Loss Distribution Evaluation for Synthetic CDOs*

Ken Jackson† Alex Kreinin‡ Xiaofang Ma§

February 12, 2007

Abstract

Efficient numerical methods for evaluating the loss distributions of synthetic CDOs are important for both pricing and risk management purposes. In this paper we discuss several methods for loss distribution evaluations. We first develop a stable recursive method. Then the improved compound Poisson approximations proposed by Hipp [12] are introduced. Finally, the normal power approximation method that has been used in actuarial science is described. The recursive method computes the loss distribution exactly, whereas the other two methods compute the loss distribution approximately. Numerical results based on these three and some known methods for synthetic CDO pricing are given.

*This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

†Department of Computer Science, University of Toronto, 10 King's College Rd, Toronto, ON, M5S 3G4, Canada; krj@cs.utoronto.ca

‡Algorithmics Inc., 185 Spadina Avenue, Toronto, ON, M5T 2C6, Canada; Alex.Kreinin@algorithmics.com

§Department of Computer Science, University of Toronto, 10 King's College Rd, Toronto, ON, M5S 3G4, Canada; csxfma@cs.utoronto.ca

1 Introduction

Pricing and risk management of synthetic collateralized debt obligations (CDOs) have been well studied by researchers from both the financial industry and academia. Efficient methods for synthetic CDO pricing are in demand for these purposes. One way to improve the efficiency is to simplify the pricing models so that analytic or at least semi-analytic pricing expressions can be obtained. Factor models, such as the reduced-form model proposed by Laurent and Gregory [16] and the structural model proposed Vasicek [23] and Li [17] are widely used in practice to obtain analytic or semi-analytic formulas to price synthetic CDOs efficiently. For a comparative analysis of different factor models, please see the paper by Burtschell, Gregory and Laurent [4].

Another way to improve the performance of pricing methods is to develop fast algorithms for basic numerical problems. One issue common to synthetic CDO pricing and related risk management is how to evaluate the pool's loss distribution accurately and efficiently. From a computational point of view, Monte Carlo simulation is the last resort because of its inefficiency, despite its flexibility. Widely used methods can be divided into two classes. The first class evaluates a pool's loss distribution exactly, based on the assumption that all obligors' losses-given-default sit on a common lattice. Among these methods are the ASB method proposed by Andersen, Sidenius and Basu [2], the LG method by Laurent and Gregory [16], and the HW method by Hull and White [14]. Note that both the ASB and 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 small number of homogeneous pools to which the HW method can be applied. 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. 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 [6] is an example of a method of this class.

In this paper, we focus on efficient numerical algorithms for the evaluation of a pool's loss distribution for a synthetic CDO. The correlation structure of default events is described by a Gaussian copula. The first proposed method computes the pool loss distribution exactly provided that each name's recovery-adjusted notional value sits on a common lattice. It formulates the problem in a way different from HW's approach. Analysis shows this recursive procedure is efficient and stable. Although it is directly applicable to homogeneous pools only, it can be extended to inhomogeneous pools in the same way as the HW method. Thus it differs from the ASB method. However, for a pool in which all recovery-adjusted notional values are either distinct or all the same, our method is the same as the ASB method. We also generalize our approach to multi-state problems. The second proposed method is based on the improved compound Poisson approximation of Hipp [12] to improve the accuracy of the basic compound Poisson approximation [6]. The third proposed method uses a normal power approximation, widely used in actuarial science, to approximate the pool's loss distribution. This approach is especially suitable for large pools for which the names' losses are not necessarily sitting on a common lattice.

The remainder of the paper is organized as follows. The basic pricing equation is described in Section 2. The one-factor Gaussian copula framework for CDO pricing is described in Section 3. The recursive procedure and its generalization are presented in Section 4. The improved compound Poisson approximation and the normal power approximation methods are described in Section 5. Accuracy and performance comparisons between the methods introduced in this paper and the ASB, the HW and the basic compound Poisson approximation methods are presented in Section 6. The paper ends with some conclusions in Section 7.

2 Pricing equation

We consider a synthetic CDO tranche of size S with an attachment point ℓ, a threshold that determines whether some of the pool losses shall be absorbed by this tranche. If the realized losses of the pool are less than ℓ, then the tranche will not suffer any loss, otherwise it will absorb losses up to its size S. The threshold $S + \ell$ is called the detachment point of the tranche.

Assume there are K names in the pool. For name k, its notional value and the recovery rate of the notional value of the reference asset are N_k and R_k, respectively. Then the loss-given-default or the recovery-adjusted notional value of name k, is $LGD_k = N_k(1 - R_k)$. Let $0 = t_0 < t_1 < t_2 < \cdots < t_n = T$ be the set of premium dates, with T denoting the maturity of the CDO tranche. Assume that the interest rates are deterministic. Then the set of discount factors, denoted by $d_1, d_2, \ldots, d_n$, are deterministic. Let $\mathscr{L}_i^P$ be the pool's cumulative losses up to time t_i. Then the losses absorbed by the specified tranche up to time t_i, denoted by L_i, is $L_i = f(\mathscr{L}_i^P; \ell, S + \ell) = \min\big(S, (\mathscr{L}_i^P - \ell)^+\big)$, where $x^+ = \max(x, 0)$. In this paper, the function f is called the payoff function of the specified tranche. It is also known as the stop-loss function in actuarial science [3], [15].

Assume that the fair spread for the tranche is a constant s per annum. The two important quantities to be determined in synthetic CDO tranche valuation are the present value of the **default leg** — the expected losses of the tranche over the life of the contract; and the present value of the **premium leg** — the expected premiums that the tranche investor will receive over the life of the contract. Mathematically, the two legs' present values are

$$\text{PV[Premium leg]} = \mathbb{E}\left[\sum_{i=1}^{n} s(S - L_i)(t_i - t_{i-1})d_i\right], \tag{1}$$

$$\text{PV[Default leg]} = \mathbb{E}\left[\sum_{i=1}^{n} (L_i - L_{i-1})d_i\right]. \tag{2}$$

Noting that the discount factors d_i are deterministic, the fair spread s is

$$s = \frac{\sum_{i=1}^{n} \mathbb{E}\left[L_i - L_{i-1}\right] d_i}{\sum_{i=1}^{n} \mathbb{E}\left[S - L_i\right] (t_i - t_{i-1}) d_i}, \tag{3}$$

where $\mathbb{E}\left[L_0\right] = 0$, due to a natural assumption that there is no default at t_0, the start of the contract. If the value of the spread is known, then the value of the tranche to the tranche investor today is

$$s \cdot \text{PV[Premium leg]} - \text{PV[Default leg]}. \tag{4}$$

Noting that the tranche size S is a given constant and the time periods $t_i - t_{i-1}$ are fixed, it follows from (3) and (4) that the valuation problem is now reduced to the computation of the expected cumulative losses $\mathbb{E}[L_i]$, $i = 1, 2, \ldots, n$. In order to compute these expectations, we have to specify the default processes for each of the names and the correlation structure of the default events.

3 One-factor model

Factor models in the conditional independence framework are the most popular approach for synthetic CDO valuation due to their tractability. Factor models were first introduced by Vasicek [23] to estimate the loan loss distribution of a pool of loans. In this paper, we use a one-factor model to illustrate our algorithms.

Assume that the risk-neutral default probabilities $\pi_k(t) = \mathbb{P}(\tau_k < t)$, $k = 1, 2, \ldots, K$, that describe the default-time distributions of all K names are available[1], where τ_k is the default time of name k. The dependence structure of the default times is determined by the creditworthiness indexes Y_k through a one-factor copula. The creditworthiness indexes Y_k are defined by

$$Y_k = \beta_k X + \sigma_k \varepsilon_k, \qquad k = 1, 2, \ldots, K, \tag{5}$$

where X is the systematic risk factor, ε_k are idiosyncratic factors that are mutually independent and are also independent of X; β_k and $\sigma_k \geq 0$ are real constants satisfying $\beta_k^2 + \sigma_k^2 = 1$. The risk-neutral default probabilities and the creditworthiness indexes are related by the copula model

$$\pi_k(t) = \mathbb{P}(\tau_k < t) = \mathbb{P}(Y_k < H_k(t)), \tag{6}$$

where $H_k(t)$ is the default barrier of the k-th name at time t. The copula model was first introduced by Li [17] and then used in portfolio credit risk analyses, including synthetic CDO valuation, by Gordy and Jones [10], Hull and White [14], De Prisco, Iscoe and Kreinin [6],

[1]These default probabilities are input to the described synthetic CDO valuation model. They can be evaluated, for example, using a method for CDS pricing [8], [13], [22].

Laurent and Gregory [16], and Schönbucher [19], to name just a few. Thus the correlations of default events are captured by the systematic risk factor X. Conditional on a given value x of X, all default events are independent. If we further assume that X and ε_k follow standard normal distributions, as we do in this paper, then we have the so called one-factor Gaussian copula model and from (6) we have that $H_k(t) = \Phi^{-1}(\pi_k(t))$. The correlation between two different indexes Y_i and Y_j is $\beta_i\beta_j$. Note that if X is a random vector then we have a multi-factor copula model. On the other hand, if we assume that X and ε_k follow two student t-distributions of different degrees of freedom, then we obtain a double-t copula [14]. Other generalizations can be found in [4].

The conditional risk-neutral default-time distribution is defined by

$$\pi_k(t;x) = \mathbb{P}\left(Y_k < H_k(t) | X = x\right). \tag{7}$$

Thus from (5) and (7) we have

$$\pi_k(t;x) = \Phi\left(\frac{H_k(t) - \beta_k x}{\sigma_k}\right). \tag{8}$$

The conditional and unconditional risk-neutral default-time probabilities at the premium date t_i are denoted by $\pi_k(i;x)$ and $\pi_k(i)$, respectively.

In this conditional independence framework, the expected cumulative tranche losses $\mathbb{E}[L_i]$ can be computed as

$$\mathbb{E}[L_i] = \int_{-\infty}^{\infty} \mathbb{E}_x[L_i] \,\mathrm{d}\Phi(x), \tag{9}$$

where $\mathbb{E}_x[L_i] = \mathbb{E}_x\left[f(\mathscr{L}_i^P;\ell,S+\ell)\right] = \mathbb{E}_x\left[\min\left(S,(\mathscr{L}_i^P-\ell)^+\right)\right]$ is the expectation of L_i conditional on $X = x$, where $\mathscr{L}_i^P = \sum_{k=1}^K LGD_k \mathbf{1}_{\{Y_k<H_k(t_i)\}}$, and the indicators $\mathbf{1}_{\{Y_k<H_k(t_i)\}}$ are mutually independent conditional on X. Generally, the integration in (9) needs to be approximated numerically using a quadrature rule:

$$\mathbb{E}[L_i] \approx \sum_{m=1}^{M} \mathrm{w}_m \mathbb{E}_{x_m}\left[f(\mathscr{L}_i^P;\ell,S+\ell)\right], \tag{10}$$

where x_m and w_m are 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 [18].

We see from (10) that the fundamental numerical problem in synthetic CDO tranche valuation is how to evaluate the conditional expectation $\mathbb{E}_{x_m}\left[f(\mathscr{L}_i^P;\ell,S+\ell)\right]$ for a fixed abscissa x_m at a fixed time t_i for the fixed attachment and detachment points ℓ and $S+\ell$. In the remainder of this paper, this expectation is denoted as $\mathbb{E}\left[f(\mathscr{L}^P;\ell,S+\ell)\right]$, 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 $\mathbf{1}_{\{k\}}=\mathbf{1}_{\{Y_k<H_k(t_i)\}}$.

4 Recursive method for loss distribution evaluation

Note that, in practice, the loss-given-default values of the referred assets are not necessarily the same; thus a CDO pool is not necessarily homogeneous. However, the assets in a pool can usually be divided into a few groups such that in each group all assets have the same loss-given-default value. Therefore, an important basic problem is how to evaluate the loss distribution for a homogeneous pool.

4.1 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 the common $LGD=1$. 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

$$\mathbf{p}_{k+1} = \begin{pmatrix} p_{k+1,k+1} \\ p_{k+1,k} \\ \vdots \\ p_{k+1,1} \\ p_{k+1,0} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_k & 0 \\ 0 & \mathbf{p}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ S_{k+1} \end{pmatrix}, \tag{11}$$

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

The algorithm based on formula (11) is numerically stable. Let ε be the *machine epsilon* (see Golub and Van Loan [9] or Heath [11] for the definition) for a floating-point system. Assume that the input probabilities Q_k are exactly representable in the computer system. 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 (11) in a floating-point system:

$$\hat{\mathbf{p}}_{k+1} = \mathrm{eval}\left(\begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} \right), \quad \hat{\mathbf{p}}_1 = \begin{pmatrix} Q_1 \\ \hat{S}_1 \end{pmatrix}. \tag{12}$$

Then we have the following result:

Proposition 1 *The error vector ϵ_{k+1} satisfies*

$$\|\epsilon_k\|_\infty \le \left(1.001^{k-1} c - 3001\right) \varepsilon, \ \textit{for } k = 1, 2, \ldots, K, \tag{13}$$

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

See the Appendix for a proof of Proposition 1.

If it is assumed that both Q_k and S_k are exactly representable in the computer system, then $\|\epsilon_1\|_\infty = 0$ and the error bound (26) for $\|\epsilon_{k+1}\|$ in the proof of Proposition 1 in the Appendix reduces to

$$\|\epsilon_{k+1}\|_\infty \le 1.001\|\epsilon_k\|_\infty + 2.001\varepsilon, \ \text{for } k = 1, 2, \ldots, K-1. \tag{14}$$

In this case, the relation (13) reduces to

$$\|\epsilon_k\|_\infty \le \left(1.001^k c' - 2001\right)\varepsilon, \quad \text{for } k = 1, 2, \ldots, K, \tag{15}$$

where $c' = 2001/1.001$.

Inequalities (13) and (15) ensure that the recursive formula (11) is numerically stable.

4.2 Loss distribution for an inhomogeneous pool

In this section we discuss how to compute the loss distribution of a general inhomogeneous pool. Suppose that an inhomogeneous pool consists of I homogeneous sub-pools with loss-given-defaults $LGD_1, LGD_2, \ldots, LGD_I$ sitting on a common lattice and the loss distribution for the i-th group is $(p_{i,0}, \ldots, p_{i,d_i})$, $i = 1, 2, \ldots, I$, where d_i is the upper bound on the number of defaults in group i, which implies that the largest possible loss for this group 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 groups 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 groups, $a = 0, 1, \ldots, A_i$, $A_i = \sum_{j=1}^{i} d_j LGD_j$. Then the loss distribution of the bigger pool consisting of the first i groups plus the $(i+1)$-st group 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 group needs to be mapped to $(p_0^{(1)}, p_1^{(1)}, \ldots, p_{d_1 LGD_1}^{(1)})$ by

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

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 4.1 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 homogeneous or completely inhomogeneous[2]. Since JKM exploits the property that the pool can usually be divided into a small number of homogeneous pools, we expect it to be faster than the ASB method. Performance comparisons shown in Section 6 support this conjecture.

4.3 Generalization to multiple states

In the previous section we proposed a recursive algorithm for computing the loss distribution of a sum of mutually independent Bernoulli random variables. This algorithm 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 an 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. 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 by the recursive formula

$$\mathbf{p}_{k+1} = \begin{pmatrix} p_{k+1,(M-1)(k+1)} \\ \vdots \\ p_{k+1,1} \\ p_{k+1,0} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_k & 0 & \cdots & 0 \\ 0 & \mathbf{p}_k & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \mathbf{p}_k \end{pmatrix} \begin{pmatrix} Q_{k+1,M-1} \\ \vdots \\ Q_{k+1,1} \\ Q_{k+1,0} \end{pmatrix}, \tag{16}$$

where $p_{k,j}$ denotes 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 (16) in a floating-point system. A result similar to that of Proposition 1 holds:

Proposition 2 *The error associated with the recursive formula (16) satisfies*

$$\|\epsilon_k\|_\infty \le \left(1.001^k c' - (1000M + 1)\right)\varepsilon, \qquad \text{for } k = 1, 2, \ldots, K, \tag{17}$$

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

[2]By completely inhomogeneous we mean all the LGD_k's are unique.

Numerical results comparing the efficiency of this generalized recursive method with the FFT based convolution method [5, Chapter 32] are presented in Table 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 [21], 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 algorithm is $O(M^2K^2)$. Thus the FFT based convolution method should be asymptotically faster than our recursive method.

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

The experiment was carried out for several combinations of K and M. Entries of the form x:y in Table 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 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 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 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.

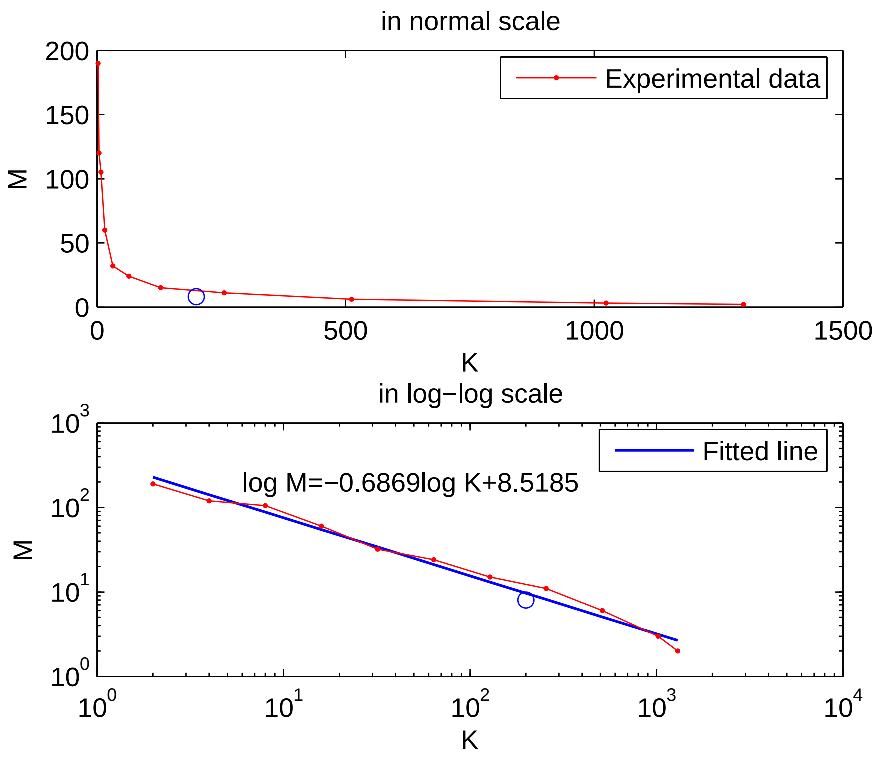

Figure 1: Comparison of computational speed between the recursive and the FFT based convolution methods

5 Approximation methods for loss distribution evaluation

In Section 4 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. For a high dispersion pool, approximation methods are preferable. A method of this kind is the compound Poisson approximation introduced for synthetic CDO valuation by De Prisco, Iscoe and Kreinin [6]. Though it is shown by the authors that this method usually gives reasonably accurate spreads compared to the Monte Carlo method, the spreads computed using this method may differ from the exact ones by as much as 20 basis points (or 0.6%) for an equity tranche. Therefore the accuracy of this approximation is not always satisfactory. As a natural extension, we introduce in subsection 5.1 the improved compound Poisson approximations of Hipp [12] for better accuracy. Both the recursive and the compound Poisson approximations require that the LGDs must sit on some common lattice, *i.e.*, the LGDs must be integer multiples of some properly chosen monetary unit. A small monetary unit may result in a high dispersion pool, for which neither of these methods works well. To ameliorate this deficiency, we introduce in subsection 5.2 the normal power approximation method to approximate the distribution of $\mathscr{L}^P = \sum_{k=1}^K LGD_k \mathbf{1}_{\{k\}}$.

5.1 Compound Poisson approximations

Instead of computing the distribution of $\mathscr{L}^P$ exactly, we can approximate it using an approximation to its characteristic 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

$$\phi_k^{(J)}(t) = \exp\left(\sum_{j=1}^{J} \frac{(-1)^{j+1}}{j} \left[Q_k \left(g_k(t) - 1 \right) \right]^j \right) \tag{18}$$

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

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

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

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

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

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

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

$$\mu_{\mathscr{L}^P} = \exp(-\lambda_1) \sum_{r=0}^{\infty} \frac{\lambda_1^r}{r!} \varphi_1^{*r}, \tag{19}$$

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 [6]. We denote it by CPA1 in this paper.

By choosing $J > 1$ in (18), 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 corresponding distribution function for $\mathscr{L}^P$ is evaluated similarly to (19) 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 paper.

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 [7].

5.2 Normal power approximation

In actuarial science, the payoff function $f(\mathscr{L}_i^P; \ell, S + \ell)$ is called the stop-loss function. It indicates that 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 of S. The distribution of $\mathscr{L}^P$ can also be approximated by the normal power (NP) distribution. Thus, $\mathbb{E}\left[f(\mathscr{L}_i^P; \ell, S + \ell)\right]$ can be expressed in terms of the cumulative distribution function Φ and the probability density function ϕ of the standard normal distribution. In this paper, we give the basic formulas only; more details can be found in [3], [15], [20].

Under the NP approximation, the expected loss of a tranche with protection level $\ell \geq 0$ and tranche size $S \geq 0$ is

$$\mathbb{E}_{SL}\left[\mathscr{L}^P; \ell, S\right] = \mathbb{E}_{SL}\left[\mathscr{L}^P; \ell + S\right] - \mathbb{E}_{SL}\left[\mathscr{L}^P; \ell\right], \tag{20}$$

where

$$\mathbb{E}_{SL}\left[\mathscr{L}^P; z\right] = (\mu - z)(1 - \Phi(y_z)) + \sigma(1 + \gamma y_z/6)\phi(y_z) \tag{21}$$

is the expected loss of a tranche with a zero attachment point and loss capped by z. The subscript SL stands for "Stop Loss", 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),

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

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 all the first three moments of the approximated distribution and also captures some other important properties of it, such as fat tails and asymmetry. The normal approximation matches the first two moments of the distribution. Thus, it is expected that the normal power approximation might approximate the true distribution better than the normal approximation.

6 Numerical Results

In this section we present numerical results that illustrate the accuracy and computational cost 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.

6.1 Two points about implementations

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

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

The second point concerns the stopping criterion for evaluating the infinite sum (19). 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 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

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

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 the tolerance $\epsilon = 10^{-4}$.

6.2 Test problems

The results presented below are based on a sample of 15 pools. For each pool, the number of names K is either 100, 200, or 400. The number of homogeneous groups in each pool is either $1, 2, 4, 5$, or $K/10$, and all homogeneous groups in a given pool have an equal number of names. The notional values for each pool are summarized in Table 2. For example, the 200-name pool with local pool ID = 3 consists of four homogeneous groups with the notional values 50, 100, 150, and 200, respectively. For convenience, we also labeled each pool with a global pool ID. For each of the 100-name pools, the global and the local IDs coincide. For each of the 200- and 400-name pools, its global pool ID (GID) is its local pool ID plus 5 or 10, respectively. For example, a 200-name 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: Selection of notional values of K-name pools

For each name, the risk-neutral cumulative default probabilities are randomly assigned one of two types, I and II, as defined in Table 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 3: Risk-neutral cumulative default probabilities

The recovery rate is assumed to be 40% for all names. Thus the LGD of name 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 interest rates are $r_1 = 4.6\%, r_2 = 5\%, r_3 = 5.6\%, r_4 = 5.8\%$ and $r_5 = 6\%$. Thus the corresponding 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%. 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 4 and 5 are based on the pricing of the first four tranches of each pool, as explained above.

6.3 Analysis of results

The accuracy results are presented in Table 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, $(2248, 928, 606, 248)$ are the spreads, evaluated by an exact method for the first four tranches of the 200-name homogeneous pool (GID=1). Since all exact evaluation 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 evaluation 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 evaluation results. For example, for the 100-name 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 both CPA2 and CPA3 produce 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 approximations is suitable for high-spread tranche pricing. If accurate results are required, the exact evaluation methods, CPA2 and CPA3 are recommended.

The computational costs are presented in Table 5. Since CPA2 is generally faster than CPA3 and NP requires almost the same computational costs as the normal approximation, we list only the computational times for each of HW, ASB, CPA1, CPA2 and NP divided by

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 4: Accuracy comparison between exact and approximate methods

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.

By considering both the accuracy and the computational cost 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 computational cost, and NP for risk management, where computational cost is generally more important than 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 5: The computational times for each of HW, ASB, CPA1, CPA2 and NP divided by that of JKM

7 Conclusions

Two types of methods for the evaluation of the loss distribution of a synthetic CDO pool are introduced in this paper. Error analysis and numerical results show that the proposed exact recursive method JKM is stable and efficient. It can be applied to CDO 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 computational cost. The normal power approximation is useful for risk management where computational cost is generally more important than accuracy.

Acknowledgments

We are grateful to Ian Iscoe for valuable comments that have improved this paper.

Appendix: Proof of Proposition 1

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 system, we have

$$\begin{aligned} & \left\| \begin{pmatrix} \mathbf{a} & 0 \\ 0 & \mathbf{a} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} - \mathrm{eval}\left(\begin{pmatrix} \mathbf{a} & 0 \\ 0 & \mathbf{a} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} \right) \right\|_\infty \\ \le\ & (2\varepsilon + \varepsilon^2) \left\| \begin{pmatrix} \mathbf{a} & 0 \\ 0 & \mathbf{a} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} \right\|_\infty \\ \le\ & (2\varepsilon + \varepsilon^2)\,(c + d)\,\|\mathbf{a}\|_\infty . \end{aligned} \tag{22}$$

Applying (22) to (12) results in

$$\begin{aligned}
&\left\| \begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} - \hat{\mathbf{p}}_{k+1} \right\|_\infty \\
\leq\ & \left(2\varepsilon + \varepsilon^2\right) \left(Q_{k+1} + \hat{S}_{k+1}\right) \|\hat{\mathbf{p}}_k\|_\infty \\
\leq\ & \left(2\varepsilon + \varepsilon^2\right) (1+\varepsilon) \|\hat{\mathbf{p}}_k\|_\infty . \qquad (23)
\end{aligned}$$

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

$$\left(2\varepsilon + \varepsilon^2\right)(1+\varepsilon) \leq 2.001\varepsilon.$$

Thus (23) can be written as

$$\left\| \begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} - \hat{\mathbf{p}}_{k+1} \right\|_\infty \leq 2.001\varepsilon \|\hat{\mathbf{p}}_k\|_\infty . \qquad (24)$$

On the other hand, using the triangle inequality we have

$$\begin{aligned}
&\left\| \begin{pmatrix} \mathbf{p}_k & 0 \\ 0 & \mathbf{p}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ S_{k+1} \end{pmatrix} - \begin{pmatrix} \hat{\mathbf{p}}_k & 0 \\ 0 & \hat{\mathbf{p}}_k \end{pmatrix} \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} \right\|_\infty \\
\leq\ & \left\| \begin{pmatrix} \mathbf{p}_k & 0 \\ 0 & \mathbf{p}_k \end{pmatrix} \right\|_\infty \left\| \begin{pmatrix} 0 \\ \varepsilon S_{k+1} \end{pmatrix} \right\|_\infty + \left\| \begin{pmatrix} \mathbf{p}_k - \hat{\mathbf{p}}_k & 0 \\ 0 & \mathbf{p}_k - \hat{\mathbf{p}}_k \end{pmatrix} \right\|_\infty \left\| \begin{pmatrix} Q_{k+1} \\ \hat{S}_{k+1} \end{pmatrix} \right\|_\infty \\
\leq\ & \varepsilon \|\mathbf{p}_k\|_\infty + (1+\varepsilon)\|\epsilon_k\|_\infty . \qquad (25)
\end{aligned}$$

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 (24) and (25) 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

$$\|\epsilon_{k+1}\|_\infty \leq 1.001\|\epsilon_k\|_\infty + 3.001\varepsilon, \text{ for } k = 1, 2, \ldots, K-1. \qquad (26)$$

Noting that $\|\epsilon_1\|_\infty \leq \varepsilon$, (26) implies that

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

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 $c = \frac{x_1(a-1)+b}{a(a-1)}$.

This completes the proof.

References

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

[2] Leif Andersen, Jakob Sidenius, and Susanta Basu. All your hedges in one basket. *Risk*, pages 61–72, November 2003.

[3] Robert Eric Beard, Teino Pentikäinen, and Erkki Pesonen. *Risk Theory: The Stochastic Basis of Insurance.* Monographs on Statistics and Applied Probability. Chapman and Hall, 3rd edition, 1984.

[4] Xavier Burtschell, Jon Gregory, and Jean-Paul Laurent. A comparative analysis of CDO pricing models. Available from http://defaultrisk.com/pp_crdrv_71.htm, April 2005.

[5] Thomas H Cormen, Charles E Leiserson, and Ronald L Rivest. *Introduction to Algorithms.* The MIT electrical engineering and computer science series. The MIT Press, Cambridge, Massachusetts, 2000.

[6] Ben De Prisco, Ian Iscoe, and Alex Kreinin. Loss in translation. *Risk*, pages 77–82, June 2005.

[7] Jan Dhaene, Bjørn Sundt, and Nelson De Pril. Some moment relations for the Hipp approximation. *ASTIN Bulletin*, 26(1):117–121, 1996.

[8] Darell Duffie and Kenneth Singleton. Modeling term structures of defaultable bonds. *The Review of Financial Studies*, 12(4):687–720, 1999.

[9] Gene H Golub and Charles F Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 2nd edition, 1991.

[10] Michael Gordy and David Jones. Random tranches. *Risk*, 16(3):78–83, March 2003.

[11] Michael T Heath. *Scientific Computing: An Introductory Survey.* McGraw-Hill, 2nd edition, 2002.

[12] Christian Hipp. Improved approximations for the aggregate claims distribution in the individual model. *ASTIN Bulletin*, 16(2):89–100, 1986.

[13] John Hull and Alan White. Valuing credit default swaps I: No counterparty default risk. *Journal of Derivatives*, 8(1):29–40, Fall 2000.

[14] John Hull and Alan White. Valuation of a CDO and an n^{th} to default CDS without Monte Carlo simulation. *Journal of Derivatives*, (2):8–23, 2004.

[15] Stuart A Klugman, Harry H Panjer, and Gordon E Willmot. *Loss Models from Data to Decisions.* John Wiley & Sons, Inc., 1998.

[16] Jean-Paul Laurent and Jon Gregory. Basket default swaps, CDOs and factor copulas. Available from http://www.maths.univ-evry.fr/mathfi/JPLaurent.pdf, September 2003.

[17] David X Li. On default correlation: A copula function approach. Technical Report 99-07, The RiskMetrics Group, 44 Wall St., New York, NY 10005, April 2000.

[18] William Press, Brian Flannery, Saul Teukolsky, and William Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[19] Philipp J Schönbucher. *Credit Derivatives Pricing Models.* Wiley Finance Series. John Wiley & Sons Canada Ltd., 2003.

[20] Jozef L Teugeles. *Encyclopedia of Actuarial Science.* John Wiley & Sons, Ltd, 2004.

[21] The Numerical Algorithmics Group. *The NAG C Libray Manual, Mark 7*. The Numerical Algorithmics Group Limited, 2002.

[22] Vanderbilt Capital Advisors. Calculating implied default rates from CDS spreads. Available from http://www.vcallc.com/mailings/additions/cdsspreads2.htm.

[23] Oldrich Vasicek. Probability of loss distribution. Technical report, KMV, February 1987. Available from http://www.moodyskmv.com/research/.