely inhomogeneous we mean that $LGD_k \neq LGD_{k'}$ for all k and $k' \in \{1, 2, \ldots, K\}$ for which $k \neq k'$.

[4]A Bernoulli distribution is a discrete probability distribution, which takes value 1 with success probability p and value 0 with failure probability $q = 1 - p$.

Proposition 2 *The error associated with the recursive formula (2.11) satisfies*

$$\|\epsilon_k\|_\infty \le \left(1.001^k c' - (1000M+1)\right)\varepsilon, \quad \text{for } k = 1, 2, \ldots, K, \tag{2.12}$$

where $\|\cdot\|_\infty$ *is the max norm of a vector and* $c' = (1000M+1)/1.001$*, provided that* $(M+2)\varepsilon \le 0.001$*.*

Numerical results comparing the efficiency of this generalized recursive method with the FFT based convolution method [11, Chapter 32] are presented in Table 2.1. Both methods are coded in C++ and run on a Pentium III 700MHZ PC in the .NET environment. For the convolution, we used the convolution function `c06ekc` from the NAG C library [55], which is optimized for .NET. A divide-and-conquer technique was used to speed-up the implementation of the convolution method. The complexity of the divide-and-conquer convolution method is $O(MK \log_2 M \log_2 K)$. The complexity of our recursive method is $O(M^2K^2)$. Thus the FFT based convolution method should be asymptotically faster than our recursive method.

The experiment was carried out for several combinations of K and M. Entries of the form x:y in Table 2.1 represent the CPU time used by our generalized recursive method and the FFT based convolution method, respectively, to compute the loss distribution with the corresponding parameters K and M. For example, the entry `0.0006:0.0030` for $K = 200$ and $M = 2$ means that for these values of K and M the recursive and the FFT based convolution methods used 0.0006 seconds and 0.0030 seconds, respectively, to compute the loss distribution of the pool.

From Table 2.1 we can see that the recursive method is faster than the FFT based convolution method for practical problems, say when $M \le 16$ and $K \le 200$. However, the recursive method is slower than the FFT based convolution method when K or M is large, as is predicted by the complexities of the two methods. Curves in Figure 2.1 show the values of K and M for which the CPU time used by the two methods is almost the same. When (K, M) lies in the region above the curves, the FFT based

	M				
K	2	4	8	16	32
64	0.0001:0.0008	0.0004:0.0012	0.0006:0.0022	0.0026:0.0040	0.0122:0.0076
128	0.0004:0.0020	0.0008:0.0030	0.0028:0.0052	0.0112:0.0094	0.0495:0.0186
256	0.001:0.0044	0.0032:0.0064	0.011:0.0118	0.0439:0.0220	0.2077:0.0459
512	0.004:0.0101	0.0121:0.0140	0.0551:0.0271	0.2223:0.0531	0.9914:0.1092
100	0.0002:0.0014	0.0004:0.0026	0.0018:0.0046	0.0068:0.0086	0.0302:0.0170
200	0.0006:0.0030	0.002:0.0062	0.0068:0.0108	0.0268:0.0202	0.1208:0.0427
300	0.0012:0.0048	0.0042:0.0078	0.0152:0.0198	0.0635:0.0426	0.2976:0.0887
500	0.003:0.0081	0.012:0.0151	0.0511:0.0260	0.2103:0.0511	0.9424:0.1062

Table 2.1: CPU times (in seconds) for the generalized recursive method and the FFT based convolution method

convolution method is faster than the recursive method; in other cases, the recursive method is faster than the FFT based convolution method. The solid line in the bottom plot is a linear fit in the log-log scale to the experimental data. The equation for the line is $\log_2 M = -0.6869 \log_2 K + 8.5185$. For a given pair (K, M) one can decide, based on this equation, which method to choose. For example for $K = 200, M = 8$, which is represented by "o" in the two plots, we can see that the recursive method is faster.

2.3 Approximation methods for loss distribution evaluation

In Section 2.2 a stable recursive method was proposed. It is efficient if the underlying pool is homogeneous or it has a low dispersion in terms of LGDs (*i.e.*, the pool has a few homogeneous sub-pools only). For a high dispersion pool, approximation methods are

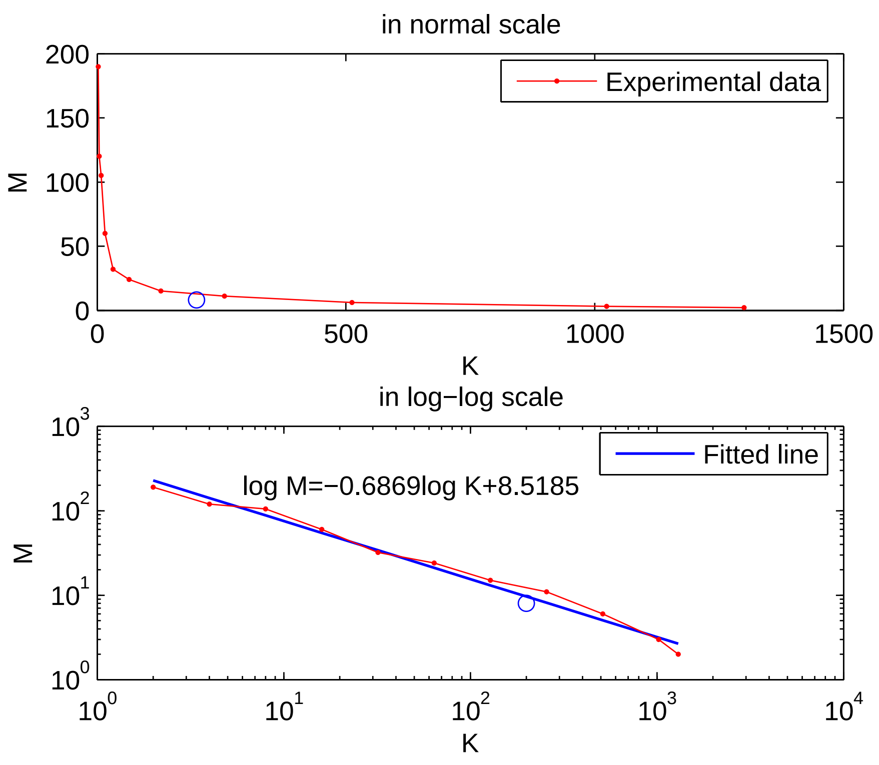

Figure 2.1: Comparison of computational speed between the recursive and the FFT based convolution methods

preferable. A method of this kind is the compound Poisson approximation introduced for synthetic CDO valuation by De Prisco, Iscoe and Kreinin [14], in which $\mathscr{L}^P$ is approximated by a random variable that follows a compound Poisson distribution. It is shown by the authors that this method usually gives reasonably accurate results. However, the error in the approximated loss distribution may result in an error in the spread of as much as 20 basis points for an equity tranche. Therefore the accuracy of this approximation is not always satisfactory. As a natural extension, we introduce in Section 2.3.1 an improved compound Poisson approximation method of Hipp [26] for better accuracy.

A random variable X is said to follow a compound Poisson distribution if its probability distribution function can be written in the form [5], [38]

$$\mu_x = \sum_{r=0}^{\infty} \exp(-\lambda)\frac{\lambda^r}{r!}\varphi^{*r}(x),$$

where $\lambda > 0$ is known as a Poisson parameter, $\varphi^{*r}(x)$ is the r-fold convolution of an associated common probability distribution function $\varphi(x)$. The characteristic function of X is

$$\phi_X(t) = \exp\left(\lambda(\psi(t) - 1)\right),$$

where $\psi(t)$ is the characteristic function of the distribution function associated with the probability distribution function $\varphi(x)$.

Note that both the recursive and the compound Poisson approximations require that the LGDs must sit on a common lattice. A small loss unit may result in a high dispersion pool, for which neither of these methods works well. To ameliorate this deficiency, we introduce in Section 2.3.2 a normal power approximation method to approximate the distribution of $\mathscr{L}^P = \sum_{k=1}^{K} LGD_k \mathbf{1}_{\{k\}}$.

2.3.1 Compound Poisson approximations

Instead of computing the distribution of $\mathscr{L}^P$ exactly, we can find an approximation to it through approximating its characteristic function and making use of the relationship between the characteristic function and the distribution function. The characteristic function, $\phi_k(t)$, of $LGD_k \mathbf{1}_{\{k\}}$ is

$$\phi_k(t) = 1 + Q_k\left(\exp(it \cdot LGD_k) - 1\right) = 1 + Q_k\left(g_k(t) - 1\right) = \exp\left(\ln\left(1 + Q_k(g_k(t) - 1)\right)\right),$$

where $g_k(t) = \exp(it \cdot LGD_k)$. Note that $LGD_k \mathbf{1}_{\{k\}}$ are conditionally mutually independent. Thus, the characteristic function, $\phi_{\mathscr{L}^P}(t)$, of $\mathscr{L}^P$ can be written in product form as

$$\phi_{\mathscr{L}^P}(t) = \prod_{k=1}^{K} \phi_k(t).$$

When $|x|$ is small, $\sum_{j=1}^{J} \frac{(-1)^{j+1}}{j} x^j$ gives a good approximation to $\ln(1 + x)$ even for a small J. Thus, it is expected that

$$\phi_k^{(J)}(t) = \exp\left(\sum_{j=1}^{J} \frac{(-1)^{j+1}}{j} \left[Q_k\left(g_k(t) - 1\right)\right]^j\right) \tag{2.13}$$

will be a good approximation to $\phi_k(t)$ if Q_k is small. Based on this approximation, the characteristic function $\phi_{\mathscr{L}^P}(t)$ is approximated by

$$\phi^{(J)}_{\mathscr{L}^P}(t) = \prod_k \phi^{(J)}_k(t) = \exp\left(\sum_{k=1}^K \sum_{j=1}^J \frac{(-1)^{j+1}}{j}\left[Q_k(g_k(t)-1)\right]^j\right).$$

Choosing $J = 1$ we obtain the first order approximation to the original characteristic function:

$$\phi_{\mathscr{L}^P}(t) \approx \phi^{(1)}_{\mathscr{L}^P}(t) = \exp\left(\sum_{k=1}^K Q_k(g_k(t)-1)\right) = \exp\left[\left(\sum_{m=1}^K Q_m\right)\left(\sum_{k=1}^K \frac{Q_k}{\sum_{m=1}^K Q_m} g_k(t) - 1\right)\right].$$

Let $\lambda_1 = \sum_{k=1}^K Q_k, \psi_1 = \sum_{k=1}^K \frac{Q_k}{\lambda_1} g_k(t)$, then

$$\phi^{(1)}_{\mathscr{L}^P}(t) = \exp\left(\lambda_1(\psi_1(t)-1)\right),$$

which is the characteristic function of a compound Poisson distributed random variable with Poisson parameter λ_1 and common distribution function

$$\varphi_1(L) = \sum_{LGD_k = L} \frac{Q_k}{\lambda_1}.$$

Thus, the distribution function of $\mathscr{L}^P$ is approximated by

$$\mu_{\mathscr{L}^P} = \exp(-\lambda_1)\sum_{r=0}^{\infty} \frac{\lambda_1^r}{r!}\varphi_1^{*r}, \tag{2.14}$$

where φ_1^{*r} is the r-fold self-convolution of φ_1 defined by a) $\varphi_1^{*0} = (1, 0, 0, \ldots, 0)$ of length $\sum_{k=1}^K LGD_k$, and b) $\varphi_1^{*(r+1)} = \varphi_1^{*r} * \varphi_1$. This approximation is also obtained in [14]. We denote it by CPA1 in this thesis.

By choosing $J > 1$ in (2.13), we might expect to improve the approximation to $\phi_{\mathscr{L}^P}(t)$. For $J = 2$, we obtain a compound Poisson approximation with Poisson parameter λ_2 and common distribution function φ_2 defined by

$$\lambda_2 = \sum_{k=1}^K \left(Q_k + \frac{Q_k^2}{2}\right),$$

$$\varphi_2(L) = \frac{1}{\lambda_2}\left[\sum_{LGD_k = L}(Q_k + Q_k^2) - \frac{1}{2}\sum_{2LGD_k = L} Q_k^2\right].$$

Similarly, for $J = 3$, the corresponding Poisson parameter λ_3 and the common distribution function φ_3 are

$$\lambda_3 = \sum_{k=1}^{K} \left(Q_k + \frac{Q_k^2}{2} + \frac{Q_k^3}{3} \right),$$

$$\varphi_3(L) = \frac{1}{\lambda_3} \left[\sum_{LGD_k = L} (Q_k + Q_k^2 + Q_k^3) - \sum_{2LGD_k = L} \left(\frac{Q_k^2}{2} + Q_k^3 \right) + \frac{1}{3} \sum_{3LGD_k = L} Q_k^3 \right].$$

For these two approximations, the distribution function for $\mathscr{L}^P$ is approximated similarly to (2.14) except that λ_1 and φ_1 are replaced by λ_J and φ_J for $J = 2$ or $J = 3$, respectively. The improved compound Poisson approximations corresponding to $J = 2$ and 3 are called CPA2 and CPA3, respectively, in this thesis.

Note that the compound Poisson approximation CPA1 matches the first moment of the true distribution; CPA2 matches the first two moments of the true distribution; and CPA3 matches the first three moments [16]. Some theoretical error bounds for these compound Poisson approximations are given in [26] and [13]. However, either they are (1) easy to estimate but too pessimistic or (2) too complicated to be computed. So we do not make use of any error analysis results for the compound Poisson approximations in this thesis.

2.3.2 Normal power approximation

In actuarial science, the payoff function $f(\mathscr{L}^P; \ell, u)$ is associated with a special insurance policy, called a stop-loss policy [5]. In this context, the reinsurer pays that part of the total amount of claims $\mathscr{L}^P$ which exceeds a certain amount, say ℓ, with the additional constraint that the reinsurer's liability is limited to an amount $S = u - \ell$. A general form of the central limit theorem [20, Theorem 4 on page 263] implies that when K, the number of claims, becomes large, $\mathscr{L}^P$ converges to a normal distribution. However, if K is small or the actual distribution has a skew structure, then significant deviation from normality appears. The goal in developing the normal power (NP) approximation is to find a transformation $X = v(Y)$ to convert a normally distributed random variable Y into another variable X, which can be better fitted to the actual distribution, the distribution of $\mathscr{L}^P$ in our case. When the distribution of $\mathscr{L}^P$ is approximated by a NP distribution, $\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right]$ can be expressed in terms of the cumulative distribution function Φ and the probability density function ϕ of the standard normal distribution. In this thesis, we give the basic formulas only; more details can be found in [5], [38], [53].

With the loss distribution being approximated by the NP formula, the expected value of $L = f(\mathscr{L}^P;\ell,u)$ is then

$$\mathbb{E}\left[L\right] = \mathbb{E}_{SL}\left[\mathscr{L}^P;\ell,S\right] = \mathbb{E}_{SL}\left[\mathscr{L}^P;\ell+S\right] - \mathbb{E}_{SL}\left[\mathscr{L}^P;\ell\right], \tag{2.15}$$

where

$$\mathbb{E}_{SL}\left[\mathscr{L}^P;z\right] = (\mu - z)(1-\Phi(y_z)) + \sigma(1+\gamma y_z/6)\phi(y_z) \tag{2.16}$$

is the expected loss of a tranche with a zero attachment point and the loss capped by z. The subscript SL stands for "Stop Loss"; Φ and ϕ are the cumulative distribution function and the probability density function of the standard normal distribution, respectively; μ,σ and γ are the mean, standard deviation and the skewness, respectively, of the pool loss $\mathscr{L}^P = \sum_{k=1}^K LGD_k \mathbf{1}_{\{k\}}$ (recall that $\mathbf{1}_{\{k\}}$ are mutually independent conditional on X):

$$\begin{aligned} \mu &= \sum_{k=1}^K LGD_k Q_k, \\ \sigma &= \sqrt{\sum_{k=1}^K (LGD_k-\mu)^2 Q_k}, \\ \gamma &= \sum_{k=1}^K (LGD_k-\mu)^3 Q_k/\sigma^3; \end{aligned}$$

and

$$y_z = \nu_\gamma^{-1}\left(\frac{z-\mu}{\sigma}\right),$$

where

$$\nu_\gamma^{-1}(f) = \begin{cases} f - g(f^2-1) + g^2(4f^3-7f)\cdot H(f_0-f) & \text{if } f<1; \\ \left(1+\frac{1}{4g^2}+\frac{f}{g}\right)^{1/2} - \frac{1}{2g} & \text{otherwise,} \end{cases}$$

with $g = \gamma/6$, $f_0 = -\sqrt{7/4}$ and

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

is the Heaviside function.

The NP approximation matches the first three moments of the true distribution and also captures some other important properties of it, such as fat tails and asymmetry. In contrast, the normal approximation [49] matches the first two moments only of the true distribution. Thus, it is expected that the normal power approximation might approximate the true distribution better than the normal approximation. For reasons similar to those explained above for the compound Poisson approximation, we do not make use of any error analysis results for the normal power approximation.

2.4 Numerical Results I

In this section we present numerical results that illustrate the accuracy and CPU time of our new methods: the recursive method JKM, the improved compound Poisson approximations CPA2 and CPA3 and the normal power approximation (NP). We also provide similar numerical results for the other three known methods: the HW method, the ASB method, and the compound Poisson approximation method CPA1.

2.4.1 Two points about the implementations

We should mention two points about the implementations of the proposed methods. The first point concerns a truncation technique used for the loss distribution evaluation. Suppose there are m tranches in a CDO. Note that, once the expected losses of the first $m - 1$ tranches, starting from the equity tranche, are available, the expected loss of the last tranche can be evaluated by

$$\mathbb{E}[\text{loss of last tranche}] = \sum_{k=1}^{K} LGD_k Q_k - \sum_{i=1}^{m-1} \mathbb{E}[\text{loss of tranche } i].$$

Thus in the remainder of this thesis, all test results are based on $m-1$, rather than m tranches. In particular, we use this result in the CPU time comparisons in this and the next chapter.

The second point concerns the stopping criterion for evaluating the infinite sum (2.14). In our implementation, the summation is stopped once the l_1-norm of the difference between the two distributions $\bar{\mu}^{(R)}_{\mathscr{L}^P}$ and $\bar{\mu}^{(R+1)}_{\mathscr{L}^P}$ is less than or equal to ϵ, where

$$\bar{\mu}^{(R)}_{\mathscr{L}^P} = \exp(-\lambda_J) \sum_{r=0}^{R} \frac{\lambda_J^r}{r!} \varphi_J^{*r},$$

for $J = 1, 2$, or 3. An alternative stopping criterion is based on the relative change of the accumulated distribution functions. In this case, the summation is stopped once

$$\frac{\|\bar{\mu}^{(R+1)}_{\mathscr{L}^P} - \bar{\mu}^{(R)}_{\mathscr{L}^P}\|_1}{\|\bar{\mu}^{(R)}_{\mathscr{L}^P}\|_1} \leq \epsilon,$$

where ϵ is a specified tolerance. These two criteria are approximately equivalent, since $\|\bar{\mu}^{(R)}_{\mathscr{L}^P}\|_1 \approx \|\mu_{\mathscr{L}^P}\|_1 = 1$. In our implementation we set $\epsilon = 10^{-4}$.

In our implementation, we used the Matlab function `conv` to compute the convolution of two vectors in the sum (2.14). As an alternative, one can use Panjer's recursive method to evaluate it [45].

2.4.2 Test problems

The results presented below are based on a sample of 15 pools. For each pool, the number of reference entities K is either 100, 200, or 400. The number of homogeneous sub-pools in each pool is either $1, 2, 4, 5$, or $K/10$, and all homogeneous sub-pools in a given pool have an equal number of reference entities. The notional values for each pool are summarized in Table 2.2. For example, the 200-reference-entity pool with local pool

ID = 3 consists of four homogeneous sub-pools with the notional values of 50, 100, 150, and 200, respectively. For convenience, we also labeled each pool with a global pool ID. For each of the 100-reference-entity pools, the global and the local IDs coincide. For each of the 200- and 400-reference-entity pools, its global pool ID (GID) is its local pool ID plus 5 or 10, respectively. For example, a 200-reference-entity pool with local ID = 3 has GID = 8.

Local Pool ID	1	2	3	4	5
Notional values	100	50, 100	50, 100, 150, 200	20, 50, 100, 150, 200	$10, 20, \ldots, K$

Table 2.2: Selection of notional values of K-reference-entity pools

For each reference entity, the risk-neutral cumulative default probabilities are randomly assigned one of two types, I or II, as defined in Table 2.3.

Type	1 yr.	2 yrs.	3 yrs.	4 yrs.	5 yrs.
I	0.0007	0.0030	0.0068	0.0119	0.0182
II	0.0044	0.0102	0.0175	0.0266	0.0372

Table 2.3: Risk-neutral cumulative default probabilities

The recovery rate is assumed to be 40% for all reference entities. Thus the LGD of reference-entity k is $0.6N_k$. The maturity of a CDO deal is five years (*i.e.*, $T = 5$) and the premium dates are $t_i = i, i = 1, \ldots, 5$ years from today ($t_0 = 0$). The continuously compounded risk-free rates are $r_1 = 4.6\%, r_2 = 5\%, r_3 = 5.6\%, r_4 = 5.8\%$ and $r_5 = 6\%$. Thus the corresponding risk-free discount factors, defined by $d_i = \exp(-t_i r_i)$, are $0.9550, 0.9048, 0.8454, 0.7929$ and 0.7408, respectively. All CDO pools have five tranches that are determined by the attachment points (ℓ's) of the tranches. For this experiment, the five attachment points are: 0, 3%, 4%, 6.1% and 12.1%, respectively. The constants β_k lie in $[0.3, 0.5]$. In practice, the β_k's are known as tranche correlations and are taken as input to the model.

All methods for this experiment were coded in Matlab and the programs were run on a Pentium III 700 PC. The results presented in Tables 2.4 and 2.5 are based on the pricing of the first four tranches of each pool, as explained above.

2.4.3 Analysis of results

The accuracy results are presented in Table 2.4. Since CPA2 and CPA3 produce the same numerical results to the basis point level, so in the table, we use CPA2(3) to represent the spreads obtained from these two methods. The four numbers in each pair of brackets in the main part of the table are the spreads, in basis points, for the first four tranches of the corresponding pool. For example, $(2248, 928, 606, 248)$ are the spreads, evaluated by an exact method for the first four tranches of the 200-reference-entity homogeneous pool (GID=6). Since all exact methods produce the same set of spreads for each pool, we use "Exact" in the table to represent the spreads obtained from all the exact methods: ASB, HW and JKM. From the table we can see that CPA1 produces reasonably accurate spreads, though for most pools the spreads differ somewhat from those of the exact methods. For example, for the 100-reference-entity pool with GID=5, the spread difference is 21 basis points, or about 0.6%, for the equity tranche. Also from the table we can say that CPA2(3) produces very accurate results, except for the homogeneous pools with GID=6 and GID=11, where the spreads for the 4-th tranches are 7 and 14 basis points higher than the exact ones, respectively. Fortunately, for a homogeneous pool we can use our efficient recursive method JKM. The last two columns in the table illustrate that neither the normal power nor the normal approximation is suitable for high-spread tranche pricing. If accurate results are required, the exact methods, CPA2 and CPA3 are recommended.

The CPU times are presented in Table 2.5. Since CPA2 is generally faster than CPA3 and produces essentially the same result and NP requires almost the same CPU times

GID	Exact	CPA1	CPA2(3)	NP	Normal
1	(2168, 926, 617, 256)	(2159, 922, 614, 256)	(2168, 926, 617, 256)	(2200, 939, 616, 256)	(2230, 940, 615, 255)
2	(2142, 945, 616, 257)	(2133, 941, 613, 257)	(2142, 945, 616, 257)	(2186, 941, 618, 257)	(2223, 941, 617, 257)
3	(2128, 941, 619, 259)	(2119, 936, 616, 259)	(2128, 941, 619,259)	(2175, 942, 619, 259)	(2217, 943, 619, 258)
4	(2098, 943, 622, 262)	(2087, 937, 619, 261)	(2097, 943, 622, 262)	(2153, 945, 623, 262)	(2205, 946, 623, 261)
5	(3069, 1166, 639, 154)	(3048, 1157, 637, 157)	(3069, 1166, 639, 154)	(3117, 1168, 640, 155)	(3188, 1180, 642, 154)
6	(2248, 928, 606, 248)	(2244, 926, 604, 248)	(2248, 928, 606, 255)	(2261, 931, 606, 248)	(2272, 931, 605, 248)
7	(2238, 931, 606, 249)	(2233, 929, 605, 249)	(2238, 931, 606, 249)	(2252, 932, 607, 249)	(2267, 932, 607, 249)
8	(2229, 932, 607, 250)	(2224, 929, 606, 250)	(2229, 932, 607, 250)	(2246, 933, 608, 250)	(2262, 933, 607, 250)
9	(2213, 934, 609, 251)	(2206, 931, 608, 251)	(2213, 934, 609, 251)	(2233, 934, 609, 251)	(2254, 935, 609, 251)
10	(3350, 1172, 606, 127)	(3337, 1167, 605, 129)	(3350, 1172, 606, 127)	(3350, 1172, 606, 127)	(3391, 1177, 607, 126)
11	(2291, 926, 600, 244)	(2289, 925, 600, 245)	(2291, 926, 601, 258)	(2295, 927, 600, 244)	(2300, 927, 600, 244)
12	(2286, 927, 601, 245)	(2283, 926, 600, 245)	(2286, 927, 601, 245)	(2291, 927, 601, 245)	(2296, 927, 601, 245)
13	(2282, 927, 601, 245)	(2279, 926, 601, 245)	(2282, 927, 601, 245)	(2288, 928, 601, 245)	(2294, 928, 601, 245)
14	(2273, 928, 602, 246)	(2270, 927, 602, 246)	(2273, 928, 602, 246)	(2280, 928, 602, 246)	(2288, 928, 602, 246)
15	(3428, 1158, 591, 122)	(3420, 1155, 591, 123)	(3428, 1158, 591, 122)	(3432, 1158, 592, 122)	(3440, 1159, 592, 122)

Table 2.4: Accuracy comparison between the exact and the approximate methods

as the normal approximation, we list only the CPU times for each of HW, ASB, CPA1, CPA2 and NP divided by that of JKM. From the table we can see that for all tested pools JKM is always faster than HW and CPA2, and much faster than ASB. For most cases JKM is slightly faster than CPA1, but slower than NP. As expected, CPA1 is faster than CPA2.

Based on both the accuracy and the CPU time of each method, we suggest using either the recursive method JKM or the second order compound Poisson approximation method CPA2 for pricing, where accuracy is generally more important than CPU time, and NP for risk management, where CPU time is generally more important than accuracy.

2.5 Conclusions I

Two types of methods for the evaluation of the loss distribution of a synthetic CDO pool are introduced in this chapter. Error analysis and numerical results show that the proposed exact recursive method JKM is stable and efficient. It can be applied to synthetic CDO tranche pricing and risk management when the underlying pool is homogeneous or has a low dispersion of loss-given-defaults. For high dispersion pools, the second order compound Poisson approximation CPA2 is recommended for pricing where accuracy is generally more important than CPU time. The normal power approximation is useful for risk management where CPU time is as important as accuracy.

GID	HW/JKM	ASB/JKM	CPA1/JKM	CPA2/JKM	NP/JKM
1	1.24	4.26	1.39	1.52	1.98
2	1.44	4.13	1.56	1.56	1.90
3	1.51	3.67	1.23	1.37	1.60
4	1.97	3.50	1.36	1.71	1.30
5	1.75	2.56	0.99	1.11	1.08
6	1.34	5.56	1.35	1.42	1.30
7	1.41	5.34	1.30	1.45	1.21
8	1.48	4.93	1.24	1.35	0.98
9	1.89	4.29	1.30	1.84	0.71
10	2.12	2.36	0.74	1.02	0.39
11	1.37	5.66	1.04	1.13	0.66
12	1.36	5.38	1.01	1.19	0.58
13	1.33	4.91	0.97	1.14	0.50
14	2.38	4.54	1.11	2.31	0.27
15	3.07	2.02	0.56	1.01	0.08

Table 2.5: The CPU times for each of HW, ASB, CPA1, CPA2 and NP divided by that of JKM

Chapter 3

A new method for approximating the expected value of the tranche loss function

In Chapter 2 we proposed three numerical methods for evaluating the distribution of $\mathscr{L}^P$. In this chapter, which is based largely on the results in [32], we focus on the tranche loss function f itself and propose a new method for approximating the expected value of the tranche loss function. First we describe a general result and then apply it to the valuation of a synthetic CDO tranche.

3.1 Approximation of the expected value of the tranche loss function

First we give a general expression of the expected value of the tranche loss function in the conditional independence framework. A central problem in this framework is how to evaluate

$$\mathbb{E}[f(Z;\ell,u)] = \int_M \mathbb{E}_M\left[f\left(Z;\ell,u\right)\right] \mathrm{d}\Phi(M),$$

where $\Phi(M)$ is the distribution of an auxiliary factor $\mathscr{M}$ (which can be a scalar or a vector),

$$\mathbb{E}_M\left[f\left(Z;\ell,u\right)\right] \equiv \mathbb{E}\left[f\left(Z;\ell,u\right)|\mathscr{M}=M\right],$$

where $Z=\sum_{k=1}^{K} Z_k$ and $Z_k \geq 0$ are mutually independent random variables, conditional on $\mathscr{M}$. It is obvious that Z is nonnegative. We denote by Ψ_M the distribution of Z conditional on $\mathscr{M}=M$, so that

$$\mathbb{E}_M\left[f\left(Z;\ell,u\right)\right] = \int_z f\left(z;\ell,u\right) \mathrm{d}\Psi_M(z). \tag{3.1}$$

Due to the piecewise linearity of f, it is clear that once the distribution Ψ_M is obtained, the expected value $\int_z f\left(z;\ell,u\right)\mathrm{d}\Psi_M(z)$ can be readily computed. Most research has focused on how to evaluate the conditional distribution of Z given the conditional distributions of Z_k; all the methods proposed in Chapter 2 are of this type. *Those methods are generally superlinear in K in complexity. In this chapter, we propose a different type of method for which the computational complexity is linear in K.* More specifically, we focus on the tranche loss function f, instead of the distribution Ψ_M of Z.

The tranche loss function can be expressed simply in terms of two transformed hockey stick functions:

$$f(z;\ell,u) = \min\left(S, \max\left(z-\ell,0\right)\right) = u\left[1-h\left(\frac{z}{u}\right)\right] - \ell\left[1-h\left(\frac{z}{\ell}\right)\right], \tag{3.2}$$

where $z \geq 0, 0 \leq \ell \leq u, S=u-\ell$, the hockey stick function $h(x)$ is defined on $[0,\infty)$ by $h(x)=1-x$ if $0 \leq x < 1$ and 0 if $x \geq 1$. In particular,

$$f(z;0,u) = u\left[1-h\left(\frac{z}{u}\right)\right].$$

The basic idea of our new method is to approximate the hockey stick function by a sum of exponentials first, then to use (3.2) together with this approximation to the hockey stick function to approximate the expected value of the tranche loss function conditional on $\mathscr{M}$ without having to estimate the loss distribution. To see this key point more clearly, let

$$h(x) \approx \sum_{n=1}^{N} \omega_n \exp(\gamma_n x), \tag{3.3}$$

where ω_n and γ_n are complex numbers. Then from (3.2) we can see that $f(z;\ell,u)$ can be approximated by a sum of exponentials:

$$\begin{aligned} f(z;\ell,u) &\approx u\left[1-\sum_{n=1}^{N}\omega_n \exp\left(\gamma_n \frac{z}{u}\right)\right] - \ell\left[1-\sum_{n=1}^{N}\omega_n \exp\left(\gamma_n \frac{z}{\ell}\right)\right] \\ &= (u-\ell) - u\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{u} z\right) + \ell\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{\ell} z\right). \end{aligned} \tag{3.4}$$

Based on this expression, $\mathbb{E}_M\left[f\left(Z;\ell,u\right)\right]$ defined in (3.1) can be approximated as follows:

$$\begin{aligned} \mathbb{E}_M\left[f\left(Z;\ell,u\right)\right] &= \int_z f\left(z;\ell,u\right) \mathrm{d}\Psi_M(z) \\ &\approx \int_z \left[(u-\ell) - u\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{u} z\right) + \ell\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{\ell} z\right)\right] \mathrm{d}\Psi_M(z) \\ &= (u-\ell) - u\sum_{n=1}^{N}\omega_n \int_z \exp\left(\frac{\gamma_n}{u} z\right) \mathrm{d}\Psi_M(z) \\ &\quad + \ell\sum_{n=1}^{N}\omega_n \int_z \exp\left(\frac{\gamma_n}{\ell} z\right) \mathrm{d}\Psi_M(z) \\ &= (u-\ell) \\ &\quad - u\sum_{n=1}^{N}\omega_n \int_{z_1,\ldots,z_K} \prod_{k=1}^{K} \exp\left(\frac{\gamma_n}{u} z_k\right) \mathrm{d}\Psi_{M,1}(z_1)\cdots \mathrm{d}\Psi_{M,K}(z_K) \\ &\quad + \ell\sum_{n=1}^{N}\omega_n \int_{z_1,\ldots,z_K} \prod_{k=1}^{K} \exp\left(\frac{\gamma_n}{\ell} z_k\right) \mathrm{d}\Psi_{M,1}(z_1)\cdots \mathrm{d}\Psi_{M,K}(z_K) \\ &= (u-\ell) - u\sum_{n=1}^{N}\omega_n \prod_{k=1}^{K} \mathbb{E}_M\left[\exp\left(\frac{\gamma_n}{u} Z_k\right)\right] \\ &\quad + \ell\sum_{n=1}^{N}\omega_n \prod_{k=1}^{K} \mathbb{E}_M\left[\exp\left(\frac{\gamma_n}{\ell} Z_k\right)\right], \end{aligned} \tag{3.5}$$

where $\Psi_{M,k}$ is the distribution of Z_k, $\mathbb{E}_M\left[\exp\left(cZ_k\right)\right]$ is the expected value of $\exp\left(cZ_k\right)$, for $c = \frac{\gamma_n}{\ell}$ or $\frac{\gamma_n}{u}$, respectively. The last equality holds since Z_k, thus cZ_k, are mutually independent conditional on a given value of $\mathscr{M}$. In this way we can see that, to compute $\mathbb{E}_M\left[f\left(Z;\ell,u\right)\right]$, we need only to compute $\mathbb{E}_M\left[\exp\left(cZ_k\right)\right]$ for each individual reference entity. That is, unlike the methods discussed in Chapter 2, we do not need to compute the distribution of Z.

For this approach to be effective, the coefficients ω_n and γ_n should be computed in advance; the real part of each γ_n should be nonpositive so that all the expected values appearing in (3.5) exist; and the approximation error of $h(x)$ should be small. Listed in Table 3.1 are the coefficients ω_n and γ_n for a 25-term exponential approximation of $h(x)$ obtained using the method described in Chapter 4. The error of this approximation is about $1.0e^{-2}$. From the table we can see that the real part of each γ_n is negative.

ω_n	γ_n
$1.68011893244425e\text{-}4 \pm i3.16256620606362e\text{-}5$	$-5.68445124827402e\text{-}2 \pm i1.44721383274924e2$
$2.03509915629236e\text{-}4 \pm i5.97831532622499e\text{-}5$	$-1.72409284138836e\text{-}1 \pm i1.32287063405070e2$
$2.69268773468033e\text{-}4 \pm i1.01521815083745e\text{-}4$	$-3.50678415544522e\text{-}1 \pm i1.19842679719888e2$
$3.86957625111202e\text{-}4 \pm i1.70219565943991e\text{-}4$	$-5.98265620413281e\text{-}1 \pm i1.07384279916896e2$
$6.01922445804571e\text{-}4 \pm i2.94018119278507e\text{-}4$	$-9.25359017045512e\text{-}1 \pm i9.49083080391091e1$
$1.01492774367573e\text{-}3 \pm i5.39110359552288e\text{-}4$	$-1.34736406458070 \pm i8.24123917927866e1$
$1.87278393479967e\text{-}3 \pm i1.08500939908606e\text{-}3$	$-1.88780310243265 \pm i6.98971865534513e1$
$3.86259704539165e\text{-}3 \pm i2.52517420285526e\text{-}3$	$-2.58365332234670 \pm i5.73707505516726e1$
$9.17405883622480e\text{-}3 \pm i7.43804670289735e\text{-}3$	$-3.49564479197934 \pm i4.48598171108201e1$
$2.44937222818637e\text{-}2 \pm i3.18666903390892e\text{-}2$	$-4.72746093364059 \pm i3.24436130440587e1$
$7.57246141516951e\text{-}3 \pm i2.09501133836536e\text{-}1$	$-6.43667314900433 \pm i2.03730372133839e1$
3.81388701388286	-9.65184479672942
$-1.45652522408126 \pm i1.02985968459737e\text{-}1$	$-8.54272349552524 \pm i9.38007996918211$

Table 3.1: Coefficients ω_n and γ_n for a 25-term exponential approximation of $h(x)$

In the remainder of this chapter, all exponential approximations to the hockey stick function are obtained using the method described in Chapter 4. For a given approximation accuracy ϵ_h, the coefficients ω_n and γ_n for (3.3) need to be computed once only and the number of terms, N, required can be determined a priori. Roughly speaking we have

$$N \approx \frac{1}{4\epsilon_h}$$

as is discussed in more detail in Chapter 4.

If the sup-norm of the error in approximating the hockey stick by a sum of exponentials is ϵ_h, then the error for the approximation (3.4) is at most $(u+\ell)\epsilon_h$, since

$$\begin{aligned}
&\left|f(z;\ell,u) - \left((u-\ell) - u\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{u}z\right) + \ell\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{\ell}z\right)\right)\right| \\
=&\left|(u-\ell) - uh\left(\frac{z}{u}\right) + \ell h\left(\frac{z}{l}\right)\right. \\
&\left.- \left((u-\ell) - u\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{u}z\right) + \ell\sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{\ell}z\right)\right)\right| \\
\leq& u\left|h\left(\frac{z}{u}\right) - \sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{u}z\right)\right| + \ell\left|h\left(\frac{z}{l}\right) - \sum_{n=1}^{N}\omega_n \exp\left(\frac{\gamma_n}{\ell}z\right)\right| \\
\leq&(u+\ell)\epsilon_h.
\end{aligned}$$

Hence, the error for approximation (3.5) is at most

$$(u+\ell)\epsilon_h.$$

Though the number of terms N may be as large as 400, we believe rounding error is not a problem when calculating the summations in (3.5), due to moderate $|\omega_n|$ and the not so small approximation error ϵ_h.

It is shown in Chapter 4 that, if γ_n is real, then ω_n is also real, and if γ_i and γ_j are a complex conjugate pair, then the corresponding ω_i and ω_j are also a complex conjugate pair, and vice versa. The data presented in Table 3.1 has this property. Exploiting this property, we can simplify the summations in (3.5) by noting that the sum of the i-th and j-th terms equals twice the real part of either one of these two terms. The data also shows that the real part of each γ_n is strictly negative. This property guarantees that the exponential approximation (3.3) converges to zero as $x \to \infty$, and thus guarantees the existence of the conditional expectation $\mathbb{E}_M[\exp(cZ_k)]$. A more detailed discussion of the exponential approximation (3.3) is given in Chapter 4.

3.2 Application to synthetic CDO valuation

Letting $Z_k = LGD_k \mathbf{1}_{\{k\}}$, we see that synthetic CDO tranche valuation is a special case of the problem described in the previous section. More specifically, we have

$$\begin{aligned}
\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right] \approx & S - u\sum_{n=1}^{N}\omega_n\mathbb{E}\left[\exp\left(\frac{\gamma_n}{u}\sum_{k=1}^{K}LGD_k\mathbf{1}_{\{k\}}\right)\right] \\
& + \ell\sum_{n=1}^{N}\omega_n\mathbb{E}\left[\exp\left(\frac{\gamma_n}{\ell}\sum_{k=1}^{K}LGD_k\mathbf{1}_{\{k\}}\right)\right] \\
= & S - u\sum_{n=1}^{N}\omega_n\prod_{k=1}^{K}\mathbb{E}\left[\exp\left(\frac{\gamma_n}{u}LGD_k\mathbf{1}_{\{k\}}\right)\right] \\
& + \ell\sum_{n=1}^{N}\omega_n\prod_{k=1}^{K}\mathbb{E}\left[\exp\left(\frac{\gamma_n}{\ell}LGD_k\mathbf{1}_{\{k\}}\right)\right],
\end{aligned} \tag{3.6}$$

where $S = u - \ell$ and

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\frac{\gamma_n}{u}LGD_k\mathbf{1}_{\{k\}}\right)\right] &= Q_k\exp\left(\frac{\gamma_n}{u}LGD_k\right) + (1-Q_k), \\
\mathbb{E}\left[\exp\left(\frac{\gamma_n}{\ell}LGD_k\mathbf{1}_{\{k\}}\right)\right] &= Q_k\exp\left(\frac{\gamma_n}{\ell}LGD_k\right) + (1-Q_k).
\end{aligned}$$

3.3 Numerical results II

In this section we present numerical results comparing the accuracy and the CPU time for the exact method JKM, our new exponential-approximation method and the SPA method proposed by Yang, Hurd and Zhang [59]. The results presented below are for the same set of pools described in Section 2.4. For the numerical experiment, the exponential-approximation method was run with different numbers of terms: 25, 50, 100, 200, and 400. The SPA method was run with a correction term (the so-called second order method in [59]). Both the new method and the SPA method were coded in Matlab and run on a Pentium III 700 PC. The results are presented in Tables 3.2, 3.3, 3.4, and 3.5.

The accuracy comparison results for the three methods are presented in Tables 3.2 and 3.3. As in Section 2.4, the four numbers in each pair of brackets in the main part of the table are the spreads in basis points for the first four tranches of the corresponding pool. For example, $(2248.16, 927.59, 605.52, 248.31)$ are the spreads evaluated by the JKM method for the first four tranches of the 200-reference-entity homogeneous pool (with global pool ID GID $= 6$). The entries under "25-term" and "400-term" are the spreads evaluated using the exponential-approximation method with 25 and 400 terms, respectively. From Table 3.2 we can see that, as the number of terms of the exponential approximation increases, the accuracy of the spreads improves. To better illustrate the accuracy of our new approach, the relative errors in the spreads obtained using exponential approximations, with different numbers of terms, compared to the spreads computed by the exact JKM method are plotted in Figures 3.1 and 3.2. From Table 3.3 we can see that the SPA method also gives very accurate spreads, though not as accurate as our 100-term exponential-approximation method.

The CPU times used by the JKM method, the SPA method and the exponential-approximation method using different numbers of terms for the test pools are presented in Tables 3.4 and 3.5, respectively. In Table 3.4 the numbers under "First tranche" and "First four tranches" are the times in seconds used by the exact JKM method to evaluate the spread for the first tranche and the spreads for the first four tranches of each pool, respectively. In Table 3.5 the numbers under "First tranche" and "First four tranches" are the times in seconds used by the SPA method and the exponential-approximation method using 25, 50, 100, 200 and 400 terms to evaluate the spread for the first tranche and the spreads for the first four tranches of each pool, respectively. Note that the CPU time for the SPA method depends approximately on the number of names only. For the exponential-approximation method, the CPU time depends on the number of names and the number of terms in the exponential approximation only. It is interesting to note that, for a given pool, to evaluate any single tranche using either the SPA method or the exponential-approximation method takes about as much time as to evaluate any other tranche. On the other hand, for the exact method, estimating the spread for the j-th tranche takes as much time as estimating the spreads for the first j tranches. Its

GID	Exact	25-term	400-term
1	(2167.69, 925.62, 616.56, 255.67)	(2165.21, 930.60, 615.90, 255.66)	(2167.54, 925.88, 616.56, 255.67)
2	(2142.13, 945.03, 615.79, 257.43)	(2141.54, 943.15, 616.96, 257.50)	(2142.08, 944.94, 615.85, 257.42)
3	(2128.39, 941.00, 618.88, 258.75)	(2128.80, 940.35, 618.92, 258.89)	(2128.35, 941.05, 618.86, 258.75)
4	(2097.58, 942.75, 622.30, 261.58)	(2097.24, 943.30, 622.47, 261.79)	(2097.55, 942.78, 622.29, 261.58)
5	(3069.39, 1165.62, 638.87, 154.37)	(3069.45, 1165.84, 639.05 154.43)	(3069.35, 1165.65, 638.88, 154.37)
6	(2248.16, 927.59, 605.52, 248.31)	(2246.74, 930.51, 605.30, 248.64)	(2248.07, 927.72, 605.52, 248.31)
7	(2237.60, 931.25, 606.10, 249.15)	(2236.79, 931.45, 606.74, 249.46)	(2237.54, 931.26, 606.12, 249.15)
8	(2229.45, 931.73, 607.47, 249.80)	(2229.10, 932.34, 607.63, 250.12)	(2229.41, 931.78, 607.47, 249.80)
9	(2212.52, 933.62, 609.33, 251.27)	(2212.51, 933.93, 609.54, 251.56)	(2212.50, 933.64, 609.33, 251.26)
10	(3350.42, 1171.60, 605.99, 127.05)	(3350.44, 1172.09, 606.28, 127.10)	(3350.40, 1171.60, 606.00, 127.05)
11	(2291.12, 925.82, 600.30, 244.49)	(2290.57, 926.86, 600.67, 244.97)	(2291.07, 925.88, 600.30, 244.49)
12	(2285.92, 926.99, 600.81, 244.90)	(2285.72, 927.32, 601.22, 245.39)	(2285.89, 926.99, 600.81, 244.90)
13	(2281.84, 927.31, 601.32, 245.25)	(2281.82, 927.68, 601.66, 245.71)	(2281.82, 927.33, 601.32, 245.25)
14	(2273.15, 928.22, 602.32, 245.99)	(2273.27, 928.50, 602.63, 246.43)	(2273.14, 928.23, 602.32, 245.99)
15	(3427.70, 1157.67, 591.47, 122.31)	(3427.84, 1158.17, 591.77, 122.40)	(3427.70, 1157.67, 591.47, 122.31)

Table 3.2: Accuracy comparison between the exact JKM method and the exponential-approximation method using 25 and 400 terms

GID	Exact	100-term	SPA
1	(2167.69, 925.62, 616.56, 255.67)	(2167.06, 926.63, 616.59, 255.65)	(2163.39, 938.24, 615.22, 255.79)
2	(2142.13, 945.03, 615.79, 257.43)	(2141.94, 944.66, 616.03, 257.43)	(2142.81, 939.63, 617.30, 257.46)
3	(2128.39, 941.00, 618.88, 258.75)	(2128.25, 941.19, 618.85, 258.77)	(2128.16, 940.73, 618.94, 258.79)
4	(2097.58, 942.75, 622.30, 261.58)	(2097.46, 942.90, 622.26, 261.58)	(2096.66, 943.07, 622.45, 261.65)
5	(3069.39, 1165.62, 638.87, 154.37)	(3069.29, 1165.80, 638.80, 154.37)	(3067.94, 1165.69, 638.94, 154.53)
6	(2248.16, 927.59, 605.52, 248.31)	(2247.83, 928.13, 605.50, 248.28)	(2248.77, 930.88, 605.45, 248.30)
7	(2237.60, 931.25, 606.10, 249.15)	(2237.36, 931.27, 606.22, 249.15)	(2236.64, 931.59, 606.57, 249.15)
8	(2229.45, 931.73, 607.47, 249.80)	(2229.31, 931.97, 607.44, 249.79)	(2228.90, 932.26, 607.46, 249.83)
9	(2212.52, 933.62, 609.33, 251.27)	(2212.45, 933.68, 609.32, 251.29)	(2212.13, 933.71, 609.39, 251.30)
10	(3350.42, 1171.60, 605.99, 127.05)	(3350.37, 1171.63, 606.02, 127.05)	(3349.56, 1171.68, 606.11, 127.12)
11	(2291.12, 925.82, 600.30, 244.49)	(2290.93, 926.17, 600.24, 244.49)	(2290.61, 926.62, 600.28, 244.49)
12	(2285.92, 926.99, 600.81, 244.90)	(2285.80, 926.97, 600.84, 244.92)	(2285.62, 927.08, 600.87, 244.92)
13	(2281.84, 927.31, 601.32, 245.25)	(2281.77, 927.40, 601.32, 245.27)	(2281.64, 927.45, 601.33, 245.26)
14	(2273.15, 928.22, 602.32, 245.99)	(2273.12, 928.26, 602.34, 246.01)	(2272.98, 928.24, 602.35, 246.01)
15	(3427.70, 1157.67, 591.47, 122.31)	(3427.70, 1157.70, 591.48, 122.31)	(3427.32, 1157.71, 591.51, 122.34)

Table 3.3: Accuracy comparison of the exact method, the saddlepoint approximation method SPA and the exponential-approximation method using 100 terms

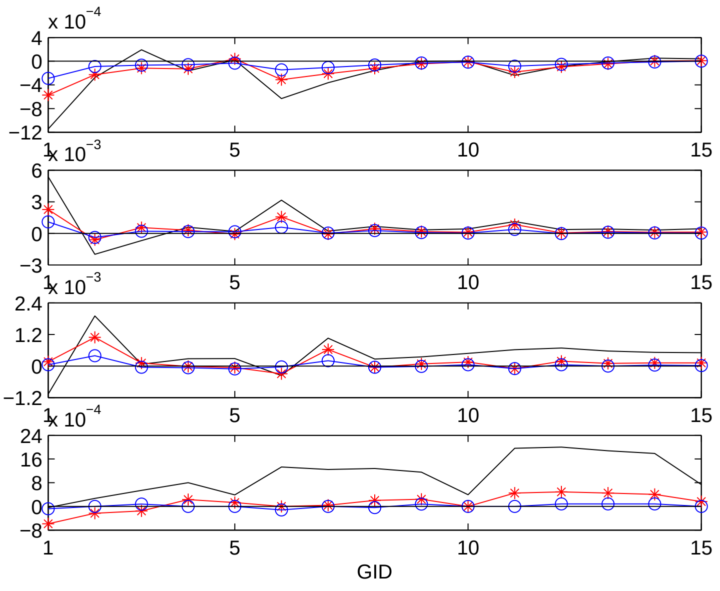

Figure 3.1: The graphs from top to bottom are the plots of the relative errors of the tranche spreads computed by our new method based on 25-, 50-, and 100-term exponential approximations compared to the exact spreads computed by the JKM method for the tranches $[0\%, 3\%]$, $[3\%, 4\%]$, $[4\%, 6.1\%]$, and $[6.1\%, 12.1\%]$, respectively. The solid line (black) is for the 25-term approximation. The line marked with small asterisks (red) is for the 50-term approximation. The line marked with small circles (blue) is for the 100-term approximation.

CPU time depends not only on the number of names but also on the structure of the underlying pool.

3.4 Conclusions II

A new method based on an exponential approximation to the "hockey stick" function has been proposed. Based on this approximation, the evaluation of the expected value of the tranche loss function of a specified tranche can be approximated by computing a series of expected values for individual reference entities. In Section 3.2, we applied

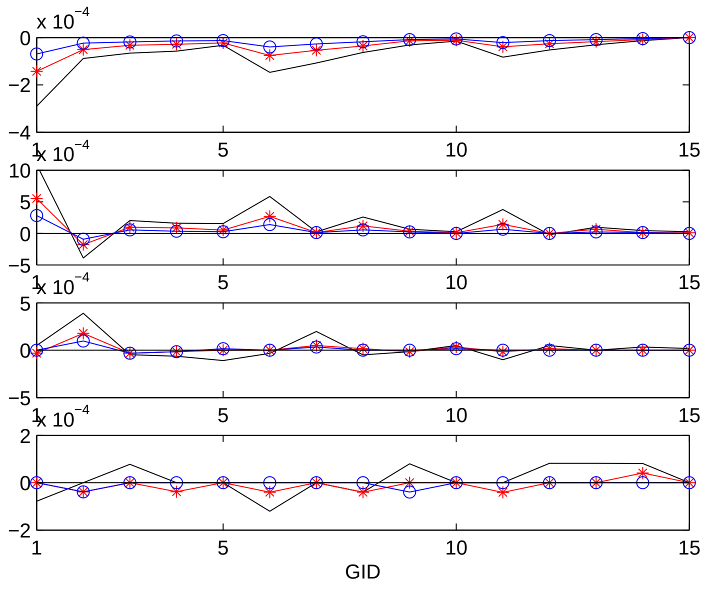

Figure 3.2: The graphs from top to bottom are the plots of the relative errors of the tranche spreads computed by our new method based on 100-, 200-, and 400-term exponential approximations compared to the exact spreads computed by the JKM method for the tranches $[0\%, 3\%]$, $[3\%, 4\%]$, $[4\%, 6.1\%]$, and $[6.1\%, 12.1\%]$, respectively. The solid line (black) is for the 100-term approximation. The line marked with small asterisks (red) is for the 200-term approximation. The line marked with small circles (blue) is for the 400-term approximation.

this method to synthetic CDO tranche valuation. This method could be applied to more general models provided that they belong to the conditional independence framework. Also our new method could be applied to a wide class of derivatives. For example, it can be applied to the pricing of options on spreads of a tranche of a synthetic CDO. Compared to the saddlepoint approximation method proposed by Antonov, Mechkov, and Misirpashaev [3] and Yang, Hurd and Zhang [59], the main advantage of our new approach is that the coefficients can be computed in advance, whereas the saddlepoint method must compute parameters dynamically.

GID	First tranche	First four tranches
1	0.39	0.46
2	0.44	0.48
3	0.52	0.57
4	0.57	0.70
5	0.81	0.85
6	0.53	0.71
7	0.58	0.76
8	0.67	0.88
9	0.76	1.26
10	1.41	2.32
11	0.86	1.41
12	0.95	1.56
13	1.06	1.86
14	1.32	3.38
15	4.50	12.31

Table 3.4: CPU time in seconds used by the JKM method to evaluate the first and the first four tranches of the test pools

	First tranche						First four tranches					
K/N	SPA	25	50	100	200	400	SPA	25	50	100	200	400
100	0.76	0.45	0.63	1.01	1.76	3.36	2.37	1.03	1.77	3.83	6.22	12
200	1.45	0.57	0.81	1.29	2.34	4.41	4.31	1.39	2.40	4.51	8.29	16.52
400	1.62	0.74	1.08	1.74	3.11	5.95	4.85	1.85	3.19	5.86	11.16	22.76

Table 3.5: CPU time in seconds used by the SPA method and the exponential-approximation method with different numbers of terms to evaluate the first and the first four tranches of the test pools

Chapter 4

Approximation of the hockey stick function

4.1 Introduction

In Chapter 3 we developed a new method to approximate $\mathbb{E}\left[f(\mathscr{L}^P;\ell,u)\right]$ based on an exponential approximation to the tranche loss function, which is expressed simply in terms of two transformed hockey stick functions. In this chapter, which is based largely on the results in [31], we describe how to approximate the hockey stick function

$$h(x) = \begin{cases} 1-x & \text{if } 0 \leq x < 1; \\ 0 & \text{if } x \geq 1, \end{cases} \tag{4.1}$$

by a sum of exponentials

$$h_{\exp}(x) = \sum_{n=1}^{N} \omega_n \exp(\gamma_n x) \tag{4.2}$$

over $[0,\infty)$, where ω_n and γ_n are complex numbers. The function (4.1) is a special case of the more general hockey stick function

$$h(x;t) = \begin{cases} t-x & \text{if } 0 \leq x < t; \\ 0 & \text{if } x \geq t, \end{cases}$$

where t is a positive number. This function plays a critical role in finance, from pricing of European options [28] to pricing and risk management of correlation-dependent derivatives [32]. Since, for a fixed positive t, $h(x;t) = t \cdot h(x/t)$, we can take $h(x)$ as the basic function. In this thesis we call function $h(x)$ the hockey stick (HS) function.

The approximation problem considered here is an example of Chebyshev approximation. For such an approximation, the weights ω_n and the exponents γ_n should be chosen to solve the minimization problem

$$\min_{\omega_n, \gamma_n \in \mathbb{C}} \left\| h(x) - \sum_{n=1}^{N} \omega_n \exp(\gamma_n x) \right\|_\infty , \tag{4.3}$$

where $\mathbb{C}$ denotes the set of complex numbers, and $\|f\|_\infty = \sup_{x \in \mathbb{X}} |f(x)|$ is the Chebyshev norm (also known as the uniform norm or the sup-norm) of f, and $\mathbb{X} = [a, b] \subset \mathbb{R}$. Theoretically, the existence of such an optimal approximation is generally not guaranteed [8, Chapters VI and VII]. Classic numerical methods for linear Chebyshev approximations, such as Remez exchange algorithm and its improvements, do not work well for finding best nonlinear Chebyshev approximations such as (4.3) [37]. Most algorithms for nonlinear Chebyshev approximations resort to solving discrete Chebyshev approximation subproblems. For exponential approximation problems, such a discrete Chebyshev approximation subproblem is equivalent to an exponential fitting problem, which is often badly-conditioned [21]. Consequently, we need to find some special methods that work well for (4.3). In this chapter, we apply the method recently proposed by Beylkin and Monzón [7] to determine the coefficients ω_n and γ_n in (4.3).

The remainder of this chapter is organized as follows. Beylkin and Monzón's method and its application to the HS function are discussed in Section 4.2. Properties of this exponential approximation are discussed in Section 4.3. This chapter ends with numerical results.

4.2 Beylkin and Monzón's method and its application to the HS function

4.2.1 Beylkin and Monzón's method

In a recent paper [7], Beylkin and Monzón proposed a numerical method to find a good exponential approximation to a function f. Instead of finding optimal ω_n and γ_n satisfying (4.3), their method finds such parameters so that the exponential approximation satisfies a given accuracy requirement. More specifically, for a given function f defined on $[0,1]$ and a given $\epsilon > 0$, their method seeks the minimal (or nearly minimal) number of complex weights ω_n and nodes $\exp(\gamma_n)$ such that

$$\left| f(x) - \sum_{n=1}^{N} \omega_n \exp(\gamma_n x) \right| \leq \epsilon, \qquad \forall x \in [0,1]. \tag{4.4}$$

This continuous problem is in turn approximated by a discrete problem: Given a positive integer $\mathcal{M}$, find the minimal positive integer number $N \leq \mathcal{M}$ of complex weights ω_n and complex nodes ζ_n such that

$$\left| f\left(\frac{m}{2\mathcal{M}}\right) - \sum_{n=1}^{N} \omega_n \zeta_n^m \right| \leq \epsilon, \qquad \text{for all integers } m \in [0, 2\mathcal{M}]. \tag{4.5}$$

Then for the continuous problem the weights and the exponents are ω_n and

$$\gamma_n = 2\mathcal{M} \log \zeta_n, \tag{4.6}$$

respectively, where $\log z$ is the principal value of the logarithm.

To describe their method, we introduce some additional notation. For theoretical background and a more detailed description of the method, see [7].

For a real $(2\mathcal{M}+1)$-vector $\mathbf{h} = (h_0, h_1, \ldots, h_{2\mathcal{M}})$, the $(\mathcal{M}+1) \times (\mathcal{M}+1)$ Hankel matrix $\mathbf{H}_\mathbf{h}$ defined in terms of $\mathbf{h}$ is

$$\mathbf{H}_\mathbf{h} = \begin{bmatrix} h_0 & h_1 & \cdots & h_\mathcal{M} \\ h_1 & \cdots & \cdots & h_{\mathcal{M}+1} \\ \vdots & & \iddots & \vdots \\ h_{\mathcal{M}-1} & h_\mathcal{M} & \cdots & h_{2\mathcal{M}-1} \\ h_\mathcal{M} & \cdots & h_{2\mathcal{M}-1} & h_{2\mathcal{M}} \end{bmatrix}.$$

That is, $(\mathbf{H}_\mathbf{h})_{i,j} = h_{i+j}$ for $0 \le i, j \le \mathcal{M}$. It is clear that $\mathbf{H}_\mathbf{h}$ is a real symmetric matrix. By the Corollary in §4.4.4 of [27, pp. 204], there exists a unitary matrix $\mathbf{U}$ and a *nonnegative* diagonal matrix Σ such that

$$\mathbf{H}_\mathbf{h} = \mathbf{U}\Sigma\mathbf{U}^T,$$

where the superscript T denotes transposition. This decomposition is called the Takagi factorization [27, pp. 204].

The main steps of the method are:

1. Sample the approximated function f at $2\mathcal{M}+1$ points uniformly distributed on $[0,1]$. That is, let $h_m = f\left(\frac{m}{2\mathcal{M}}\right)$, $0 \le m \le 2\mathcal{M}$.

2. Form $\mathbf{h} = (h_0, h_1, \ldots, h_{2\mathcal{M}})$ and the Hankel matrix $\mathbf{H}_\mathbf{h}$.

3. Compute the Takagi factorization of $\mathbf{H}_\mathbf{h} = \mathbf{U}\Sigma\mathbf{U}^T$, where $\Sigma = \operatorname{diag}(\sigma_0, \sigma_1, \ldots, \sigma_\mathcal{M})$ and $\sigma_0 \ge \sigma_1 \ge \ldots \ge \sigma_\mathcal{M} \ge 0$.

4. Find the largest σ_N satisfying $\sigma_N \le \epsilon$.

5. Let $\mathbf{u} = (u_0, u_1, \ldots, u_\mathcal{M})^T$ be the $(N+1)$-st column of $\mathbf{U}$.

6. Find N roots of the polynomial $\sum_{m=0}^{\mathcal{M}} u_m z^m$ with the largest moduli and denote these roots by $\zeta_1, \zeta_2, \ldots, \zeta_N$.

7. Compute the N weights ω_n, $1 \leq n \leq N$, by solving the linear least squares problem for the overdetermined Vandermonde system
$$h_m = \sum_{n=1}^{N} \omega_n \zeta_n^m, \qquad \text{for } 0 \leq m \leq 2\mathcal{M}.$$
8. Compute parameters γ_n using formula (4.6).

Remark 1 *This algorithm works for functions defined on* $[0,1]$. *To apply it to a function* f *defined on a finite interval* $[a,b]$, $a < b$, *we could consider the function* $\hat{f}(t) = f\left(t(b-a)+a\right)$ *for* $t \in [0,1]$. *For a function defined on an infinite interval, such as* $[0,\infty)$, *the interval could first be truncated to a finite interval, say* $[a,b] \subset [0,\infty)$, *then the finite interval could be mapped to the standard interval* $[0,1]$ *and the same approximation could be applied to* $[0,\infty)\backslash[a,b]$.

Remark 2 *For a general function the number of sample points is not known in advance. Thus* $\mathcal{M}$ *should be large enough or be increased gradually until a satisfactory accuracy is achieved. All critical points of the approximated function should be sampled. For example, for the HS function* $h(x)$, *both* $x = 0$ *and* $x = 1$ *should be sampled.*

Remark 3 *In practice it is not necessary to compute* $\mathbf{H_h}$*'s Takagi factorization explicitly. From the spectral theorem for Hermitian matrices [27, pp. 171] we know that there is a* **real** *orthogonal matrix* $\mathbf{V}$ *and a* **real** *diagonal matrix* $\Lambda = \operatorname{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{\mathcal{M}})$, *with* $|\lambda_i|$ *nonincreasing, such that* $\mathbf{H_h} = \mathbf{V}\Lambda\mathbf{V}^T$. *Noting that generally* $\mathbf{H_h}$ *is not positive semidefinite,* Λ *may have negative element(s). Thus* $\mathbf{V}\Lambda\mathbf{V}^T$ *is not necessarily the Takagi factorization of* $\mathbf{H_h}$. *However, we could construct a Takagi factorization based on its spectral factorization in the following way. Let* $\Sigma = \operatorname{diag}(|\lambda_0|, |\lambda_1|, \ldots, |\lambda_{\mathcal{M}}|)$ *and* $\mathbf{U} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{\mathcal{M}})$, *where* $\mathbf{u}_m = \mathbf{v}_m$ *if* $\lambda_m \geq 0$; *and* $\mathbf{u}_m = \sqrt{-1}\mathbf{v}_m$, *if* $\lambda_m < 0$. *It is easy to check that* $\mathbf{U}$ *is a unitary matrix and* $\mathbf{H_h} = \mathbf{U}\Sigma\mathbf{U}^T$.

Remark 4 *To compute* ω_n *from the linear least squares problem in Step 7, the* N *roots determined in Step 6 must be distinct. If this condition is not met,* ω_n *should be computed* *by a different method [7], [35]. This condition may be difficult to verify in theory. For numerical solutions, we should check its validity, as suggested by Beylkin and Monzón.*

4.2.2 Application to the HS function

In this subsection we apply Beylkin and Monzón's method to the hockey stick function $h(x)$. Recall that $h(x)$ is defined on $[0, \infty)$. The infinite interval is first truncated to a finite interval $[0, b]$ for a large enough b. (In fact $b = 2$ is large enough as explained below.) Then $h(b \cdot t)$ is sampled at $2\mathcal{M} + 1$ points:

$$h_m = h\left(bt_m\right) = \begin{cases} 1 - bt_m & \text{if } bt_m < 1 \\ 0 & \text{otherwise} \end{cases},$$

where $t_m = \frac{m}{2\mathcal{M}}$ and $0 \leq m \leq 2\mathcal{M}$. To guarantee the critical point $x = 1$ of $h(x)$ is sampled, it suffices that $bt_m = 1$ for some m. This implies that $\frac{2\mathcal{M}}{b}$ must be an integer. The corresponding Hankel matrix $\mathbf{H_h}$ is

$$\mathbf{H_h} = \left[\begin{array}{ccccc|ccc} 1 & 1 - \frac{b}{2\mathcal{M}} & 1 - 2\frac{b}{2\mathcal{M}} & \cdots & \frac{b}{2\mathcal{M}} & 0 & \cdots & 0 \\ 1 - \frac{b}{2\mathcal{M}} & 1 - 2\frac{b}{2\mathcal{M}} & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ 1 - 2\frac{b}{2\mathcal{M}} & \cdots & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & & & & & & \\ \frac{b}{2\mathcal{M}} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \hline \cdots & \cdots & & & & & & \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right].$$

To keep the neat form of $\mathbf{H_h}$ it may be required that $b \geq 2$. If $b < 2$, the last nonzero row of $\mathbf{H_h}$ may have more than one nonzero element. A direct consequence of this is that equation (4.8) may not hold, thus the properties of the approximation discussed in Section 4.3 may not hold. Thus in the remainder of this thesis it is assumed that $b \geq 2$.

Let $\mathcal{N} = \frac{2\mathcal{M}}{b}$ and

$$\mathbf{H}_{\mathcal{N}} = \begin{bmatrix} \mathcal{N} & \mathcal{N}-1 & \mathcal{N}-2 & \cdots & 1 \\ \mathcal{N}-1 & \mathcal{N}-2 & \cdots & \cdots & 0 \\ \mathcal{N}-2 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & & & \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{4.7}$$

Then we have

$$\mathbf{H}_{\mathbf{h}} = \frac{1}{\mathcal{N}} \left[\begin{array}{c:c} \mathbf{H}_{\mathcal{N}} & \mathbf{0}_{12} \\ \hdashline \mathbf{0}_{12}^T & \mathbf{0}_{22} \end{array}\right]. \tag{4.8}$$

where $\mathbf{0}_{12}$ and $\mathbf{0}_{22}$ are zero matrices of the proper dimensions.

Let $\mathbf{U}\Sigma\mathbf{U}^T$ be a Takagi factorization of $\mathbf{H}_{\mathcal{N}}$, where $\Sigma = \text{diag}(\sigma_{\mathcal{N}1}, \sigma_{\mathcal{N}2}, \ldots, \sigma_{\mathcal{N}\mathcal{N}})$ and $\sigma_{\mathcal{N}1} \geq \sigma_{\mathcal{N}2} \geq \cdots \geq \sigma_{\mathcal{N}\mathcal{N}} \geq 0$. Then a Takagi factorization of $\mathbf{H}_{\mathbf{h}}$ can be obtained by

$$\mathbf{H}_{\mathbf{h}} = \frac{1}{\mathcal{N}} \begin{bmatrix} \mathbf{U} & \mathbf{0}_{12} \\ \mathbf{0}_{12}^T & \mathbf{I}_{22} \end{bmatrix} \begin{bmatrix} \Sigma & \mathbf{0}_{12} \\ \mathbf{0}_{12}^T & \mathbf{0}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{U}^T & \mathbf{0}_{12}^T \\ \mathbf{0}_{12} & \mathbf{I}_{22} \end{bmatrix}.$$

Remark 5 *Proposition 4 in Section 4.3 together with Theorems 2 and 3 of [7] imply that, for a given accuracy ϵ, $\mathcal{M}$ must be large enough such that $\frac{1}{4}\frac{b}{2\mathcal{M}} \leq \epsilon$. From this relation and noting that $\mathcal{N} = \frac{2\mathcal{M}}{b}$, we can see the only requirements are $b \geq 2$, $\frac{2\mathcal{M}}{b}$ is an integer and*

$$\mathcal{N} \geq \frac{1}{4\epsilon}. \tag{4.9}$$

Thus we choose $b = 2$ for simplicity and $\mathcal{N} = \mathcal{M} \geq \frac{1}{4\epsilon}$.

Once $\mathbf{H}_{\mathcal{N}}$'s Takagi factorization is computed, we take $\mathbf{u}_{\mathcal{N}} = (u_0, u_1, \ldots, u_{\mathcal{N}-1})^T$ to be the last column of $\mathbf{U}$. Then find the $\mathcal{N}-1$ roots $\zeta_1, \zeta_2, \ldots, \zeta_{\mathcal{N}-1}$ of the polynomial $\sum_{n=0}^{\mathcal{N}-1} u_n z^n = 0$. Next the $\mathcal{N}-1$ weights ω_n are obtained by solving

$$h_m = \sum_{n=1}^{\mathcal{N}-1} \omega_n \zeta_n^m, \qquad \text{for } 0 \leq m \leq 2\mathcal{M},$$

in the least squares sense. Finally, parameters γ_n are obtained by formula (4.6).

In summary, the algorithm for determining coefficients ω_n and γ_n is (note that $b = 2$):

1. Input $\epsilon > 0$ as given accuracy.

2. Find the smallest integer $\mathcal{N}$ such that $\mathcal{N} \geq \frac{1}{4\epsilon}$.

3. Compute the spectral factorization of the matrix $\mathbf{H}_{\mathcal{N}} = \mathbf{V}\Lambda\mathbf{V}^T$.

4. Let $\mathbf{u} = (u_0, u_1, \ldots, u^T_{\mathcal{N}-1})$ be the last column of $\mathbf{V}$.

5. Find all roots $\zeta_1, \zeta_2, \ldots, \zeta_{\mathcal{N}-1}$ of the polynomial $\sum_{n=0}^{\mathcal{N}-1} u_n z^n = 0$ and check whether they are distinct. If they are not distinct then exit.

6. Solve $h_m = \sum_{n=1}^{\mathcal{N}-1} \omega_n \zeta_n^m$, $0 \leq m \leq 2\mathcal{N}$, in the least squares sense for ω_n.

7. Compute $\gamma_n = 2\mathcal{N} \log \zeta_n$.

Before ending this section we want to say a little more about ω_n and γ_n. As mentioned in Remark 3, $\mathbf{u}_{\mathcal{N}}$ is either a real vector or the product of a real vector and the imaginary unit $\sqrt{-1}$. In either case, the roots of $\sum_{n=0}^{\mathcal{N}-1} u_n z^n = 0$ will always be either real or pairwise complex conjugate. Thus ω_n are also real or pairwise complex conjugate, correspondingly. That is, if ζ_n is real, then ω_n is real too; whereas, if ζ_i and ζ_j are a complex conjugate pair, then ω_i and ω_j are a complex conjugate pair too, and vice versa. Furthermore, since $\gamma_n = 2\mathcal{N} \log \zeta_n$ we can see that $\exp(\gamma_n) = \zeta_n^{2\mathcal{N}}$ possesses the same conjugacy property. Thus $\omega_n \exp(\gamma_n x)$ are either real or pairwise complex conjugate for all real x. This result simplifies the calculation of $h_{\exp}(x) = \sum_{n=1}^{\mathcal{N}-1} \omega_n \exp(\gamma_n x)$. For real ω_n the term $\omega_n \exp(\gamma_n x)$ is evaluated as usual, whereas for the complex conjugate pair indexed by i and j, only one term needs to be evaluated, say $\omega_i \exp(\gamma_i x)$, and then the contribution of the complex conjugate pair of terms is $2\Re(\omega_i \exp(\gamma_i x))$, where $\Re(z)$ denotes the real part of the complex number z.

4.3 Properties of the approximation

In this section we discuss some properties of this approximation. One of the main results is noted above in Remark 5:

Proposition 3

$$\mathcal{N} \geq \frac{1}{4\epsilon}.$$

Noting that the diagonal matrix Σ is the same as the diagonal matrix of $\mathbf{H}_{\mathcal{N}}$'s singular value decomposition, we call $\sigma_{\mathcal{N}n}$, $n = 1, 2, \ldots, \mathcal{N}$, its singular value. Direct calculation shows that

$$\mathbf{H}_{\mathcal{N}}^{-1} = \begin{bmatrix} & & & & & & 1 \\ & & & & & 1 & -2 \\ & & & & 1 & -2 & 1 \\ & & & \iddots & \iddots & \iddots & \\ & & \iddots & \iddots & \iddots & & \\ & 1 & -2 & 1 & & & \\ 1 & -2 & 1 & & & & \end{bmatrix} \tag{4.10}$$

Proposition 4 *As $\mathcal{N}$ tends to infinity, the smallest singular value $\sigma_{\mathcal{N}\mathcal{N}}$ of the matrix $\mathbf{H}_{\mathcal{N}}$ tends to* $1/4$.

Proof Since $\mathbf{H}_{\mathcal{N}}$ is nonsingular, its singular values are positive. Proving that $\sigma_{\mathcal{N}\mathcal{N}}$ tends to $1/4$ as $\mathcal{N}$ tends to infinity is equivalent to proving that $\sigma_{\mathcal{N}\mathcal{N}}^{-1}$, the largest singular value of $\mathbf{H}_{\mathcal{N}}^{-1}$, tends to 4 as $\mathcal{N}$ tends to infinity.

From Gerschgorin's theorem [22, pp. 320] [27, pp. 344] [58, pp. 71], we know that all eigenvalues of $\mathbf{H}_{\mathcal{N}}^{-1}$ lie in the disc

$$D = \{z \in \mathbb{C} : |z| \leq 4\}.$$

Since $\mathbf{H}_{\mathcal{N}}^{-1}$ is real symmetric, we can conclude that all singular values of $\mathbf{H}_{\mathcal{N}}^{-1}$ are bounded by 4. Therefore, it suffices to prove that $\sigma_{\mathcal{N}\mathcal{N}}^{-1}$ approaches 4. To this end, note that, if $\mathbf{A}$

is a symmetric matrix, then $\max_{\mathbf{x}\neq 0}\frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{A}$. Hence, if for each $\mathcal{N}$, we can find a vector $\mathbf{x}_\mathcal{N}$ such that the Rayleigh quotient $\frac{\mathbf{x}_\mathcal{N}^T\mathbf{H}_\mathcal{N}^{-T}\mathbf{H}_\mathcal{N}^{-1}\mathbf{x}_\mathcal{N}}{\mathbf{x}_\mathcal{N}^T\mathbf{x}_\mathcal{N}} \to 16$ as $\mathcal{N}\to\infty$, then we can conclude that $\sigma_{\mathcal{N}\mathcal{N}}^{-1}\to 4$ as $\mathcal{N}\to\infty$.

For even $\mathcal{N}\geq 6$, let $\mathcal{N} = 2n$. Define a vector $\mathbf{x}_\mathcal{N} = (x_1, x_2, \ldots, x_\mathcal{N})^T$ by

$$\begin{aligned}
&x_1 = 1,\\
&x_i = x_{\mathcal{N}-i+2} = (-1)^{i-1}(i-1),\ \text{for } i = 2, 3, \ldots, n,\\
&x_{n+1} = -x_n.
\end{aligned}$$

Through direct calculation we obtain

$$\mathbf{x}_\mathcal{N}^T\mathbf{x}_\mathcal{N} = 1 + 2\sum_{i=1}^{n-1} i^2 + (n-1)^2, \tag{4.11}$$

and

$$\mathbf{H}_\mathcal{N}^{-1}\mathbf{x}_\mathcal{N} = \left[\begin{array}{c}
-1\\
4\\
\hdashline
-8\\
\vdots\\
(-1)^{n-1}4(n-2)\\
\hdashline
(-1)^n(4n-5)\\
(-1)^{n+1}4(n-1)\\
(-1)^n(4n-5)\\
\hdashline
(-1)^{n-1}4(n-2)\\
\vdots\\
-8\\
\hdashline
5
\end{array}\right],$$

which implies

$$\mathbf{x}_\mathcal{N}^T\mathbf{H}_\mathcal{N}^{-T}\mathbf{H}_\mathcal{N}^{-1}\mathbf{x}_\mathcal{N} = 10 + 2\cdot 4^2\sum_{i=1}^{n-2} i^2 + 4^2(n-1)^2 + 2(4n-5)^2.$$

This result together with (4.11) implies that for large even $\mathcal{N}$

$$\frac{\mathbf{x}_\mathcal{N}^T\mathbf{H}_\mathcal{N}^{-T}\mathbf{H}_\mathcal{N}^{-1}\mathbf{x}_\mathcal{N}}{\mathbf{x}_\mathcal{N}^T\mathbf{x}_\mathcal{N}} \approx \frac{2\cdot 4^2\sum_{i=1}^{n-1} i^2}{2\sum_{i=1}^{n-1} i^2} = 16.$$

Therefore, the largest singular value of $\mathbf{H}_\mathcal{N}^{-1}$ approaches 4 as $\mathcal{N}\to\infty$.

For odd $\mathcal{N} \geq 7$, let $\mathcal{N} = 2n+1$. Construct an $\mathcal{N}$-vector $\mathbf{x}_{\mathcal{N}} = (x_1, x_2, \ldots, x_{\mathcal{N}})^T$ with

$$\begin{aligned} &x_i = x_{\mathcal{N}-i+1} = (-1)^i i, \ \text{ for } i = 1, 2, \ldots, n, \\ &x_{n+1} = -x_n. \end{aligned}$$

Similar to the case for even $\mathcal{N}$, we have

$$\mathbf{x}_{\mathcal{N}}^T \mathbf{x}_{\mathcal{N}} = 2\sum_{i=1}^{n} i^2 + n^2, \tag{4.12}$$

$$\mathbf{H}_{\mathcal{N}}^{-1}\mathbf{x}_{\mathcal{N}} = \left[\begin{array}{c} -1 \\ 4 \\ \hdashline -8 \\ \vdots \\ (-1)^n 4(n-1) \\ \hdashline (-1)^{n+1}(4n-1) \\ (-1)^n 4n \\ (-1)^{n+1}(4n-1) \\ \hdashline (-1)^n 4(n-1) \\ \vdots \\ -8 \end{array}\right],$$

$$\mathbf{x}_{\mathcal{N}}^T \mathbf{H}_{\mathcal{N}}^{-T} \mathbf{H}_{\mathcal{N}}^{-1} \mathbf{x}_{\mathcal{N}} = 17 + 2 \cdot 4^2 \sum_{i=3}^{n} (i-1)^2 + 2(4n-1)^2 + 4^2 n^2. \tag{4.13}$$

Thus, from (4.12) and (4.13) we have that for large odd $\mathcal{N}$

$$\frac{\mathbf{x}_{\mathcal{N}}^T \mathbf{H}_{\mathcal{N}}^{-T} \mathbf{H}_{\mathcal{N}}^{-1} \mathbf{x}_{\mathcal{N}}}{\mathbf{x}_{\mathcal{N}}^T \mathbf{x}_{\mathcal{N}}} \approx \frac{2 \cdot 4^2 \sum_{i=1}^{n} i^2}{2\sum_{i=1}^{n} i^2} = 16.$$

The proof is completed.

As explained before, $\mathbf{u}_{\mathcal{N}}$ is either a real vector or the product of a real vector and the imaginary unit. In either case, the next three propositions hold. For simplicity we assume in the proofs that $\mathbf{u}_{\mathcal{N}}$ is a real vector.

Proposition 5 *The smallest singular value $\sigma_{\mathcal{N}\mathcal{N}}$ satisfies the relation $\sigma_{\mathcal{N}+2,\mathcal{N}+2} < \sigma_{\mathcal{N}\mathcal{N}}$, $\mathcal{N} \geq 2$.*

Proof Let $\lambda_{\mathcal{N}}$ be the eigenvalue of $\mathbf{H}_{\mathcal{N}}$ corresponding to $\sigma_{\mathcal{N}\mathcal{N}}$ and $\mathbf{u}_{\mathcal{N}} = (u_0, u_1, \ldots, u_{\mathcal{N}-1})^T \in \mathbb{R}^{\mathcal{N}}$ be a corresponding eigenvector. Thus $\mathbf{H}_{\mathcal{N}}\mathbf{u}_{\mathcal{N}} = \lambda_{\mathcal{N}}\mathbf{u}_{\mathcal{N}}$. Without loss of generality, assume $\|\mathbf{u}_{\mathcal{N}}\|_2 = 1$. Consequently, $\mathbf{u}_{\mathcal{N}}^T\mathbf{H}_{\mathcal{N}}^{-2}\mathbf{u}_{\mathcal{N}} = \sigma_{\mathcal{N}\mathcal{N}}^{-2}$.

Now we show that u_0 and u_1 cannot both be zero. Suppose $u_0 = u_1 = 0$ for some $\mathcal{N} > 2$. Note that $\mathbf{H}_{\mathcal{N}}^{-1}\mathbf{u}_{\mathcal{N}} = \lambda_{\mathcal{N}}^{-1}\mathbf{u}_{\mathcal{N}}$. That is

$$\begin{bmatrix} & & & & & & 1 \\ & & & & & 1 & -2 \\ & & & & 1 & -2 & 1 \\ & & & \iddots & \iddots & \iddots & \\ & & \iddots & \iddots & \iddots & & \\ & 1 & -2 & 1 & & & \\ 1 & -2 & 1 & & & & \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{\mathcal{N}-2} \\ u_{\mathcal{N}-1} \end{bmatrix} = \lambda_{\mathcal{N}}^{-1} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{\mathcal{N}-2} \\ u_{\mathcal{N}-1} \end{bmatrix}. \tag{4.14}$$

By comparing the two sides of the system of equations, we obtain $u_{\mathcal{N}-1} = 0$ from the first row; then $u_{\mathcal{N}-2} = 0$ from the second row; and then $u_2 = u_3 = 0$ from the last two rows. Continuing this process we end with $\mathbf{u}_{\mathcal{N}} = 0$, which contradicts $\mathbf{u}_{\mathcal{N}} \neq 0$.

Let $\mathbf{u}_{\mathcal{N}+2} = (0, 0, u_0, u_1, \ldots, u_{\mathcal{N}-1})^T$, then $\|\mathbf{u}_{\mathcal{N}+2}\|_2 = 1$, and

$$\mathbf{H}_{\mathcal{N}+2}^{-1}\mathbf{u}_{\mathcal{N}+2} = \begin{bmatrix} \mathbf{H}_{\mathcal{N}}^{-1}\mathbf{u}_{\mathcal{N}} \\ -2u_0 + u_1 \\ u_0 \end{bmatrix}.$$

Furthermore we have

$$\begin{aligned} \sigma_{\mathcal{N}+2,\mathcal{N}+2}^{-2} = \max_{\|\mathbf{x}\|=1} \mathbf{x}^T\mathbf{H}_{\mathcal{N}+2}^{-2}\mathbf{x}^T &\geq \mathbf{u}_{\mathcal{N}+2}^T\mathbf{H}_{\mathcal{N}+2}^{-2}\mathbf{u}_{\mathcal{N}+2} \\ &= \mathbf{u}_{\mathcal{N}}^T\mathbf{H}_{\mathcal{N}}^{-1}\mathbf{H}_{\mathcal{N}}^{-1}\mathbf{u}_{\mathcal{N}} + (u_1 - 2u_0)^2 + u_0^2 \\ &= \sigma_{\mathcal{N}\mathcal{N}}^{-2} + (u_1 - 2u_0)^2 + u_0^2 \\ &> \sigma_{\mathcal{N}\mathcal{N}}^{-2}. \end{aligned}$$

The last inequality follows from the observation above that u_0 and u_1 cannot both be zero. This completes the proof.

Proposition 6 *$u_0 \neq 0$ and $u_{\mathcal{N}-1} \neq 0$ for $\mathcal{N} \geq 4$.*

Proof From equation (4.14) we know that $u_0 = 0$ implies $u_{\mathcal{N}-1} = 0$, and vice versa. We will finish the proof by way of contradiction. Suppose that for some $\mathcal{N}$ we have $u_0 = 0$. Then by equation (4.14) we have

$$\left[\begin{array}{c|ccccc|c} & & & & & & 1 \\ \hline & & & & & 1 & -2 \\ & & & & 1 & -2 & 1 \\ & & & \iddots & \iddots & \iddots & \\ & & \iddots & \iddots & \iddots & & \\ & 1 & -2 & 1 & & & \\ \hline 1 & -2 & 1 & & & & \end{array}\right] \left[\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{\mathcal{N}-2} \\ \hline u_{\mathcal{N}-1} \end{array}\right] = \lambda_{\mathcal{N}}^{-1} \left[\begin{array}{c} u_0 \\ \hline u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{\mathcal{N}-2} \\ \hline u_{\mathcal{N}-1} \end{array}\right].$$

Since $u_0 = u_{\mathcal{N}-1} = 0$, deleting the first and the last rows (columns) of the matrix and correspondingly the first and the last elements of $\mathbf{u}_{\mathcal{N}}$, *i.e.*, u_0 and $u_{\mathcal{N}-1}$, results in a new system of equations

$$\begin{bmatrix} & & & & 1 \\ & & & 1 & -2 \\ & & \iddots & \iddots & \\ & \iddots & \iddots & & \\ 1 & -2 & 1 & & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{\mathcal{N}-2} \end{bmatrix} = \lambda_{\mathcal{N}}^{-1} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{\mathcal{N}-2} \end{bmatrix},$$

which says that $\lambda_{\mathcal{N}}$ is an eigenvalue of $\mathbf{H}_{\mathcal{N}-2}$. Note $(u_1, \ldots, u_{\mathcal{N}-2})^T \neq 0$, otherwise $\mathbf{u}_{\mathcal{N}} = 0$, which contradicts the definition of an eigenvector. By definition, $\sigma_{\mathcal{N}-2,\mathcal{N}-2} \leq |\lambda_{\mathcal{N}}|$. However, $\sigma_{\mathcal{N}\mathcal{N}} = |\lambda_{\mathcal{N}}|$, whence $\sigma_{\mathcal{N}\mathcal{N}} \geq \sigma_{\mathcal{N}-2,\mathcal{N}-2}$, which contradicts Proposition 5. Thus we conclude that $u_0 \neq 0$. This completes the proof.

Since $u_0 \neq 0$, zero is not a root of $\sum_{n=0}^{\mathcal{N}-1} u_n z^n = 0$, *i.e.*, $\zeta_n \neq 0$. To locate ζ_n we apply Schur's Theorem:

Theorem 1 (Schur) *[48, pp. 220] [42, pp. 109] The roots of the polynomial*

$$c_0 + c_1 z + \cdots + c_{n-1} z^{n-1} + c_n z^n = 0$$

are on or within the unit circle if and only if the quadratic form

$$\sum_{i=0}^{n-1} \left[(c_n x_i + c_{n-1} x_{i+1} + \cdots + c_{i+1} x_{n-1})^2 - (c_0 x_i + c_1 x_{i+1} + \cdots + c_{n-i-1} x_{n-1})^2 \right] \quad (4.15)$$

is positive semidefinite.

Proposition 7 *All the roots $\zeta_1, \zeta_2, \ldots, \zeta_{\mathcal{N}-1}$ of the polynomial $\sum_{n=0}^{\mathcal{N}-1} u_n z^n = 0$ are on or within the unit circle and they are either real or pairwise complex conjugate.*

Proof Let $\hat{\mathcal{N}} = \mathcal{N} - 1$; then the polynomial of interest can be written as $\sum_{n=0}^{\hat{\mathcal{N}}} u_n z^n = 0$. As explained in Section 4.2.2, all its roots $\zeta_1, \zeta_2, \ldots, \zeta_{\mathcal{N}-1}$ are either real or pairwise complex conjugate. To prove that all roots are on or within the unit circle, it suffices to prove the quadratic form

$$\sum_{n=0}^{\hat{\mathcal{N}}-1} \left[\left(u_{\hat{\mathcal{N}}} x_n + u_{\hat{\mathcal{N}}-1} x_{n+1} + \cdots + u_{n+1} x_{\hat{\mathcal{N}}-1} \right)^2 - \left(u_0 x_n + u_1 x_{n+1} + \cdots + u_{\hat{\mathcal{N}}-n-1} x_{\hat{\mathcal{N}}-1} \right)^2 \right] \tag{4.16}$$

is positive semidefinite. Note that this claim is implied by the positive semidefiniteness of the quadratic form

$$\sum_{n=0}^{\hat{\mathcal{N}}} \left[\left(u_{\hat{\mathcal{N}}} x_n + \cdots + u_{n+1} x_{\hat{\mathcal{N}}-1} + u_n x_{\hat{\mathcal{N}}} \right)^2 - \left(u_0 x_n + \cdots + u_{\mathcal{N}-n-1} x_{\mathcal{N}-1} + u_{\hat{\mathcal{N}}-n} x_{\hat{\mathcal{N}}} \right)^2 \right],$$

or equivalently the positive semidefiniteness of the matrix

$$\mathbf{C}^T\mathbf{C} - \mathbf{D}^T\mathbf{D},$$

where

$$\mathbf{C} = \begin{bmatrix} u_{\hat{\mathcal{N}}} & u_{\hat{\mathcal{N}}-1} & \cdots & u_0 \\ & u_{\hat{\mathcal{N}}} & \cdots & u_1 \\ & & \cdots & \cdots \\ & & & u_{\hat{\mathcal{N}}} \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{\hat{\mathcal{N}}} \\ & u_0 & \cdots & u_{\hat{\mathcal{N}}-1} \\ & & \cdots & \cdots \\ & & & u_0 \end{bmatrix}. \tag{4.17}$$

From Proposition 6 we know that $u_{\hat{\mathcal{N}}} \neq 0$, so $\mathbf{C}^{-1}$ exists. Therefore

$$\mathbf{C}^T\mathbf{C} - \mathbf{D}^T\mathbf{D} = \mathbf{C}^T \left(\mathbf{I} - \mathbf{C}^{-T}\mathbf{D}^T\mathbf{D}\mathbf{C}^{-1} \right) \mathbf{C}.$$

Let $\mathbf{Y} = \mathbf{D}\mathbf{C}^{-1}$. It is easy to verify that

$$\mathbf{Y} = \lambda_{\mathcal{N}} \begin{bmatrix} 1 & -2 & 1 & & & \\ & \cdots & \cdots & & & \\ & & \cdots & \cdots & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \\ & & & & & 1 \end{bmatrix} = \lambda_{\mathcal{N}} \mathbf{P}\mathbf{H}_{\mathcal{N}}^{-1},$$

where $\lambda_{\mathcal{N}} \neq 0$ is the eigenvalue of $\mathbf{H}_{\mathcal{N}}$ that corresponds to the eigenvector $\mathbf{u}_{\mathcal{N}}$, *i.e.*, $\mathbf{H}_{\mathcal{N}}\mathbf{u}_{\mathcal{N}} = \lambda_{\mathcal{N}}\mathbf{u}_{\mathcal{N}}$, and

$$\mathbf{P} = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & \end{bmatrix}$$

is a permutation matrix. Consequently,

$$\mathbf{Y}^T\mathbf{Y} = \lambda_{\mathcal{N}}^2 \mathbf{H}_{\mathcal{N}}^{-T}\mathbf{P} \cdot \mathbf{P}\mathbf{H}_{\mathcal{N}}^{-1} = \sigma_{\mathcal{N}\mathcal{N}}^2 \mathbf{H}_{\mathcal{N}}^{-T}\mathbf{H}_{\mathcal{N}}^{-1} = \sigma_{\mathcal{N}\mathcal{N}}^2 \mathbf{H}_{\mathcal{N}}^{-2}$$

since $|\lambda_{\mathcal{N}}| = \sigma_{\mathcal{N}\mathcal{N}}$ and $\mathbf{P}^{-1} = \mathbf{P}$. Let $\mathbf{H}_{\mathcal{N}} = \mathbf{V}\Lambda\mathbf{V}^T$ be the spectral decomposition of $\mathbf{H}_{\mathcal{N}}$, where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{\mathcal{N}})$, $|\lambda_n| = \sigma_{\mathcal{N}n}$, for $1 \leq n \leq \mathcal{N}$, and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. Thus

$$\mathbf{I} - \mathbf{Y}^T\mathbf{Y} = \mathbf{V}\left[\mathbf{I} - \sigma_{\mathcal{N}\mathcal{N}}^2 \Lambda^{-2}\right]\mathbf{V}^T = \mathbf{V}\left[\mathbf{I} - \sigma_{\mathcal{N}\mathcal{N}}^2 \left[\mathrm{diag}(\sigma_{\mathcal{N}1}, \sigma_{\mathcal{N}2}, \ldots, \sigma_{\mathcal{N}\mathcal{N}})\right]^{-2}\right]\mathbf{V}^T.$$

Since $\sigma_{\mathcal{N}1} \geq \sigma_{\mathcal{N}2} \geq \ldots \geq \sigma_{\mathcal{N}\mathcal{N}} > 0$, $\mathbf{I} - \sigma_{\mathcal{N}\mathcal{N}}^2 \left[\mathrm{diag}(\sigma_{\mathcal{N}1}, \sigma_{\mathcal{N}2}, \ldots, \sigma_{\mathcal{N}\mathcal{N}})\right]^{-2}$ is positive semidefinite. Thus $\mathbf{C}^T\mathbf{C} - \mathbf{D}^T\mathbf{D}$ is positive semidefinite. This completes the proof.

Since $\gamma_n = 2\mathcal{M}\log\zeta_n$, Proposition 7 implies that $\Re(\gamma_n) \leq 0$, but all our numerical results show that all roots ζ_n are strictly within the unit circle, whence $\Re(\gamma_n) < 0$. Thus $\exp(\gamma_n x)$ converges to zero as x goes to infinity. This leads to the following conjecture.

Conjecture 1 *All the roots $\zeta_1, \zeta_2, \ldots, \zeta_{\mathcal{N}-1}$ of the polynomial $\sum_{n=0}^{\mathcal{N}-1} u_n z^n = 0$ are strictly within the unit circle.*

4.4 Numerical results III

Presented below are plots related to different numbers, N, of terms in the exponential approximation $h_{\exp}(x)$ to the hockey stick function $h(x)$. In Figure 4.1 we plot the parameters for the 25-term exponential approximation. Conjugacy of ω_n and also γ_n is clearly shown in the plot. In Figure 4.2 we present the singular values of the Hankel matrix $\frac{1}{\mathcal{N}}\mathbf{H}_{\mathcal{N}}$ associated with this 25-term exponential approximation.

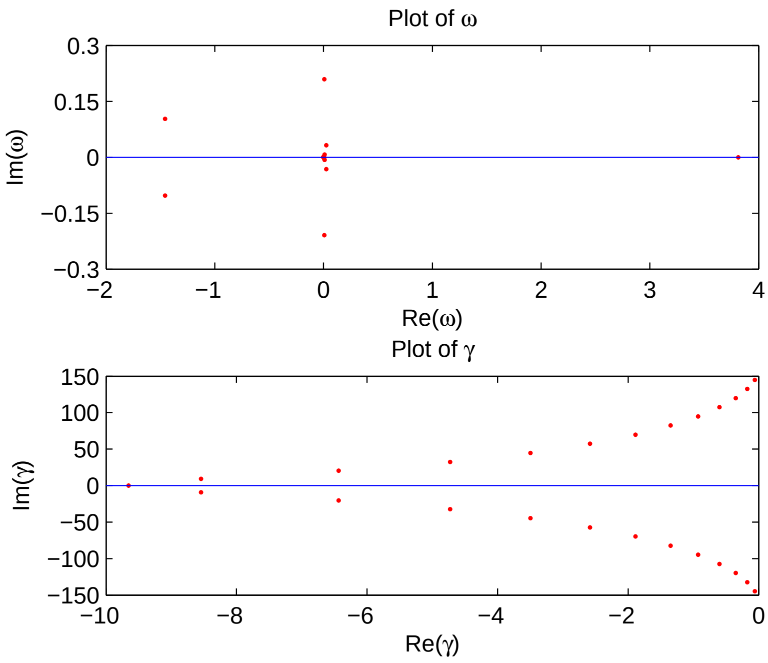

Figure 4.1: The parameters ω_n and γ_n for the 25-term exponential approximation

In Figure 4.3 we illustrate the hockey stick function and its approximations. The left panel shows the hockey stick function and its 5-term exponential approximation. The right panel shows the hockey stick function and its 50-term exponential approximation. From the plots we can see that a sum of 50 exponential terms approximates the hockey stick function very well.

Finally in Figure 4.4 we plot the approximation errors of the 25-, 50-, 100-, 200-, and 400-term approximations over the interval $[0, 30]$. From these plots we can see that the approximation errors for all five choices of $\mathcal{N}$ converge to zero as the variable x increases.

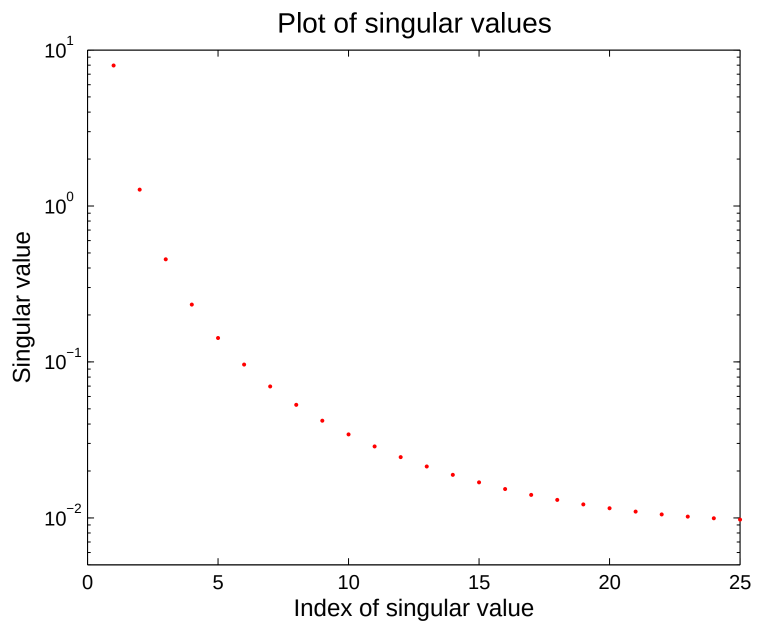

Figure 4.2: The singular values associated with the 25-term exponential approximation

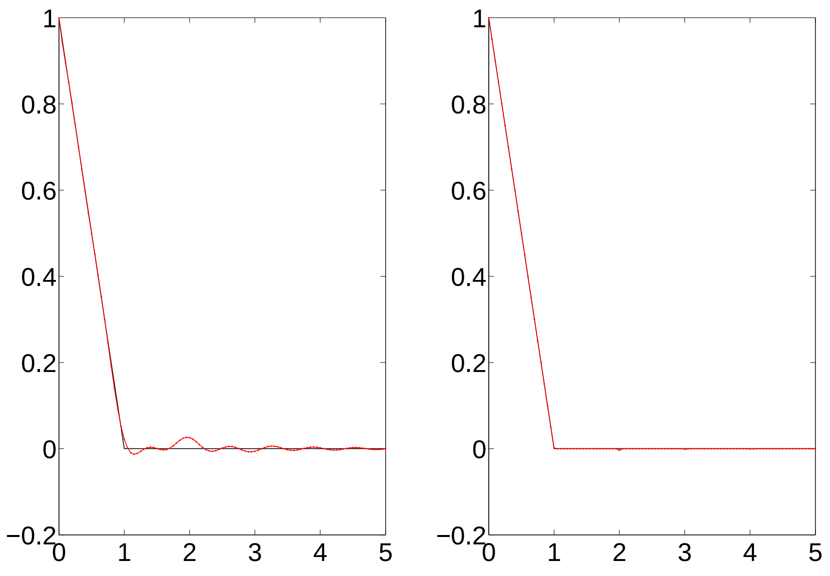

Figure 4.3: Left panel: 5-term exponential approximation; Right panel: 50-term exponential approximation

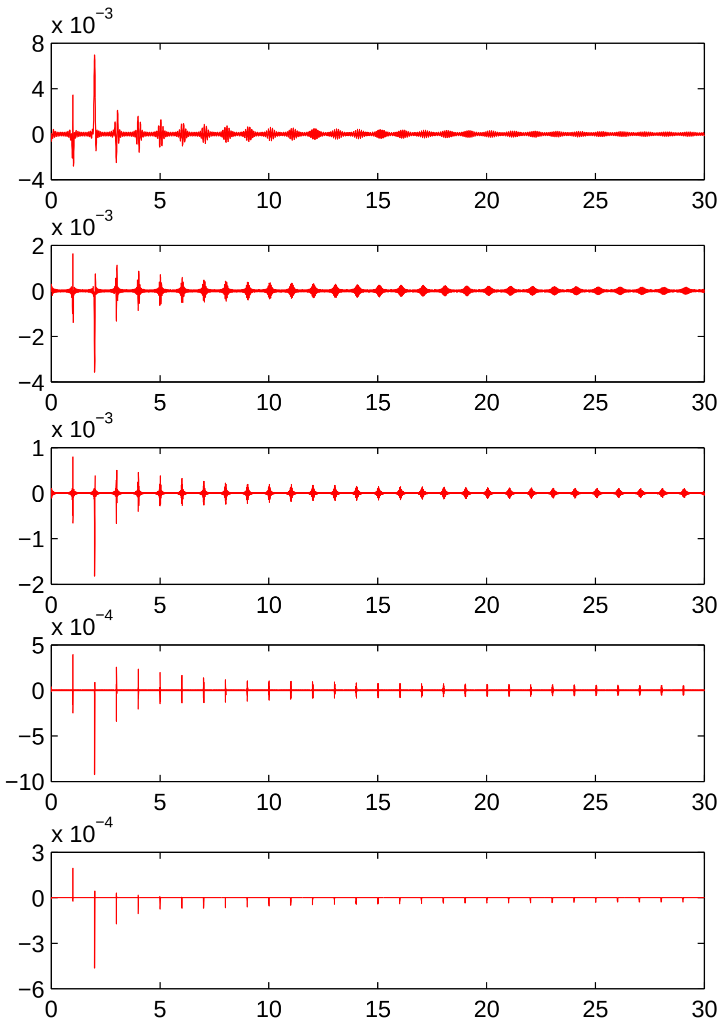

Figure 4.4: The panels from top to bottom are the approximation errors of the 25-term to 400-term exponential approximations to the HS function over $[0, 30]$, with the number of terms doubling in successive panels.

Chapter 5

Conclusions and discussion

In this thesis we proposed four new numerical methods for estimating the expected value of a synthetic CDO tranche loss. We first proposed three new numerical methods for estimating the loss distribution of the underlying pool: a stable recursive method, an improved compound Poisson approximation method, and a normal power approximation method. The recursive method computes the exact loss distribution, whereas the other two methods approximate the loss distribution. We showed that the recursive method is stable. Numerical experiments illustrate that it is efficient when the underlying pool is homogeneous in terms of loss-given-defaults or has a low dispersion of loss-given-defaults. The improved compound Poisson approximations are efficient for high dispersion pools while the normal power approximation is an alternative for large pools or when computational cost is generally more important than accuracy, as is the case in risk management.

We also proposed a new method that focuses on the tranche loss function itself. The tranche loss function is expressed simply in terms of two basis functions. Each of the two basis functions is a transformation of the hockey stick function. By approximating the hockey stick function by a sum of exponentials, the tranche loss function is approximated by a sum of exponentials. In this way, the estimation of the expected value of the tranche loss function is reduced to the estimation of a series of expected values of the individual reference entities in the underlying pool. A main advantage of this method is that the distribution of the pool loss need not be estimated.

In the thesis we also studied theoretical properties of the exponential approximation to the hockey stick function. Some of the results presented in Chapter 4 could be strengthened. In particular, we hope to find a proof of Conjecture 1 on page 62.

As noted above, the hockey stick function $h(x)$ is approximated by a sum of exponentials:

$$h(x) \approx h_{\exp}(x) = \sum_{n=1}^{N} \omega_n \exp(\gamma_n x).$$

Based on this approximation, the cost of computing $\mathbb{E}\left[f(\mathscr{L}^P; \ell, u)\right]$ grows linearly with K, the number of reference entities in the underlying pool. On the other hand, as shown in Chapter 4, the error associated with this approximation is also a linear function of N, the number of terms in the approximation. This is a disadvantage of this approximation, though for a reasonably large N, say $N = 100$, the computed spreads are accurate enough for practical applications. We hope to find similar approximations that converge faster.

Bibliography

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*, Dover Publications, 1965.

[2] L. Andersen, J. Sidenius, and S. Basu, *All your hedges in one basket*, Risk, (2003), pp. 61–72.

[3] A. Antonov, S. Mechkov, and T. Misirpashaev, *Analytical techniques for synthetic CDOs and credit default risk measures.* Available from http://www.defaultrisk.com/, May 2005.

[4] E. Banks, *Synthetic and Structured Assets*, John Wiley & Sons, Ltd, 2006.

[5] R. E. Beard, T. Pentikäinen, and E. Pesonen, *Risk Theory: The Stochastic Basis of Insurance*, Monographs on Statistics and Applied Probability, Chapman and Hall, 3rd ed., 1984.

[6] A. Bernand, F. Pourmokhtar, B. Jacquard, D. Baum, M. Levy, L. Gibson, L. Andersen, and J. Sidenius, *The Bank of America Guide to Advanced Correlation Products*, Bank of America, 2004.

[7] G. Beylkin and L. Monzón, *On approximation of functions by exponential sums*, Applied and Computational Harmonic Analysis, 19 (2005), pp. 17–48.

[8] D. Braess, *Nonlinear Approximation Theory*, vol. 7 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin Heidelberg, 1986.

[9] X. BURTSCHELL, J. GREGORY, AND J.-P. LAURENT, *A comparative analysis of CDO pricing models*. Available from http://www.defaultrisk.com/, April 2005.

[10] A. CIFUENTES AND M. DESAI, *Structured Products Research: CDOs*, Wachovia Securities, 2004.

[11] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, The MIT electrical engineering and computer science series, The MIT Press, Cambridge, Massachusetts, 2000.

[12] C. L. CULP, *Structured Finance and Insurance*, John Wiley & Sons, Inc., 2006.

[13] N. DE PRIL AND J. DHAENE, *Error bounds for compound Poisson approximations of the individual risk model*, ASTIN Bulletin, 22 (1992), pp. 136–148.

[14] B. DE PRISCO, I. ISCOE, AND A. KREININ, *Loss in translation*, Risk, (2005), pp. 77–82.

[15] J. DEACON, *Global Securitisation and CDOs*, John Wiley & Sons, Ltd, 2004.

[16] J. DHAENE, B. SUNDT, AND N. DE PRIL, *Some moment relations for the Hipp approximation*, ASTIN Bulletin, 26 (1996), pp. 117–121.

[17] F. J. FABOZZI AND M. CHOUDHRY, *The Handbook of European Structured Financial Products*, The Frank J. Fabozzi Series, John Wiley & Sons, Inc., 2004.

[18] FITCHRATINGS, *Global credit derivatives survey: Risk dispersion accelerates*, tech. report, FitchRatings, 2005. Special Report November 17, 2005.

[19] C. FRANCIS, A. KAKODKAR, AND B. MARTIN, *Credit Derivative Handbook 2003*, Merrill Lynch, April 16, 2003.

[20] D. A. S. FRASER, *Probability & Statistics: Theory and Applications*, Institute of Theoretical Statistics, Toronto, Canada, 1976.

[21] G. GOLUB AND V. PEREYRA, *Separable nonlinear least squares: The variable projection method and its applications*, Inverse Problems, 19 (2003), pp. R1–R26.

[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 2nd ed., 1991.

[23] L. S. Goodman and F. J. Fabozzi, *Collaterialized Debt Obligations: Structures and Analysis*, John Wiley & Sons, Inc., 2002.

[24] M. Gordy and D. Jones, *Random tranches*, Risk, 16 (2003), pp. 78–83.

[25] M. T. Heath, *Scientific Computing: An Introductory Survey*, McGraw-Hill, 2nd ed., 2002.

[26] C. Hipp, *Improved approximations for the aggregate claims distribution in the individual model*, ASTIN Bulletin, 16 (1986), pp. 89–100.

[27] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

[28] J. Hull, *Options, Futures, and Other Derivatives*, Prentice Hall, Upper SaddleRiver, New Jersey 07458, 5th ed., 2003.

[29] J. Hull and A. White, *Valuing credit default swaps* I*: No counterparty default risk*, Journal of Derivatives, 8 (2000), pp. 29–40.

[30] ——, *Valuation of a CDO and an n^{th} to default CDS without Monte Carlo simulation*, Journal of Derivatives, (2004), pp. 8–23.

[31] I. Iscoe, K. Jackson, A. Kreinin, and X. Ma, *On exponential approximation to the hockey-stick function.* Available from http://www.cs.toronto.edu/NA/reports.html, January 2007.

[32] ——, *Pricing correlation-dependent derivatives based on exponential approximations to the hockey stick function.* Available from http://www.cs.toronto.edu/NA/reports.html, January 2007.

[33] I. Iscoe and A. Kreinin, *Valuation of synthetic CDOs.* Submitted to the Journal of Banking & Finance, 2005.

[34] I. Iscoe, A. Kreinin, and D. Rosen, *An integrated market and credit risk portfolio model*, Algo Research Quarterly, 2 (1999), pp. 21–38.

[35] L. G. Ixaru and G. V. Berghe, *Exponential Fitting*, Kluwer Academic Publishers, 2004.

[36] K. Jackson, A. Kreinin, and X. Ma, *Loss distribution evaluation for synthetic CDOs*. Available from http://www.cs.toronto.edu/NA/reports.html, February 2007.

[37] Z. Jing and A. T. Fam, *An algorithm for computing continuous Chebyshev approximations*, Mathematics of Computation, 48 (1987), pp. 691–710.

[38] S. A. Klugman, H. H. Panjer, and G. E. Willmot, *Loss Models from Data to Decisions*, John Wiley & Sons, Inc., 1998.

[39] J.-P. Laurent and J. Gregory, *Basket default swaps, CDOs and factor copulas*, Journal of Risk, 7 (2005), pp. 103–122.

[40] D. X. Li, *On default correlation: A copula function approach*, Journal of Fixed Income, 9 (2000), pp. 43–54.

[41] D. Lucas, *CDO Handbook*, JP Morgan, Global Structured Finance Research, May 2001. Available from http://www.mayerbrownrowe.com/.

[42] M. Mignotte and D. Stefanescu, *Polynomials — An Algorithmic Approach*, Springer-Verlag, 1999.

[43] P. Moore, *The ABC of CDO*, Credit, (May 2004), pp. 1–33.

[44] D. O'Kane, M. Naldi, S. Ganapati, A. Berd, C. Pedersen, L. Schloegl, and R. Mashal, *The Lehman Brothers Guide to Exotic Credit Derivatives*, Lehman Brothers, 2003.

[45] H. H. Panjer, *Recursive evaluation of a family of compound distributions*, ASTIN Bulletin, 12 (1981), pp. 22–26.

[46] A. PIERRON, *Collateralized debt obligations market*, tech. report, Celent, October 31, 2005.

[47] W. PRESS, B. FLANNERY, S. TEUKOLSKY, AND W. VETTERLING, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[48] A. RALSTON, *A First Course in Numerical Analysis*, McGraw-Hill Book Company, 1965.

[49] C. RAMSAY, *A note on the normal power approximation*, ASTIN Bulletin, 21 (1991), pp. 147–150.

[50] P. J. SCHÖNBUCHER, *Credit Derivatives Pricing Models*, Wiley Finance Series, John Wiley & Sons Canada Ltd., 2003.

[51] STANDARD & POOR'S, *Structured Finance – Glossary of Securitization Terms*, Standard & Poor's, 2003.

[52] J. M. TAVAKOLI, *Collateralized Debt Obligations and Structured Finance*, John Wiley & Sons, Inc., 2003.

[53] J. L. TEUGELES, *Encyclopedia of Actuarial Science*, John Wiley & Sons, Ltd, 2004.

[54] THE BOND MARKET ASSOCIATION, *The Bond Market Association global CDO marekt issuance data*, July 10, 2006.

[55] THE NUMERICAL ALGORITHMICS GROUP, *The NAG C Libray Manual, Mark 7*, The Numerical Algorithmics Group Limited, 2002.

[56] VANDERBILT CAPITAL ADVISORS, *Calculating implied default rates from CDS spreads.* Available from http://www.vcallc.com/mailings/additions/cdsspreads2.htm.

[57] O. VASICEK, *Probability of loss distribution*, tech. report, KMV, February 1987. Available from http://www.moodyskmv.com/research/.

[58] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.

[59] J. YANG, T. HURD, AND X. ZHANG, *Saddlepoint approximation method for pricing CDOs*, Journal of Computational Finance, 10 (2006), pp. 1–20.