All your hedges in one basket

Leif Andersen, Jakob Sidenius and Susanta Basu present new techniques for single-tranche CDO sensitivity and hedge ratio calculations. Using factorisation of the copula correlation matrix, discretisation of the conditional loss distribution followed by a recursion-based probability calculation, and derivation of analytical formulas for deltas, they demonstrate a significant improvement in computational speeds.

In a traditional synthetic collateralised debt obligation (CDO), the arranger tranches out credit losses on a pool of credit default swaps (CDSs) and passes them through to different investors. Assuming that investors for all tranches can be identified, the arranger is typically left with fairly moderate market exposure. For various reasons, placing the entire pool capital structure is increasingly difficult and many recent credit basket derivatives expose the dealer to significant market risk. For instance, the recent single-tranche CDO (STCDO) product involves the sale of a single CDO tranche to a single customer, leaving it to the arranger to manage the risk of the remaining capital structure. As STCDs and other custom products offer significant customer benefits and are much less difficult to originate than traditional CDOs, such products are likely to become more important. This is especially true for managed trades where the custumer has certain rights to alter the composition of the reference portfolio over time. A basic prerequisite for active management of the risk of a credit basket derivative is the ability to accurately calculate the sensitivity of the security with respect to market and model parameters, most prominently the par CDS spreads of the underlying reference pool. The numbers of such sensitivities can be very large — many thousands — and can put considerable strain on computing resources. Moreover, the calculation of each of these derivatives can be significantly more challenging than calculating the price. For instance, in the popular Monte Carlo method, one often finds that the relative accuracy of CDS spread sensitivities can be hundreds of times lower than the relative price precision.

In this article, we discuss a number of practical techniques to improve and speed up sensitivity computations of CDOs, STCDs and other credit basket securities. The discussion is broken into two parts, depending on the complexity of the security in question. In the main line of discussion, we tackle the class of relatively simple derivatives that allow for fast quasi-analytical approximations based on factorisation and discretisation of the portfolio loss distribution. As we show, CDOs and STCDs belong to this class and allow for the application of extremely efficient recursions for the calculation of both prices and hedge parameters. Despite having received little attention in the literature so far, recursive algorithms for credit baskets appear to be increasingly popular with practitioners. We expose the method in detail here and back up its practical usage with efficient algorithms for loss unit discretisation and factor reduction of arbitrary correlation matrices.

In another line of discussion, we investigate whether techniques used for the quasi-analytical approximations are helpful in a Monte Carlo setting. Monte Carlo techniques are inherently slower than the quasi-analytical technique mentioned above, but can handle a larger number of exotic variations on the basic CDO/STCDO theme.

Notation and model

Consider a portfolio of credit default swaps on N different companies, each associated with a term curve of CDS par spreads and a recovery rate. For credit k, let I_k and R_k be the CDS notional and recovery rate per unit notional, respectively; all recovery rates are assumed constant for simplicity. Further, define p_k(T) as the risk-neutral default probability measure denoted of default hazard rate function. The functions p_k(T), k = 1, ..., N can be bootstrapped from market data.

Equation (1) fully establishes the risk-neutral marginal distribution of each default time τ_k. To construct the joint distribution of all default times, we need to use a Student-t copula, which we will define for reference. Defining vectors \( \mathbf{\tau} = (\tau_1, ..., \tau_N)^T \) and \( \mathbf{T} = (T_1, ..., T_N)^T \), the joint default time distribution in the Student-t copula, becomes:

\[
Q(\mathbf{\tau} \leq \mathbf{T}) = Q_t \left( Q_v \left( p_1(T_1) \right) \cdots Q_v \left( p_N(T_N) \right) \right)
\]

Where \( Q_v \) is the risk-neutral probability measure and \( \lambda_k \) is a (forward) default hazard rate function. The functions \( p_k(T) \) can be bootstrapped from market data.

According to (4), a Student-t random variable is defined as a Gaussian variable with fatter tails than the Gaussian distribution and is known to generate tail dependence in the joint distribution. Econometric evidence suggests that fat tails are more likely than the Gaussian distribution and are generated to known tail dependence in the joint distribution. Econometric evidence suggests that fat tails are more likely than the Gaussian distribution and are generated to known tail dependence in the joint distribution.

According to (4), a Student-t random variable is defined as a Gaussian variable with fatter tails than the Gaussian distribution and is known to generate tail dependence in the joint distribution. Econometric evidence suggests that fat tails are more likely than the Gaussian distribution and are generated to known tail dependence.

At its most abstract level, a credit basket security is simply a series of payouts given by a functional dependence on the realised value of \( \mathbf{\tau} \). Letting this function be denoted \( f(\mathbf{\tau}) \), the time 0 value of the security can, by standard arbitrage theory, be written as:

\[
V(0) = E[f(\mathbf{\tau})]
\]

where \( V(0) \) is the time 0 value of the security and \( E[f(\mathbf{\tau})] \) is the expected value of \( f(\mathbf{\tau}) \).

Notation and model

Consider a portfolio of credit default swaps on N different companies, each associated with a term curve of CDS par spreads and a recovery rate. For credit k, let I_k and R_k be the CDS notional and recovery rate per unit notional, respectively; all recovery rates are assumed constant for simplicity. Further, define p_k(T) as the risk-neutral default probability measure on [0, T] for credit k. With \( \tau_k \) denoting the default time of default for company k, we thus have:

\[
p_k(T) = Q(\tau_k \leq T) = 1 - e^{-\int_0^T \lambda_k(u) \, du}
\]
where \(E(\cdot)\) is the risk-neutral expectation operator. Notice that we implicitly assume that \(f\) contains the necessary discounting of future cashflows.

With our assumption (2) about the distribution of \(\mathbf{\tau}\), the expectation in (5) can be written explicitly. Specifically, differentiation of (2) produces the density \(\varphi\) of the default time vector \(\mathbf{\tau}\):
\[
\varphi(\mathbf{T}) = C_N \left(1 + v^{-1} \mathbf{z}^\top \Sigma^{-1} \mathbf{z}\right)^{-\frac{N}{2}} \prod_{k=1}^{N} \lambda_k \left(1 - p_k(\mathbf{T}_k)\right) \eta_{\nu} \left(z_\nu \right)^{-1}
\]
where \(z_\nu = r_{\nu,k}^{-1}(p_k(\mathbf{T}_k))\). From (5), then:
\[
V(0) = \int_{\mathbb{R}^N} f(\mathbf{T}) \varphi(\mathbf{T}) \, d\mathbf{T}
\]

As it stands, (7) is an \(N\)-dimensional integral over all default times and, for a general payout, will have to be evaluated by Monte Carlo methods. However, in a number of cases of practical importance, the complexity of the problem can be reduced significantly by recasting the payout in terms of cumulative portfolio losses. To provide some notation for this idea (for which we shall provide details later), define the aggregate portfolio loss \(\Lambda(t)\) as the sum of all credit losses on horizon \([0, t]\),
\[
\Lambda(t) = \sum_{k=1}^{K} I_{\Lambda_k}(1-R_k).
\]
where \(I_{\Lambda_k}\) is a deterministic function.

As it turns out, the price of many credit basket derivatives can be related directly to expectations of the form:
\[
E\left[\Lambda_{\max}(t)\right]
\]
where \(A\) is some known function. Examples of this will be given shortly. Calculation of (8) involves construction of the discrete distribution of \(\Lambda(t)\). For tractability, it is often useful to coarsen the loss distribution through a loss unit, \(u\). Then, the \(k\)th company in the reference pool is assigned a (non-negative integer) loss weight, \(w_k\), calculated by rounding \((1-R_k)I_{\Lambda_k}/u\) to the nearest integer. The maximal loss \(\Lambda_{\max}\) is \(u \times \Lambda_{\max}\), where \(I_{\Lambda_k}\) is the sum of all the weight losses. This discretisation will in general lead to rounding errors (except in simple cases such as, for example, when \(I_{\Lambda_k}(1-R_k)\) is independent of \(k\)) and the loss unit will have to be chosen small enough to keep these tolerable. On the other hand, the total computational effort is inversely proportional to the loss unit so a compromise must be found.

See the Appendix for one approach to this.

With the loss unit procedure, we have discrete loss probabilities \(P(l; t)\), \(l = 0, 1, \ldots, I_{\Lambda_{\max}}\) and (8) is approximated by:
\[
E\left[\Lambda_{\max}(t)\right] = \sum_{l=0}^{I_{\Lambda_{\max}}} A(l) P(l; t)
\]
(9)

If defaults of the individual companies in the reference basket are independent, the construction of the loss distribution is most efficiently accomplished by the following recursive argument (see Pykhtin & Dev, 2003, for a special case of the recursion). Suppose we know the loss distribution \(P^K(l; t), l = 0, \ldots, I_{\Lambda_{\max}}\) for a reference pool of some size \(K \geq 0\). Now suppose we add another company to the pool with loss weight \(W_{K+1}\) and known default probability \(P_{K+1}(l; t)\). Then using the independence of defaults we find for the loss distribution of the larger basket:
\[
P^{K+1}(l; t) = P^K(l; t) P_{K+1}(l; t) + W_{K+1} \times \left(1 - P^K(l; t) \right)
\]
(10)

This recursive relation can be used to build the loss distribution \(P(l; t) = P^0(l; t)\) from the boundary case of the empty basket \(P^K(l; t) = 0\) where Kronecker’s delta \(\delta_{jk}\) is one for \(i=j\) and zero otherwise. Note that the asymptotic complexity order of the recursion is \(O(N^{K+1})\). Since an increase in basket size will usually lead to the same relative increase in the maximal loss, the cost of building the conditional loss distribution grows as roughly the square of the basket size. One might think that this could be improved by using Fourier transform methods (see, for example, Gregory & Laurent, 2003, and Merino & Nyfeler, 2002). However, due to the computational burden associated with evaluating the characteristic function, this turns out to be often significantly slower than the recursion. For the examples in this article, we found that the Fourier approach (with code taken from the IMSL C standard library) involved roughly five to 25 times more computing time than the approach shown here.

In the Student-\(t\) copula framework, we obviously do not in general have the independence of defaults required for (10). As we shall discuss below, however, often we can approximate the co-dependence structure by one of conditional independence.

**Risk management of credit baskets**

As is the case for all derivatives, risk management of credit basket derivatives is a matter of combining the basket portfolio with a dynamic portfolio of plain vanilla securities (CDSs, coupon bonds, asset swaps, etc) such that overall sensitivities to key exposure variables are within acceptable bounds. For a credit basket, the primary exposures are obviously to default-related variables such as the CDS credit spreads and recovery rates of the individual companies in the CDS pool. Of these, the former is typically the most important for investment-grade companies in the pool, with the latter gaining in importance as credit spreads increase, and the trader needs to deal with the likelihood of having to physically settle the CDS.

Of the key hedge parameters, the credit spread sensitivities are generally the most challenging to calculate numerically and shall consequently receive most of our attention. We note that credit spreads enter the valuation formula (7) through the dependence of (6) on the hazard rate curves \(\lambda_k(\cdot)\), which again are bootstrapped out of the CDS credit spread quotes observed in the market. By applications of the chain-rule, we can always convert hazard rate sensitivities into credit spread sensitivities; we shall focus solely on the former here. For flexibility, we here define sensitivities to hazard rates as Gateaux (functional) derivatives with respect to arbitrary perturbation functions. For instance, consider a hazard rate curve \(k\) and define a deterministic perturbation function \(\xi(\cdot)\). Then we are interested in calculating the derivative:
\[
\frac{\partial}{\partial \xi} V = \int_{\mathbb{R}^N} V(0; \lambda_1, \ldots, \lambda_n + \mu \xi_1, \ldots, \lambda_n + \mu \xi_n) \, d\nu(\xi)
\]
(11)

where \(\mu\) is a scalar and \(\xi_j\) is the \(j\)th natural basis vector in \(\mathbb{R}^N\). To convert such quantities to a finite number of derivatives with respect to credit spreads at various maturities, we would need to calculate the Gateaux derivatives for exactly \(n\) different spanning perturbation functions \(\xi(\cdot)\), \(j = 1, \ldots, n\). These could, for instance, be chosen to be piecewise flat:
\[
\xi_j(t) = \xi_j^0 + \xi_j^1 \cdot \text{sgn}(t - s_j)
\]
However, due to jumps/gaps in credit spreads and to high market friction, which makes frequent rehedging impractical, a successful hedging strategy may take into account the sensitivities to finite-size changes in spreads. Such sensitivities can be calculated efficiently with the techniques presented in this article, although for brevity we have chosen to focus on the calculation of spread deltas. Another important risk factor is the basket correlation matrix. This should ideally be implied from the market, but this is not (yet) practical, and it is therefore important to be able to calculate the impact of a possible misspecification. Our techniques rely only on generic properties of basket correlation matrices and so are ideally suited – in conjunction with the speed of valuation – to allow calculation of implied correlation as well as the impact of correlation changes.

**Factor reduction of correlation matrices**

It has been noted in many recent articles (see, for example, Merino & Nyfeler, 2002, and Frey & McNeil, 2001) that specialising to correlation matrices with a factor structure can yield significant improvements in speed for many basket default securities. In this section, we shall briefly review the concept of factor structure, and then move on to the important practical question of how to calculate an optimal factor structure approximation to any correlation matrix.

Going back to equation (4), we recall that a central piece of our model is an \(N\)-dimensional standard Gaussian vector \(\mathbf{Y}\) with correlation matrix \(\Sigma\). In
practice, \( \Sigma \) is typically estimated by factor analysis or regression of equity returns. We say that \( \Sigma \) has an \( M \)-dimensional factor structure if we can write:

\[
Y = cX + \epsilon
\]

\[ (12) \]

where \( X \) is an \( M \)-dimensional vector of independent standard Gaussian variables (the 'factors'), \( c \) is an \( N \times M \) loading matrix and \( \epsilon \) is an \( N \)-dimensional vector of independent zero-mean Gaussian variables with variances less than one. Our hope is that \( M < N \), either exactly or to a good degree of accuracy. For the empirical correlation matrices one encounters automatically guard against the constraint that all diagonal elements of \( \Sigma \) be less than one. Our hope is that \( M = 1 \) or 2 provides sufficient accuracy. In rare cases – particularly for small baskets – it might be necessary to use \( M = 3 \) or higher.

Let us turn to the problem of calculating, for given \( M \), a weighting matrix \( c \) such that the correlation matrix generated by the right-hand side of (12) is as close as possible to a given \( \Sigma \). Using a least-squares (Frobenius) norm on all non-diagonal elements, this problem reduces to solving the minimisation problem:

\[
\min_{c} \text{tr} \left( \Sigma - cc^T - F \right) \left( \Sigma - cc^T - F \right)^T
\]

\[ (13) \]

where \( \text{tr} \) is the usual matrix trace operator (sum of diagonal) and \( F \) is a diagonal matrix ensuring that the diagonal of the factor-reduced correlation matrix is one throughout, that is:

\[
diag F = 1 - diag \, cc^T
\]

\[ (14) \]

Note that (12) implies that all diagonal elements of \( cc^T \) are less than or equal to one and we therefore must solve (13) with this constraint.

Solution of (13) can, in principle, be obtained by brute force through the application of a multi-dimensional constrained optimisation algorithm. For large baskets, however, the dimension of the optimisation problem can exceed 1,000, making such methods impractical. Instead, we can use the fact that for a given \( F \), the solution to (13) can be found by principal components analysis (PCA). Specifically, let \( E \) be the matrix of normalised (column) eigenvectors of \( \Sigma - F \); then the solution to (13) is \( c = E \Lambda^\frac{1}{2} \), where \( \Lambda \) is a diagonal matrix containing the \( M \) largest eigenvalues of \( \Sigma - F \). The solution found this way, however, does generally not satisfy (14), so we introduce a few additional steps.

\[ \square \]

\[ \square i) \] perform a PCA decomposition of \( \Sigma - F \); yielding \( c = E \Lambda^\frac{1}{2} \).

\[ \square ii) \] calculate \( F_0 \) from (14).

\[ \square iii) \] stop when \( F^{(k+1)} \) is sufficiently close to \( F^{(0)} \) (in a least-squares sense). It is not difficult to show that, for an arbitrary starting guess of \( F \), \( \text{tr} (F^{(0)} - F^{(k+1)})^2 \to 0 \) for large \( k \), as desired. In practice, only a few iterations are necessary. We note that the algorithm above does not automatically guard against the constraint that all diagonal elements of \( cc^T \) are strictly less than one. While the algorithm above will rarely violate the constraint in practice, if necessary the constraint can be satisfied by suitably adjusting the number of factors.

**STCDO pricing from loss distribution**

We now give an example of the portfolio loss approach discussed earlier by applying it to the pricing of STCDOs (and, by extension, CDOs). To first properly define the STCDO payout, we define a loss tranche of a pool by applying it to the pricing of STCDOs (and, by extension, CDOs). To illustrate the application of a multi-dimensional constrained optimisation algorithm.

The floating (or 'asset' or 'option') leg of the STCDO is given by:

\[ \alpha \theta_i \left( \frac{1}{2} (I_{\text{tranche}}(L - c_i + e_{i-1}) + I_{\text{tranche}}(T_i) - e_i) \right) \]

\[ (15) \]

Introducing a time zero zero-coupon curve \( K(i) \), we can now finally write the time zero value of the STCDO (as seen from the fixed leg receiver) as:

\[
V(0) = \sum_{i=1}^{n} \alpha \theta_i D(t_i) \left( H - L - \frac{1}{2} (e_i + e_{i-1}) \right) - \sum_{i=1}^{n} D(t_i) \left( e_i + e_{i-1} \right)
\]

\[ (16) \]

To calculate (16), we now turn to the problem of estimating the quantities \( e_i \) defined in (15). From equation (9) we get:

\[
e_i = \sum_{i=1}^{n} \left( \min \left( L + \frac{1}{2} \sqrt{n - l_i}, H - \frac{1}{2} \sqrt{n - l_i} \right) - P_i(t; T_i) \right) - \sum_{i=1}^{n} P_i(t; T_i)
\]

\[ (17) \]

where \( l_i := \min \left( \max \left( l_i, 0 \right) \right) \) and \( l_i := \max \left( \min \left( l_i, H \right) \right) \). It now only remains to calculate the loss probabilities \( P_i(t; T_i) \).

Assume now that we are working in a Student-\( t \) copula where the correlation matrix has an \( M \)-dimensional factor structure of the form (12). Define \( \Omega = (X, \theta) \) where \( \theta \) is the scalar random variable in (4). The probability of default of the \( k \)th company in the CDS pool conditioned on \( \Omega \) is given by:

\[
p_k(t | \Omega) = \Phi \left[ \frac{\sum_{i=1}^{n} \left( \min \left( l_i, H \right) - P_i(t) \right)}{\sqrt{1 - c_i^2}} \right]
\]

\[ (18) \]

where \( c_i \) are the relevant factor loadings (i.e. \( k \)th row of the load matrix \( c \) in (12)), and \( \Phi \) is the cumulative Gaussian distribution function.

Now let \( P(\Omega; \Omega) \) be the portfolio loss distribution conditioned on \( \Omega \). Given \( \Omega \), all companies in the pool are easily seen from (4) and (12) to be independent, allowing us to use the recursion formula (10) for \( P(\Omega; \Omega) \) once we substitute \( p_k(t | \Omega) \) for \( p(t) \). The unconditional loss distribution can be obtained by

\[ \square \]

Note that \( M \) is unrelated to the rank of the correlation matrix and that approximating \( \Sigma \) by a factor structure (as we shall do below) cannot be accomplished simply by a truncated eigenvalue analysis (principal components analysis). The key here is the fact that equation (12) allows for arbitrary residuals, whereas a principal components analysis would set all residuals to zero.

\[ \square \]

In our notation, \( \lfloor x \rfloor \) denotes the integer part of the real number \( x \). Note that \( 1 \leq i \leq k \)

---

**A. CPU times for factor computation**

<table>
<thead>
<tr>
<th>( N )</th>
<th>( N = 25 )</th>
<th>( N = 50 )</th>
<th>( N = 100 )</th>
<th>( N = 200 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M = 1 )</td>
<td>0.01</td>
<td>0.09</td>
<td>0.55</td>
<td>4.32</td>
</tr>
<tr>
<td>( M = 2 )</td>
<td>0.06</td>
<td>0.23</td>
<td>1.26</td>
<td>7.42</td>
</tr>
<tr>
<td>( M = 3 )</td>
<td>0.09</td>
<td>0.34</td>
<td>1.80</td>
<td>10.49</td>
</tr>
</tbody>
</table>

Notes: CPU times for factor computations on historically estimated correlation matrices. All times in seconds on a 1GHz Pentium IV processor.
then be calculated by integrating out \( \Omega \):

\[
P(t; \mathbf{x}) = \int_{\mathbb{R}^n} P(t; \mathbf{\Omega}) q(\mathbf{\Omega}) d\mathbf{\Omega}
\]

where the joint density of \( \mathbf{\Omega} \), \( q(\mathbf{\Omega}) \), is simply the product of \( n \) standard Gaussian densities and the one-dimensional chi-square density with \( v \) degrees of freedom. This \( M + \ldots + 1 \)-dimensional integral can easily be done by Gaussian quadrature, after which we can complete, as desired, the calculation of equation (17). The use of Gaussian quadrature implies that the computational workload is multiplied for each additional correlation factor by the number of integration nodes for that factor. Fortunately, it turns out that the number of nodes required for good accuracy decreases with the number of factors.

Finally, notice that the popular Gaussian copula is nested inside Student-\( t \) copula. To recover the Gaussian copula, in equation (18) replace \( t \) with \( \Phi^{-1} \) and fix \( g = v \). In the calculation of \( P(t) \) we can then omit the integration over \( g \), making the Gaussian copula faster to evaluate than the Student-\( t \) copula.

**Sensitivity calculations in the portfolio loss framework**

At the heart of the technique used in the previous section is the evaluation of conditional expectations of the form:

\[
E(A(\mathbf{\Omega})) = \lim_{M \to \infty} A(i) P(i; \mathbf{\Omega})
\]

where we have suppressed the time argument for simplicity. Calculating sensitivities of basket derivatives in the portfolio loss approach ultimately amounts to calculating sensitivities of the expectation above, a problem to which we now turn.

Specifically, consider the calculation of the Gateaux hazard rate sensitivities (defined earlier) of (19) for \( \lambda \to \lambda + \mu \xi \), for a fixed value of \( k \). We write:

\[
\frac{\partial E(A(\mathbf{\Omega}))}{\partial \mu} = \frac{dP_i(\mathbf{\Omega})}{d\mu} \frac{\partial E(A(\mathbf{\Omega}))}{\partial P_i(\mathbf{\Omega})} = \frac{dP_i(\mathbf{\Omega})}{d\mu} \left[ \sum_{k=0}^{i} \frac{\partial P(\mathbf{\Omega})}{\partial P_k(\mathbf{\Omega})} \right] (20)
\]

where we have used the fact that \( \mu \) will not affect \( P_i(\mathbf{\Omega}) \) for \( j \neq k \). We calculate (20) in two steps corresponding to the two factors on the right-hand side. First note that since \( \frac{dP_i(\mathbf{\Omega})}{d\mu} \) is zero for \( j \neq k \), we have \( \frac{dP_i(\mathbf{\Omega})}{d\mu} = \frac{dP_i(\mathbf{\Omega})}{d\mu} \times \frac{dP_i(\mathbf{\Omega})}{d\mu} \), where:

\[
\frac{dP_i(\mathbf{\Omega})}{d\mu} = \left(1 - p_i\right) \int_0^T \xi(t) dt
\]

This equation follows from the fact that a shock \( \mu \xi(t) \) to the \( k \)th hazard rate will multiply the survival probability \( q_k = 1 - p_k \) by a factor \( (1 - \mu \xi(t) dt) \). From equation (18), we also get by a direct calculation:

\[
\frac{dP_i(\mathbf{\Omega})}{d\mu} = \frac{\sqrt{v}}{2\pi} \frac{\left(1 - p_i\right) B\left(\frac{1}{2}, \frac{v}{2}\right)}{\sqrt{\left[1 - c_k e_k^2\right]}} \int_0^T \xi(t) dt
\]

\[
\left(\frac{v + (\mu - p_i)}{v}\right)^{\frac{v}{2}} \exp\left(-\frac{1}{2} \left(\frac{\left(\mu - p_i\right) - c_k X_k}{1 - c_k e_k^2}\right)^2\right)
\]

where \( B(x, y) \) is the classic beta function.

Turning to the calculation of \( \frac{\partial E(A(\mathbf{\Omega}))}{\partial P_k(\mathbf{\Omega})} \) in (20), we now fix \( k \) and consider the reduced basket obtained by removing the \( k \)th name from the original basket. Letting \( P^{(k)}(\mathbf{\Omega}) \) be the conditional loss distribution for the reduced basket, we have:

\[
P(t; \mathbf{\Omega}) = P^{(k)}(\mathbf{\Omega})P(t - w_k | \mathbf{\Omega}) + P^{(k)}(t - w_k | \mathbf{\Omega}) P_k(\mathbf{\Omega})
\]

from this equation it is trivial to directly determine the distribution \( P^{(k)}(t) \) and, in turn, we can use this to calculate the partial derivative:

\[
\frac{\partial P(t; \mathbf{\Omega})}{\partial P_k(\mathbf{\Omega})} = P^{(k)}(t; \mathbf{\Omega})P(t - w_k | \mathbf{\Omega})
\]

**B. CPU times for quasi-analytical STCDO pricing**

<table>
<thead>
<tr>
<th></th>
<th>N = 50</th>
<th>N = 100</th>
<th>N = 200</th>
<th>Value only</th>
<th>N = 50</th>
<th>N = 100</th>
<th>N = 200</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gaussian</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M = 1 )</td>
<td>0.02</td>
<td>0.04</td>
<td>0.17</td>
<td>0.30</td>
<td>0.64</td>
<td>1.63</td>
<td></td>
</tr>
<tr>
<td>( M = 2 )</td>
<td>0.10</td>
<td>0.25</td>
<td>0.92</td>
<td>0.42</td>
<td>1.06</td>
<td>3.81</td>
<td></td>
</tr>
<tr>
<td><strong>Student-t</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M = 1 )</td>
<td>0.08</td>
<td>0.24</td>
<td>0.94</td>
<td>0.42</td>
<td>1.18</td>
<td>4.14</td>
<td></td>
</tr>
<tr>
<td>( M = 2 )</td>
<td>0.36</td>
<td>1.16</td>
<td>4.67</td>
<td>1.01</td>
<td>3.74</td>
<td>16.12</td>
<td></td>
</tr>
</tbody>
</table>

Notes: CPU times for computations on actual STCDO deals. All times are in seconds on a 1GHz Pentium IV processor. No simplifying assumptions were made for credit spreads, recovery rates, correlations or deal structure. The underlying deal maturity is five years, on a quarterly schedule. Computed sensitivities in the last three columns are parallel shift hazard rate sensitivities, default positions and recovery rate sensitivities for all credits.

\[
\frac{\partial P(t; \mathbf{\Omega})}{\partial P_k(\mathbf{\Omega})} = P^{(k)}(t - w_k | \mathbf{\Omega}) - P^{(k)}(t | \mathbf{\Omega})
\]

With (23) and (21) we can calculate (20), after which the usual integration over the risk factors \( \Omega \) will yield the desired Gateaux derivative \( \frac{dE(A(t))}{d\mu} \).

The techniques outlined above can easily be applied to gauge recovery sensitivities and the effects of sudden default of individual names (the ‘default positions’). For instance, to calculate default positions, we simply remove company \( k \) from the loss pool as in (22), set \( p_k(\mathbf{\Omega}) = 1 \), and reconstitute the full loss distribution portfolio. This ‘trick’ of removing a single name (or several names) from the loss distribution is very useful and can be applied to more general scenario analysis. For instance, it is straightforward to find ‘gamma’ and ‘cross-gamma’ effects by calculating the effect of simultaneous finite-size shocks to hazard rates.

We should point out that the ease and speed with which various sensitivities can be calculated and derived in the recursion-based framework is a significant practical advantage of this method over competing algorithms. To illustrate the computational speed of our quasi-analytical approximation, table B shows CPU times for some computations on actual STCDO deals.

**Monte Carlo simulations**

We now move on to more complicated basket derivatives where the portfolio loss framework cannot be used. This is the case, say, for the ‘nested’ CDOs mentioned in footnote 1 and for CDOs with collection accounts and other complications. For this, we wish to use Monte Carlo to evaluate the expectation (5) and, more importantly, the derivative (11). In this setting, we want to investigate whether two key techniques introduced in the previous section are useful: factor decomposition of the correlation matrix and forming the derivative of the loss density in the delta calculation.

Assuming that the correlation matrix of the Student-\( t \) copula has an \( M \)-dimensional factor structure, a basic Monte Carlo algorithm takes the form:

\( \Box \)

i) draw an \( M + N \)-dimensional sample \( \mathbf{U} \) of uncorrelated standard Gaussian numbers. Let the first \( M \) numbers represent \( \mathbf{E} \) in equation (12), let the remaining numbers be denoted \( \mathbf{E}^c \).

\( \Box \)

ii) generate the correlated sample \( \mathbf{Y} \) as \( \mathbf{Y} = \mathbf{AU} \), \( \mathbf{U} = [\mathbf{X}' \mathbf{E}']' \), where \( \mathbf{A} = [\mathbf{E} \mathbf{E}^c] \) is an \( N \times (N + M) \) matrix. (See (14) for the definition of the diagonal matrix \( \mathbf{F} \)).

\( \Box \)

iii) draw a single chi-square sample \( g \) with \( v \) degrees of freedom to generate a correlated Student-\( t \) sample vector \( \mathbf{Z} \) from equation (4).

\( \Box \)

iv) calculate the random default time vector \( \tau \) by \( \tau = \tau^* \), \( \tau^* = p_k^* (t, \xi(Z), k = 1, \ldots, N) \).

\( \Box \)

v) evaluate the payout function \( f_k \), including discounting.

Steps i-v would be repeated a total of \( n \) times and the average of \( f_k \) would be used as the price estimate. The convergence of the algorithm is \( O(n^{-1/2}) \).

For efficiency reasons, the map between \( \mathbf{Z} \) and \( \xi \) obviously should be pre-tabulated on discrete grids outside the Monte Carlo loop.
As the matrix $F$ is diagonal, the workload of steps i–ii is $O(NM)$ down from the $O(N^2)$ operations needed for general correlation matrices where step ii would involve a multiplication with a non-diagonal Cholesky matrix. For $N$ of the magnitude of 100–200 or more, this reduction can obviously be significant, although the overall effort of the Monte Carlo algorithm is often dominated by the, typically complex, payout evaluation in step v.

Consider now calculating the Gateaux derivative (11). In a brute-force approach, this derivative is calculated by finite differencing:

$$\frac{\partial \delta \hat{X}^V}{\partial \delta} = \frac{V(0, \lambda + \Delta \xi_j) - V(0, \lambda)}{\Delta}$$

where $\Delta$ is some finite scalar. In the Monte Carlo algorithm, we would need to insert the following two steps$^6$ for each index $k$ for which we want to evaluate the derivative:

1. vi) perturb the kth default time $t_k$ to $t_k^* = \bar{t}_k^* (t_k, \xi)$, where $\bar{t}_k^* (t_k, \xi) = 1 - \exp (-\frac{1}{2} \kappa_k (u + \Delta 2^k (u) du)$.

2. vii) compute $\delta \hat{X}_k^V = \delta \hat{X}_k^V (t_1^*, \ldots, t_k^* = t_k, \ldots, t_N^*)/\Delta$.

The average of $\delta \hat{X}_k^V$ over the $m$ Monte Carlo paths would form the estimate of the Gateaux derivative.

The scheme above is simple, but sometimes problematic. First, the determination of $\Delta$ can be tricky: if picked too low in absolute magnitude, statistical noise of the derivative estimator will overwhelm the result; if picked too high, convexity effects (‘gamma’) can make the derivative estimate biased and unreliable. Second, the fact that the algorithm requires a re-evaluation of the payout routine in step vii leads to a potentially enormous amount of payout function calls: to calculate different Gateaux sensitivities for all $N$ companies in the pool would involve $O(N \times m \times n)$ payout evaluations. For complicated payout functions – which are the primary candidates for Monte Carlo simulation – the resulting computing effort would often be very significant.

Inspired by (20) and Boyle, Broadie & Glasserman (1997), we now introduce a direct technique to calculate the derivative that avoids some of the problems of the finite difference approach. Combining (7) and (11), we get:

$$\frac{\partial \delta \hat{X}^V}{\partial \delta} = \int_{\mathbb{R}^2} f^V (t) \frac{\partial}{\partial \lambda} \ln \Phi (t \mid \lambda + \mu \Delta \xi_j) \bigg|_{\lambda = 0} \frac{d \mathbb{P}}{d \mathbb{G}} d t =$$

$$\int_{\mathbb{R}^2} f^V (t) \frac{\partial}{\partial \lambda} \ln \Phi (t \mid \lambda + \mu \Delta \xi_j) \bigg|_{\lambda = 0} \frac{d \mathbb{P}}{d \mathbb{G}} d t$$

which is, that we can use the required derivative by, in effect, substituting for a new payout function $f^V (t) = f^V (t) \frac{\partial}{\partial \lambda} \ln \Phi (t \mid \lambda + \mu \Delta \xi_j) \bigg|_{\lambda = 0} \frac{d \mathbb{P}}{d \mathbb{G}} d t$.

The required stochastic weights $h_{k, j}$ can be calculated explicitly from (3):

$$h_{k, j} (t) = \frac{\xi_j (t)}{h_k (t)} = \frac{\int_0^1 \xi (du)}{h_k (t)} + \frac{(v + 1)^{\frac{1}{2}} Z_k}{1 + v^{\frac{1}{2}} Z_k^2} \alpha (t, Z_k) - \frac{(1 + N/n) \alpha (t, Z_k)}{1 + v^{\frac{1}{2}} Z_k^2} e^{\frac{1}{2} Z_k^2}$$

(24)

where, as before, $Z_k = r_k^{\frac{1}{2}} \lambda (Z_k)$, and:

$$\alpha (t, Z_k) = \frac{1 - P_k (t)}{h_k (Z_k)} = \frac{1 - P_k (t)}{h_k (Z_k)}$$

We note that the Gateaux shifts we are often most interested in are of the form $\xi_j (t) = \int_0^1 \xi (du)$ in which case the expressions above simplify considerably.

In terms of changes to the simulation algorithm, incorporating the derivative calculation above is simply a matter of inserting an additional calculation of the necessary derivative weights $h_{k, j}$ after step v in our basic algorithm. The computation of $Z_k^2 - Z_k^2$ in (24) can be reduced to $O(N)$ operations (as $Z_k^2 - Z_k^2 = Z_k X_k^2$, whereby the computational bottleneck of the delta-computations becomes the expression $e^{\frac{1}{2} Z_k^2} = e^{\frac{1}{2} Z_k^2} Z_k^2$, which is of order $O(N)$ per company $k$. That is, to calculate $h_{k, j}$ for all $k = 1, \ldots, N$ will involve $O(N^2)$ operations per path. We stress that the payout function needs only to be evaluated once per Monte Carlo loop, and the weights $h_{k, j}$ are independent of the payout function. The latter point allows us to efficiently reuse the weights when evaluating multiple payout functions at once (as is the case, say, in the simultaneous pricing of different tranches of a CDO).

The formula (24) holds whether or not the correlation matrix has a factor structure or not. As the $O(N^2)$ operations can become quite significant for large $N$, we now wish to take advantage of the assumed correlation factor-structure to reduce the cost of the algorithm. As before, set $\Omega = (X, g)$. Conditioned on this vector, $Z$ is an $N$-dimensional vector of independent Gaussian samples with mean $m = \sqrt{g} \mathbf{c}$ and standard deviation $s = (\sigma_1, \ldots, \sigma_N), \sigma_j = \sqrt{g} \times F_j$. We can then define a conditional Gateaux derivative as:

$$\frac{\partial \delta \hat{X}^V}{\partial \delta} = \int_{\mathbb{R}^2} f^V (t) \frac{\partial}{\partial \lambda} \ln \Phi (t \mid \lambda + \mu \Delta \xi_j) \bigg|_{\lambda = 0} \frac{d \mathbb{P}}{d \mathbb{G}} d t =$$

$$\int_{\mathbb{R}^2} f^V (t) \frac{\partial}{\partial \lambda} \ln \Phi (t \mid \lambda + \mu \Delta \xi_j) \bigg|_{\lambda = 0} \frac{d \mathbb{P}}{d \mathbb{G}} d t$$

where the Gaussian density $\Phi (t \mid \Omega)$ and the weights $h_{k, j}^{\Omega}$ (T) are easily shown to be:

$$\Phi (t \mid \Omega) = \frac{1}{\sqrt{2 \pi} \sqrt{\det \Omega}} \exp \left[ -\frac{1}{2} (t - m)^T \Omega^{-1} (t - m) \right]$$

$$h_{k, j}^{\Omega} (T) = \frac{\xi_j (T)}{h_k (T)} = \frac{\int_0^1 \xi (u) du}{h_k (T)} + \frac{(v + 1)^{\frac{1}{2}} Z_k}{1 + v^{\frac{1}{2}} Z_k^2} \alpha (t, Z_k) - \frac{(1 + N/n) \alpha (t, Z_k)}{1 + v^{\frac{1}{2}} Z_k^2} e^{\frac{1}{2} Z_k^2}$$

(25)

where $Z_k = r_k^{\frac{1}{2}} \lambda (Z_k)$ and where we assume that $\sigma_j > 0$ for all $k$.

Importantly, by the law of iterated conditional expectations, simply applying our basic Monte Carlo algorithm to the modified payout $f^V (t) = f^V (t) \frac{\partial}{\partial \lambda} \ln \Phi (t \mid \lambda + \mu \Delta \xi_j) \bigg|_{\lambda = 0} \frac{d \mathbb{P}}{d \mathbb{G}}$ will give an unbiased estimate for the derivative $\frac{\partial \delta \hat{X}^V}{\partial \delta}$. For given $m = \sqrt{g} \mathbf{c}X$ (which is calculated in step ii of the basic Monte Carlo algorithm) we note that (25) requires only $O(N)$ operations for the joint estimation of $\frac{\partial \delta \hat{X}^V}{\partial \delta}$ for all $k$. This is, of course, a consequence of the conditional independence of $Z$, given $\Omega$, which removes the term $\sum_k Z_k^2$ from (24). The reduction from $O(N^2)$ to $O(N)$ can, for large baskets, easily save a factor of two to three, with very minor changes to the basic algorithm.$^7$

While not an application that would normally justify Monte Carlo simulations, for reference we now list some simulation results for (parallel shift) hazard rate sensitivity for a regular six-tranche synthetic CDO. Let $\nu_{PF}$ and $\nu_{FD}$ be the sample variance and total computation time associated with the finite difference method, and let $\nu_{PF}$ and $\nu_{FD}$ be defined analogously for the conditional payout weighting method in (25). The efficiency of the latter method relative to the former is then here defined as the ratio:

$$ER = \nu_{PF} / \nu_{FD}$$

In the example shown in table C, the payout weighting method is many orders of magnitude faster than the finite difference method, even for the relatively straightforward payout function used. For small to medium CDOs, the sample variances of the former method are also considerably smaller than those of the latter, but this picture eventually reverses for large CDOs (for $N = 150$, the two methods produce about the same sample variances). Note that the finite difference numbers in the table are based on 1-basis-point shifts. Increasing the shift size will, as discussed earlier, increase the relative efficiency of this method, at the expense of a convexity bias in the resulting numbers. For the example above, we find empirically that an increase in the shift size to 5bp, say, improves the net efficiency of the finite difference results method by about a factor of three to four. On the other hand, increases in deal maturity, CDS spreads and deal complexity will tend to reduce the relative efficiency of the finite difference method, sometimes dramatically so. In practice, some experimentation is generally needed to $^6$ Notice that if $\Delta \times \Delta$ is chosen to be negative, we know that $h_{k, j}$ will be zero for paths where $t_k$ lies beyond the maturity of the basket derivative. For these paths, steps (vi)–(vii) should be skipped for performance reasons.

$^7$ For the data in table C, the factor-structure efficiency savings vary from 10% ($N = 25$) to 20% ($N = 150$).

For instance, the finite difference technique could be applied to calculate low-spread deltas, and the payout weighting technique to medium- and high-spread deltas. See footnote 6 for the rationale behind this (the lower the spread, the less likely it is that a default will take place before the deal maturity).
C. Timing and efficiency of hazard rate sensitivities of six-tranche CDO

A. Timing results (seconds). T: price only

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>TPV</th>
<th>T_p</th>
<th>T_P</th>
<th>TPV/T_P</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>83</td>
<td>84</td>
<td>448</td>
<td>448</td>
<td>18.7%</td>
</tr>
<tr>
<td>50</td>
<td>130</td>
<td>132</td>
<td>1,169</td>
<td>1,169</td>
<td>11.3%</td>
</tr>
<tr>
<td>100</td>
<td>219</td>
<td>226</td>
<td>3,589</td>
<td>3,589</td>
<td>6.3%</td>
</tr>
<tr>
<td>150</td>
<td>320</td>
<td>331</td>
<td>10,040</td>
<td>10,040</td>
<td>3.3%</td>
</tr>
</tbody>
</table>

B. Efficiency ratios for hazard rate sensitivities for tranches A–F

<table>
<thead>
<tr>
<th>N</th>
<th>ER_A</th>
<th>ER_B</th>
<th>ER_C</th>
<th>ER_D</th>
<th>ER_E</th>
<th>ER_F</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>124.1</td>
<td>84.6</td>
<td>79.1</td>
<td>145.5</td>
<td>111.0</td>
<td>253.1</td>
</tr>
<tr>
<td>50</td>
<td>41.8</td>
<td>44.4</td>
<td>45.4</td>
<td>70.8</td>
<td>55.6</td>
<td>117.0</td>
</tr>
<tr>
<td>100</td>
<td>13.3</td>
<td>30.1</td>
<td>23.1</td>
<td>38.5</td>
<td>32.4</td>
<td>64.1</td>
</tr>
<tr>
<td>150</td>
<td>15.6</td>
<td>19.3</td>
<td>20.6</td>
<td>31.2</td>
<td>24.9</td>
<td>52.0</td>
</tr>
</tbody>
</table>

Notes: all numbers based on m = 4 × 10^8 Student-t Monte Carlo simulations on a 1GHz PC. Homogeneous pool with all CDS spreads equal to 175bp and all copula correlations 25%, allowing for a one-factor correlation structure (M = 1). The deal maturity is five years (quarterly scheduled). Irrespective of the pool size N, the tranche definitions were as follows. A: [0%, 2%]; B: [2%, 5%]; C: [5%, 10%]; D: [10%, 15%]; E: [15%, 25%]; F: [25%, 100%]. For the finite difference sensitivities, a shift of 1bp was used along with the speed-up trick in footnote 6.

Appendix. Choosing the loss unit

If all losses in default for individual names are integer multiples of the loss unit, there will be no approximation error arising from the discretisation of the loss distribution. We shall use the term theoretical loss unit for the largest value with this property. Since the computational effort is roughly inversely proportional to the size of the loss unit, the theoretical loss unit leads to the least possible computational effort without discretisation error. In some cases of practical interest, it is indeed possible to use the theoretical loss unit, but in general some approximation error must be accepted in order to reduce the computational effort.

For a given loss unit u the error for a loss W_k is given by |W_k – w_k]|u rounded to the nearest integer. This error (or rather its maximum over the basket) translates roughly into an error in the loss distribution and hence into an error in the apportioning of losses between tranches. Consequently the size of this error should be set at the level of pricing error tolerance for individual tranches. Note that sensitivities (to hazard rates, defaults, etc) will be much less affected by discretisation error.

Suppose that this tolerance is given and denote it ε. If W is the smallest loss value (> ε), we define u := W/n, n = 1, 2,..., W > ε/n. We further define w_k(u) as the integer rounded ratio W_k/|u|. Let be the smallest integer such that |W_k – w_k(u)| < ε for all k. Then we take as our loss unit:

\[ u = \sum_{k=1}^{\infty} w_k(u) \]

Note that the final step attempts to preserve the sum of all losses. Also note that this algorithm will produce the theoretical loss unit whenever this is greater than ε.

Finally, let us note that while the ability to efficiently compute various risk measures is a prerequisite for good hedging, there is much left to be said about the application of the risk measures in actual hedging strategies with transaction costs and discrete-time re-balancing. As the gamma properties of CDOs are often quite complicated, such hedge analysis is an interesting area of future research with a number of open empirical and theoretical issues.

Leif Andersen is a managing director, Jakob Sidenius is a principal and Susanta Basu is a vice-president at Banc of America Securities. They wish to express their gratitude to Jesper Andreae and Vladimir Piterbarg for constructive comments and ideas. E-mail: leif.andersen@bofasecurities.com, jakob.sidenius@bankofamerica.com and susanta.basu@bofasecurities.com

REFERENCES

Boyle P, M Broadie and P Glasserman, 1997
Simulation methods for security pricing
Journal of Economic Dynamics and Control 21, pages 1,267–1,321

Frey R and A McNeil, 2001
Modelling dependent defaults
Working paper, University of Zurich

Glasserman P, P Heidellerger and P Shahabuddin, 1999
Asymptotically optimal importance sampling and stratification for pricing path-dependent options
Mathematical Finance 9, pages 117–152

Gregory J and A Laurent, 2003
I will survive
Risk June, pages 103–107

Masahl R and M Naldi, 2002
Extreme events and default baskets
Risk June, pages 119–122

Merino S and M Nyfeler, 2002
Calculating portfolio loss
Risk August, pages 82–86

Pykhtin M and A Dev, 2003
Coarse-grained CDOs
Risk January, pages 119–116

Schmidt W and I Ward, 2002
Pricing default baskets