DYNAMIC DEFAULT CORRELATION MODELLING: BINOMIAL LATTICES, STOCHASTIC OPTIMIZATION AND PERFECT MATCH

TAO PENG AND ERIK SCHLÖGL
School of Finance and Economics
University of Technology, Sydney
PO Box 123
Broadway, NSW 2007
Australia
Ph: +61 2 9514 7777
Tao.Peng-1@student.uts.edu.au

ABSTRACT. In this paper, the common factor driving obligor defaults is modelled by a binomial lattice in a manner which allows model prices to match market quotes for Collateralized Debt Obligations tranches exactly. Fast calibration is achieved by a cross entropy - Nelder Mead hybrid algorithm. The precise pricing make it possible to back out market correlation information for pricing and hedging of bespoke tranches and bespoke portfolio.

1. Introduction

Since the credit model meltdown in 2005, there have been a few prominent pricing models which have improved calibration results dramatically. Most of those models are 'bottom up', common factor driven models. There are basically two type of these models - structural and intensity models. The current industry model (Li 2000) is of the first kind. It suffers from the correlation smile which comes from the assumption that the common factor is normally distributed. The gamma model proposed by (Baxter 2007), model the common factor by a levy type gamma process which improved on the Gaussian model. However, the model can’t have exact pricing because of parametrization constraint - the correlation is assumed the same for all obligors in the portfolio. The idiosyncratic parameter doesn’t govern the correlation of that obligor with the common factor. Thus there is huge model risk in hedging. An exemplar paper from the second kind of models is (Joshi & Stacy 2005) where the intensity of the common factor is modelled by a sum of gamma processes. It is in some way similar to our model. However the model doesn’t have analytical solutions and thus its application is greatly compromised. In this paper, we push these common factor driving defaults models to the limit by modelling the common factor by a binomial lattice dynamics. This non-parametric
approach gives the model the flexibility to match precisely to any market price given any dependence structure and in the same time reduces model risk. The simple design not just gives us flexibility but fast and almost error free pricing. Some results on pricing of bespoke tranches are provided at the end. However, this paper will focus on model design and calibration algorithm. The more interesting issues of pricing more exotic bespoke products and hedging issues will be provided when the results come out.

2. Model

2.1. Binomial process. The default process of an obligor is modelled by a Cox process.

\[
P\{\tau_i > t \mid (M_s), 0 < s < t\} = \exp\left\{-\int_0^t \lambda_i(M_s)ds\right\}
\]

is the survival probability of company \(i\) until time \(s\) conditional on state variable \(M\) which is scalar process representing the general economy.

Common factor \(M\) is the state variable that all companies in a credit portfolio depend on. Here \(M\) is assumed to evolve according to binomial lattice dynamics. Denote by \(M_{k,j}\) the value of the common factor at step \(k\) in state \(j\), \(1 \leq j \leq k\), with \(M_{1,1}=1\). Assume that \(M_{k,j}\) moves up to \(M_{k+1,j} = (1 + a)M_{k,j}\) with probability \(q_{k,j}\) and down to \(M_{k+1,j+1}\) with probability \(1 - q_{k,j}\). Note that while the transition probabilities can be time and state dependent, the proportional sizes of the "up" and "down" moves are not. The latter ensures that \(M\) evolves according to a recombining lattice.

For modelling, the lattice has to be put on a time line. The state value remains constant during the time between steps. Since loss distribution is calculated on payments dates, we also discretise the time line in such a manner that it matches the payment dates of a CDO structure.

The number of steps of a required binomial lattice doesn’t have to be the same as the number of payments left in the CDO. Typically for accurate enough pricing the latter can be much fewer than latter. For instance, we find that a six step binomial lattice is minimum for precise pricing of CDX or ITRAXX five year CDO tranches with 20 remaining payments.

Therefore we allow one step in a binomial lattice cover a certain number of payments in the time line. Financially it means there is no perceived economic situation change for a few payment dates. The state change for \(M\) always happens at payment dates.

Consider a discrete set of time points \(t_u, 0 \leq u \leq N\), where \(t_0 = 0\) denotes initial time ("now"), \(t_u\) is payment date and also observation time for the common factor.
$M$ and $N$ is the last number of payment. We can transform the steps-denoted $M_k$ to time-subscribed notation $M_{t_u,j}$ or vice versa in a variety of ways.\footnote{For instance, a transformation scheme $k = f(t_u)$ can be that each step covers roughly equal number of payments and therefore the payments between two steps take the $M$ value of the lower end.} Accordingly the hazard rate process is discretized. Therefore the Cox process can be represented as,

$$P\{\tau_i > t_N \mid (M_{t_u}), 1 < u < N\} = \exp\{-\sum_{u=1}^{N} \lambda_i(M_{t_u})\Delta t_u\}$$

where $t_N = \sum_{u=1}^{N} \Delta t_u$

In this model, the hazard rate has the following parametrization:

$$\lambda_i(M_{t_u}) = \lambda_{t_u,i}M_{t_u}$$

It means default hazard rate for the $i$-th obligor in the time interval $[t_{u-1}, t_u)$ is $\lambda_{t_u,i}M_{t_u}$.

\subsection*{2.2. Marginal default probabilities matching.}
Denote by $P_i(\tau > t)$ the survival probability by time $t$ of the $i$th obligor. It is the expected value of the conditional probability $P_i(\tau > t \mid M_t)$ which is given by $a$, the $q_{k,j}$ and $\lambda_{t,i}$. Thus $\lambda_{t,i}$ can be calibrated to the risk-neutral survival probabilities extracted from market spreads for credit default swaps (CDS)

\subsection*{2.3. CDO pricing.}
In order to price the tranches of collateralised debt obligations (CDO), we require the portfolio loss distribution on a portfolio of obligors up to each payment time. To calculate these, we follow the method proposed by (Andersen, Sidenius & Basu 2003)

Premium payments are taken to only occur at the payment times $T_u$ and default happens in the middle of payment times. Assuming equal recovery rates and equal exposure sizes for each name in the portfolio underlying the CDO, the present value of the default payments can be calculated if the expectation of the number of defaults up to time $t_u$ is known.

Denote by $L(t_u)$ the number of defaults up to time $t_u$, and by $L$ and $\bar{L}$ the attachment and detachment points, respectively, of a tranche on a CDO running from $t_0$ to $t_N$. Then the present value of the default payments on this tranche is

$$\text{defaultPV}(L, \bar{L}) = \sum_{u=1}^{N} B(0, t_u) E[\max(0, \min(L, (1 - \text{recoveryRate})L(t_u)) - \bar{L})]$$

where $B(0, t_u)$ is the time 0 price of a defaultfree zero coupon bond maturing in $t_u$. 

$$\lambda_{t_u,i}M_{t_u}$$

The present value of the default payments on this tranche is
Similarly, if premium payments are assumed to be made at time $t_u$ based on the remaining (undefaulted) notional of the CDO tranche at time $t_u-1$ the present value of a basis point (PVBP) of premium is

$$\text{spreadPV}(L, \bar{L}) = 10000 \cdot \sum_{u=1}^{N} \left( t_u - t_u - 1 \right) B(0, t_u) (\bar{L} - L) - E[\max(0, \min(\bar{L}, (1 - \text{recoveryRate})L(t_u - 1)) - L)]$$

Then the fair spread $\bar{s}$ (quoted in basis points) on the CDO tranche is

$$\bar{s} = \frac{\text{defaultPV}(L, \bar{L})}{\text{spreadPV}(L, \bar{L})}$$

Thus it remains to calculate the

$$E[\max(0, \min(\bar{L}, (1 - \text{recoveryRate})L(t_u)) - L)]$$

for each $t_u$. These can be calculated from the unconditional loss distribution for time $t_u$, which is given by the probabilities $P(L(t_u) = l)$ of there having been exactly $l$ defaults up to time $t_u$. The $P(L(t_u) = l)$ can be written in terms of the conditional loss probabilities as

$$P(L(t_u) = l) = \sum_{j=1}^{f(t_u)} P(L(t_u) = l|M_{t_u,j})Q_{t_u,j}$$

where $f(t_u)$ is step number for time $t_u$ since the binomial lattice is combining and $Q_{t_u,j}$ is the unconditional probability of the binomial process for $M$ being in state $j$ at time $t_u$, and the conditional loss probabilities $P(L(t_u) = l|M_{t_u,j})$ are assembled recursively by the method of (Andersen et al. 2003), which requires the individual conditional probabilities of survival to time $t_u$, $P_i(t_u|M_{t_u,j})$.

2.4. Notation simplification.

$$P\{\tau > t_u\} = P(t_u)$$

$$q(M_{k+1,j} | M_{k,j}) = q_{k,j}$$

$$q(M_{k+1,j+1} | M_{k,j}) = 1 - q_{k,j}$$

where $k = f(t_u)$

2.5. Arbitrage free pricing. Notice that, unlike static copula models, the model is automatically arbitrage free since conditional on a realization of the state variable, the survival probability of an obligor is a decreasing process.

$$P(T_{k+1} | M_{k+1}M_k \cdots M_1) = P(T_k | M_k \cdots M_1) \exp\{-\lambda_{i,k+1}M_{k+1}\Delta t_{k+1}\} < P(T_k | M_k \cdots M_1)$$
2.6. **Algorithmic construction of the model.** For pricing purpose, we are only interested in the survival probability not conditional on the path but conditional on the current state variable \( M_k \) - \( P(T_k | M_k) \). Due to the recombining feature of our binomial lattice, the derivation involves two steps.

2.6.1. *The first step.* If we already know the current state dependent survival probability \( P(T_k | M_k) \), derive a one step path dependent survival probability given that the obligor has survived to the next time step under a new state variable \( M_{k+1} \).

\[
P(T_{k+1} | M_{k+1}M_k) = P(T_k | M_k)P\left(\frac{T_{k+1} | M_{k+1}M_k}{T_k | M_k}\right)
\]

Mathematically this transitional probability is \( \exp\{-\lambda_{i,k+1}M_{k+1}\Delta t_{k+1}\} \).

**Proof.**

From

\[
P(T_{k+1} | M_{k+1}M_k \cdots M_1) = P(T_k | M_k \cdots M_1)\exp\{-\lambda_{i,k+1}M_{k+1}\Delta t_{k+1}\}
\]

The left side becomes

\[
\frac{P(T_{k+1}, M_{k+1}M_k \cdots M_1)}{P(M_{k+1}M_k \cdots M_1)} = \frac{P(T_{k+1}, M_{k+1}M_k)P(M_{k-1} \cdots M_1 | T_{k+1}, M_{k+1}M_k)}{P(M_{k+1}M_k)P(M_{k-1} \cdots M_1 | M_{k+1}M_k)}
\]

The right side becomes

\[
\frac{P(T_k, M_k \cdots M_1)}{P(M_k \cdots M_1)} = \frac{P(T_k, M_k)P(M_{k-1} \cdots M_1 | T_k, M_k)}{P(M_k)P(M_{k-1} \cdots M_1 | M_k)}\exp\{-\lambda_{i,k+1}M_{k+1}\Delta t_{k+1}\}
\]

Since past events don’t depend on future events, the two terms cancel out.

\[
P(M_{k-1} \cdots M_1 | M_{k+1}M_k) = P(M_{k-1} \cdots M_1 | M_k)
\]

\[
P(M_{k-1} \cdots M_1 | T_{k+1}, M_{k+1}M_k) = P(T_k, M_k)P(M_{k-1} \cdots M_1 | T_k, M_k)
\]

So

\[
P(T_{k+1} | M_{k+1}M_k) = \exp\{-\lambda_{i,k+1}M_{k+1}\Delta t_{k+1}\}P(T_k | M_k)
\]

2.6.2. *The second step.* Work out the new conditional survival probability \( P(T_{k+1} | M_{k+1}M_k) \) by forward induction given below.

**Initialization**

We have

\[
Q_{1,1} = 1
\]

\[
P_i(T_1) = P_i(T_1 | M_{1,1})
\]

\[
P_i(T_1 | M_{1,1}) = \exp\{-\lambda_{i,1}M_{1,1}\}
\]
Forward induction

\[ Q_{k,j} = \begin{cases} 
q_{k-1,j}Q_{k-1,j} & j = 1 \\
(1 - q_{k-1,j})Q_{k-1,j-1} + q_{k-1,j}Q_{k-1,j} & 1 < j < k \\
(1 - q_{k-1,j-1})Q_{k-1,j-1} & j = k
\end{cases} \]

See Figure 1 for a three step probability transition.

**Figure 1.** Binomial process for the common factor
where \( q(M_{tu,j-1} \mid M_{tu+1,j}) \) is the conditional probability that the process \( M \) was in state \( j - 1 \) at time \( tu \) given that it is in state \( j \) at time \( tu+1 \), which can be calculated from known probabilities using Bayes’ Theorem, i.e.

See figure 2 for a three step transition. \(^2\)

\[
q(M_{tu,j-1} \mid M_{tu+1,j}) = q(M_{tu+1,j} \mid M_{tu,j-1}) \frac{Q(M_{tu,j-1})}{Q(M_{tu+1,j})}
\]

and similarly

\[
q(M_{tu,j} \mid M_{tu+1,j}) = q(M_{tu+1,j} \mid M_{tu,j}) \frac{Q(M_{tu,j})}{Q(M_{tu+1,j})}
\]

3. Multi-objective optimization

The calibration of the model to market prices is a multi-objective optimization problem. Multi-objective optimization is concerned with the minimization of a vector of objective functions \( F_i(x) \) that can be subject to a number of constraints or bounds.

In a standard Itraxx or CDX structure, there are five tranches from equity to supersenior. The objective is to match the five model tranche prices precisely to mid-market quotes or at least fit them within their respective bid and ask spreads (Equation 1). Note that the range of values for all five prices to achieve at the same time is constraint for the fact that they are based on different sections of the same loss distribution and the loss distribution of by a certain time has a sum of 1.

\[
\min F_i(p) = |\text{price}_i - \text{quote}_i|
\]

where \( p \) is the model parameters vector with constraint \( 0 < p_j < 1 \) and \( 1 < i < 5 \).

For trading or structured products pricing, different accuracies might be required for different purposes. The advantage of our model is that any price within bid and ask spread can be achieved up to machine accuracy. For different applications, the concern here is that the rate of convergence for all tranches should be roughly kept at the same proportion. And this proportion should be the same as that of different tranche spreads. For this reason, it is natural to adopt a weighted sum strategy.

The weighted sum method converts the multi-objective problem of minimizing the vector \( F \) into a scalar problem by constructing a weighted sum of all the objective functions. The the weights for each tranche is in reverse proportion to the spreads of the tranches.

\(^2\)In the example, the number of step equals the number of payments for simplification
Therefore the following objective function is objective function.

$$\min F = \sum_i \frac{|\text{price}_i - \text{quote}_i|}{\text{spread}_i}$$

To achieve different accuracies, for instance, if we want to fit within a specific spread range, we design the second objective function below. As soon as all prices are fit within desired bounds, the objective function becomes zero and calibration stops.
\[
\min F = \sum_i \max \left( \frac{|s_i - p_i|}{\text{spread}_i} - a, 0 \right) \tag{1}
\]

where \(a\) is the threshold, e.g. if \(a = 1\), it means we try to fit them between quoted market spread.

4. Stochastic optimization

The non-parametric design of the model poses as a highly nonlinear and highly dimensional problem with possibly many local minima. It is well-suited for a class of stochastic optimization algorithms. Stochastic optimization algorithms include local search algorithms such as Nelder-Mead (ND) Simplex method and global optimization ones like genetic algorithms. Here adopted a new global stochastic optimization method - Cross Entropy (CE) method, which is very robust but hasn’t been adopted widely. We show that combine the CE method with local search ND method we can speed up the whole calibration procedure dramatically.

First we give a brief introduction on both CE and ND methods. Then we show the testing results for CE method alone and that how we reduce the speed substantially by combining the two.

4.1. Cross entropy method. The Cross Entropy (CE) method (Rubinstein & Kroese 2004) involves an iterative procedure where each iteration can be broken down into two phases:

- Generate a random data sample according to a specific mechanism.
- Update the parameters of the random mechanism based on the data to produce a “better” sample in the next iteration.

Applied to our model, the procedure can be specifically described in the following steps.

1. Randomize the deterministic optimization problem by defining a probability distribution - beta distribution, for all \(q_{kj}\) and for the relative change \(a\) for the common factor \(M\). Denote the vector of distribution parameters by \(v\).
2. Set the iteration counter \(t = 1\) and \(\alpha = 1, \beta = 1\) for the beta distribution. i.e. it starts as uniformly distributed.
3. Generate \(n\) parameter vectors \(v_t\) according to a beta density function \(f(\cdot, v_t)\).
4. Calculate the value of the objective function for each of these sets of model variables. Rank these sets in sequence by the objective function values. Take the \(\rho\)-quantile of these values (for an exogenously chosen level \(\rho\)).
5. If the top candidate falls under the desired threshold, then terminate the iteration. Otherwise calculate the mean and variance of \(\rho\)-quantile of the model values and work out the new \(\alpha\) and \(\beta\) of the beta distribution parameters for the next iteration.
(6) Test the variance against a convergence criteria (by experience). If it is small enough, terminate iteration otherwise go to step (3).

4.1.1. **CE Results.** Using objective function 1 as target and setting 4 times spreads as initial thresholds, the test is done by halving the threshold every time the objective function passes through it. Table 4.1.1 shows how many iterations it takes for the algorithm to pass through the thresholds. Here we take $n = 100$ and $\rho = 10\%$.

<table>
<thead>
<tr>
<th>Thresholds: a ($4 \times$ spread)</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1 1 2 1 1 1 2 1 1 1 2 1</td>
</tr>
<tr>
<td>2</td>
<td>6 6 14 4 9 5 2 8 7 9 14 5</td>
</tr>
<tr>
<td>1</td>
<td>47 31 28 33 29 8 13 19 26 10 28 15 32 21 51</td>
</tr>
<tr>
<td>0.5</td>
<td>58 43 59 49 56 58 56 46 53 27 35 43 48 56 65</td>
</tr>
<tr>
<td>0.25</td>
<td>71 73 71 74 67 84 71 72 73 64 71 74 63 69 82</td>
</tr>
<tr>
<td>0.125</td>
<td>82 77 93 83 96 84</td>
</tr>
</tbody>
</table>

**Table 1. Calibration results with CE algorithm**

Figure 3 shows further details of the stochastic process that the two objective functions converge towards market quotes.

5. **Nelder Mead Simplex method**

The Nelder-Mead (ND) method makes use of a construct called a *simplex*, which is a polytope of $N + 1$ vertices in $N$ dimensions.

The Rules

- Reflect the point with the highest WSS through centroid (center) of the simplex
- If this produces the lowest WSS (best point) expand the simplex and reflect further
• If this is just a good point start at the top and reflect again
• If this the highest WSS (worst point) compress the simplex and reflect closer

These rules are repeated until the convergence criteria are met.

5.0.1. ND Results. The calibration results with ND method is not provided here for comparison for the fact that being a local search algorithm convergence to global optimum is not guaranteed for ND method. The success of the search largely depends on where the search starts. If it starts in the proximity of a global optimum, the search will converge very quickly. Otherwise it might get stuck in a local optimum and can’t get out. Therefore calibration results with ND method are unreliable for comparison.

On the other hand, however, since the ND algorithm is faster than the CE method to converge near a global optimum, we can combine these two methods in our calibration algorithm design.

6. Hybrid Algorithm

Hybridizing the ND method with CE method can greatly speed up convergence by taking the advantages of both. The CE method is a very stable in searching global optimization. It is generally slower than local search algorithm and the speed also depends on the specific problem structure. We combine the two methods the following way.

Firstly define a set of thresholds in our objective function. In practice our calibration goal might be to fit within any of these thresholds and any local minimum that falls under that thresholds is then seen as a global minimum for that optimization search.

Next start with the CE algorithm. At the end of each iteration, if one or a number of solutions passes a certain threshold, the best candidate is kept and later on passed on to the ND algorithm.

Then terminate the CE method after it passes a desired number of thresholds. Finally run ND algorithm from local solutions at the end of each thresholds and test how far the ND algorithm can take us.

One of the aims is to identify when the CE search goes within a global optimum. It is problem specific and can only be identified by testing. After a certain number of runs, we can roughly find out which threshold as a near global optimal point boundary and that is the threshold to pass before a local search is initiated.

7. Hybrid Results

The results are shown in table 2 to table 5. Each table shows one run of the CE method taking results from table 4.1.1 and conduct local search after passing each threshold. Notice here the number of iterations is replaced with the number of function evaluations (Pricings). Each row shows how far a local search from that
The threshold will lead towards global optimum. The results confirm the observation that a local search algorithm is generally faster in convergence but can be trapped in a local minimum. For example, in the second row of Table 2, it takes one iteration or 100 function evaluations to fit within four times bid and ask spread. From that point, it takes 665 evaluations for the ND algorithm to fit within bid and ask spread whereas it takes a further $1000 - 100 = 900$ function evaluations for the CE method to do so. However, ND method can’t go any further from here.

It also shows interesting results of how CE algorithm get out of a local optimum. At line 3 where the CE method passes through bid and ask spread, from there the ND algorithm doesn’t go any further. It suggests the solution might be near a close by local minimum. However, the CE method takes us to further down the thresholds ladder and therefore suggests that it get out of that local minimum point.

From results of all the tables, one can observe that with the CE algorithm as soon as the evaluation falls below bid and ask spread it is safe to pass it on to the local search algorithm and quickly converge which suggests that global optimum might be near bid and ask spread.

### Table 2. First Run

<table>
<thead>
<tr>
<th>Tresholds</th>
<th>No of Fn Eval</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>0.5</th>
<th>0.25</th>
<th>0.125</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>200</td>
<td>1</td>
<td>389</td>
<td>665</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>900</td>
<td>1</td>
<td>1</td>
<td>103</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>2700</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>89</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>6400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>8200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3. Second Run

<table>
<thead>
<tr>
<th>NO. of function evaluations</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>0.5</th>
<th>0.25</th>
<th>0.125</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>200</td>
<td>1</td>
<td>260</td>
<td>461</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1400</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2800</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>421</td>
<td>502</td>
</tr>
<tr>
<td>0.5</td>
<td>3500</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>115</td>
<td>236</td>
</tr>
<tr>
<td>0.25</td>
<td>7100</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.125</td>
<td>7700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 8. Pricing of Structured Credit Products

Precise pricing of all quoted tranches with one set of parameters gives our model a huge advantage over most other models which can only do proximate pricing. Without precise calibration bespoke tranches and bespoke portfolios can’t be properly priced. However the flexibility of the model also means there are likely more solutions for one set of market prices. This is actually an advantage of the model.
Since the hedging activities are model dependent, calculated market risk are different for different models. More solutions can give traders a range of risk profiles which are all arbitrage free.

Pricing bespoke tranches have two parts - pricing bespoke detachment points and bespoke maturities.

8.1. Bespoke detachment points. Once calibrated to standard tranches, bespoke tranches can be priced. Table 6 shows the results. It is interesting to see that the unquoted tranche can be priced once calibration is achieved.

To test model stability, a 'shock' is normally given to a particular tranche and hold other tranches fixed. The unquoted tranche has to change for the fact that the loss distribution has a sum of 1.

Table 7 shows what is the percentage change to the unquoted tranche if one of the traded tranche prices is raised one percent and keep the rest unchanged.

Table 8 shows if the loss distributions backed out from 7 are used to price bespoke tranches, what are the changes to those tranche prices.
Table 7. Perturbation

<table>
<thead>
<tr>
<th></th>
<th>0-3</th>
<th>3-6</th>
<th>6-9</th>
<th>9-12</th>
<th>12-22</th>
<th>22-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>-15.93%</td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>-4.82%</td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>0%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>-1.56%</td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>1%</td>
<td>0%</td>
<td>-1.00%</td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>1%</td>
<td>-1.14%</td>
<td></td>
</tr>
</tbody>
</table>

Table 8. Perturbation

<table>
<thead>
<tr>
<th>B Tch(%)</th>
<th>0-1.5</th>
<th>1.5-4.5</th>
<th>4.5-7.5</th>
<th>7.5-10.5</th>
<th>10.5-17</th>
<th>17-27</th>
<th>27-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>0.45%</td>
<td>0.78%</td>
<td>-0.39%</td>
<td>0.24%</td>
<td>-0.09%</td>
<td>-3.30%</td>
<td>-35.53%</td>
</tr>
<tr>
<td>3-6</td>
<td>-0.27%</td>
<td>0.83%</td>
<td>0.61%</td>
<td>-0.21%</td>
<td>0.06%</td>
<td>-0.87%</td>
<td>-11.73%</td>
</tr>
<tr>
<td>6-9</td>
<td>-0.22%</td>
<td>0.42%</td>
<td>0.30%</td>
<td>0.77%</td>
<td>-0.20%</td>
<td>-0.17%</td>
<td>-4.27%</td>
</tr>
<tr>
<td>9-12</td>
<td>0.16%</td>
<td>-0.25%</td>
<td>-0.19%</td>
<td>0.71%</td>
<td>0.34%</td>
<td>-0.24%</td>
<td>-2.66%</td>
</tr>
<tr>
<td>12-22</td>
<td>0.17%</td>
<td>-0.89%</td>
<td>2.82%</td>
<td>-1.78%</td>
<td>1.30%</td>
<td>0.41%</td>
<td>-4.12%</td>
</tr>
</tbody>
</table>

8.2. Bespoke maturity.

9. Bespoke portfolio

10. Hedging/risk management

11. Conclusion and further research

The model has huge advantages over most of the current models. The accurate pricing makes better risk management practices possible for bespoke tranches. The non-parametric approach also poses the problem that it is not certain how many steps on the lattice is needed for all tranche prices. Sometimes it has to be tested a few times for a particular set of prices. However it is not a concern for the ultimate usage of the model. The model needs to be further tested on its stability in pricing bespoke tranches. With the flexibility of the model, it is possible to satisfy the requirements of most trading needs. More exotic products can be priced with the model and will be provided in due course.

References


