A Monte Carlo pricing engine for n^{th} to default credit swaps in the Li model

Giuseppe Milicia
milicia@brics.dk

Abstract. The aim of this work is the implementation of pricing engine for Nth to default credit swaps. The pricing model we consider is the "Li model" [Li, 2000]; our implementation is based on the techniques described in [Joshi and Kainth, 2004]. The model described in [Li, 2000] uses Gaussian Copulae to model asset correlation. It is well-known that Gaussian Copulas do not model satisfactorily tail dependencies which can have a significant impact in the pricing of credit derivative. We consider using a T-copula as to better model tail dependencies and study their impact on the credit swaps prices.

1. Introduction

A *CREDIT DERIVATIVE* is a derivative contract which transfers the risk of certain *credit events* (e.g. company defaults) from the *buyer* to the *seller* of the contract without transferring the underlying assets. Indeed credit derivatives can be thought as insurances against credit events.

Credit derivatives allow trading and hedging of *credit risk*. Although most of the credit market is still *over the counter* (OTC) some of the contracts are rather liquid.

Some of the most popular contracts are *credit default swaps* (CDS), *collateralized debt obligations* (CDO), *credit linked notes*, etc [jpm, 1999, leh, 2003].

In this paper we concentrate on *n-th to default credit default swaps* (N2D).

1.1. Nth to default credit swaps.

A CDS is a contract which provides protection to the buyer over the credit risk associated with an underlying asset. The buyer pays, at regular intervals, a spread to the seller till contract expiration or till a *credit event* occurs. The type of credit events covered are specified in the contract, these could include bankruptcy, failure to meet a payment in a timely fashion and debt restructuring. Typically, the more lax the definition of credit event the more expensive the contract becomes. If a default occurs before the contract expired the seller will pay the buyer the face value of the contract minus a *recovery rate*. We have thus two legs for a CDS: the spread payment (premium leg) and the default payment (protection leg). Typical maturities for CDS are three, five and ten years. This structure is sketched in Figure 1.

CDS owe their popularity to a number of reasons:

An electronic version of the article and the corresponding Matlab code are available online at http://www.brics.dk/~milicia/libraryGM.htm

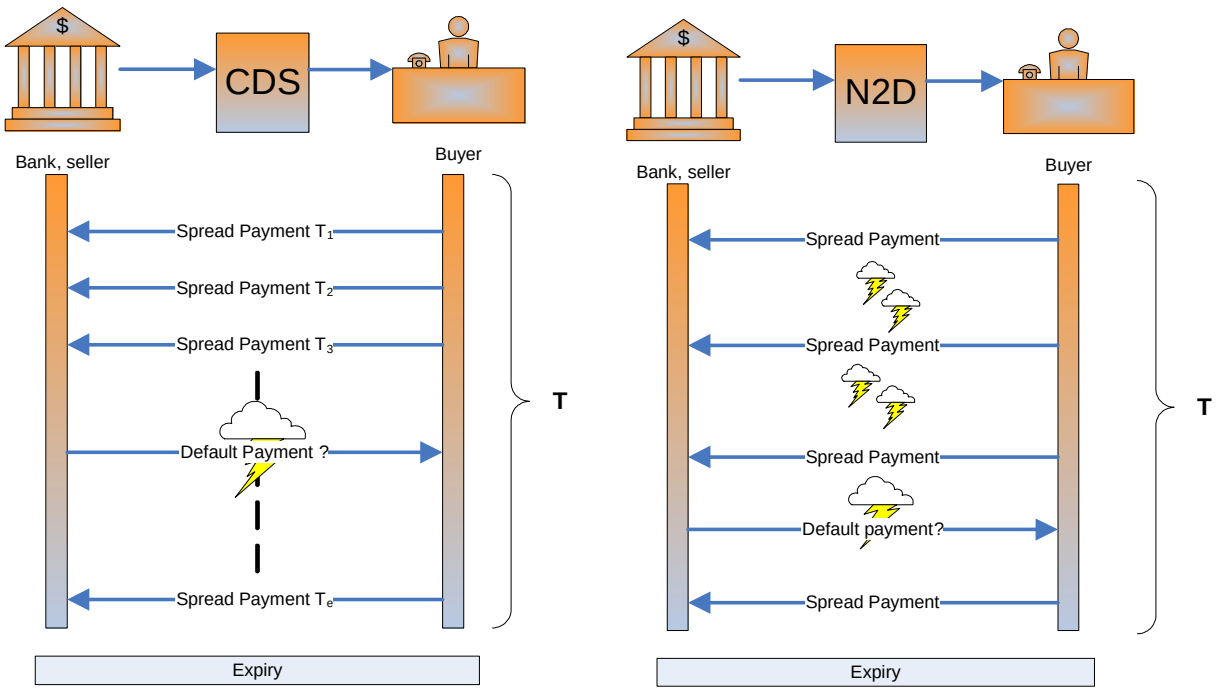


Figure 1. CDS and N2D structures

Customisation: CDS are OTC instruments and hence can be customised to meet specific investors requirements.

Trading: CDS introduce new trading opportunities as they allow investors to bet on their views on credit.

Leverage: As it is the case for most derivatives, CDS allow investors to take highly leveraged positions based on their view of the credit market.

A N2D [**jpm, 1999**, **leh, 2003**] is a generalised multi-asset version of a CDS contract. A N2D provides protection over the credit risk of a basket of obligors. Let T denote the life of the contract. As for a CDS the buyer pays spreads $s_1, \ldots, s_e$ to the seller at times $0 < T_1, \ldots, T_e < T$, however the other side of the deal is different. The seller will pay the buyer only after the n-th default occurs, in that case the seller will pay the face value of the asset which has defaulted minus its recovery rate, note, however, that a number of variations of this basic scheme are in use. This structure is depicted in Figure 1. Popular N2D contracts are *first to default* baskets (F2D), with $n = 1$ and *second to default* baskets (S2D) with $n = 2$.

Before jumping into a quantitative analysis of the N2D contract and its pricing, we can start with a qualitative investigation of their behaviour. Intuitively the value of a N2D contract must depend on the a number of variables including:

- Maturity.
- Basket size.
- Value of n, e.g. F2D, S2D, etc.

- Credit rating of the obligors in the basket
- Recovery rates.
- Correlation of obligors in the basket.

We briefly analyse each of them.

Maturity. It stands to reason that the maturity of the contract will influence its price although it might not be clear in which way. For instance some obligors might become less risky in the longer term (a company can have stormy time when it is starting up but becoming more trustworthy as time goes by) or increasingly risky (a company doing well in the present might face tough market conditions in five years time).

Basket size. We would expect the contract to become more expensive as the size of the basket increases, indeed the defaults must become more likely as we add more obligors to a basket.

Value of n. From the point of view of the protection seller, the higher the n the less likely she is to pay: at least n defaults must happen before protection payments occurs. Hence we would expect an F2D to be more expensive than an S2D and so forth.

Credit rating. The higher the quality of the obligors the less likely are the defaults, hence a basket of high quality credit will be cheaper than a basket of low quality credit.

Recovery rates. Instruments with better recovery rates should results in lower spreads.

Correlation. Correlated assets will tend to default at the same time, whilst for independent assets any default will give us no insight on the possibility of observing other default. Hence N2D are *correlation* instrument: their value depends strongly on the dependence profile of the underlying asset basket. However, getting an intuitive feel of how default correlation affects the value of an N2D is not trivial. Let us consider some simple cases. Consider a F2D for a basket of n *independent* assets, intuitively this contract is very similar to n CDS and we would expect its premium to be equivalent to the sum of those n CDS. If the correlation of the basket increases the F2D will loose value as assets will tend to default similarly. The situation can be very different for higher order swaps: if we have independent assets the chances of observing multiple defaults will be lower than in the case of strongly correlated assets! Hence a F2D and higher order swaps will react to asset correlation in the opposite manner. The difference is remarkable, it appears that investors can be either long or short correlation depending on the order of the swap and on its correlation profile. We will come back to this point once we obtain a quantitative framework for N2D pricing.

We are now ready to formalise our intuitions. Consider a n-th to default CDS with a basket of size N and maturity T; if we do not have n defaults before expiry the buyer will pay e spreads at regular intervals T_i. Let τ_i denote the default time of the i-th instrument, with $\tau_i = \infty$ if the asset never defaults. We then indicate with τ the of the n-th default. Let $B(t)$ be the discount factor at time t. Let $\mathcal{I}(\tau < T)$ be the indicator function of the event $\tau < T$, that is whether the n-th default occurred *before* the contract expiration time. Formally the payoff of the two legs of the contract can be defined as:

$$\widetilde{V}(\tau_1, \tau_2, \ldots, \tau_N) = V_v(\tau_1, \tau_2, \ldots, \tau_N) - V_p(\tau_1, \tau_2, \ldots, \tau_N)$$

Where

$$V_v(\tau_1, \tau_2, \ldots, \tau_N) = (1 - r_n)B(\tau)\mathcal{I}(\tau < T)$$

And

$$V_p(\tau_1, \tau_2, \ldots, \tau_N) = \begin{cases} \sum_{i=1}^{j} B(T_i)s_i + s_{j+1}B(\tau)\frac{\tau - T_j}{T_{j+1} - T_j} & \text{if } T_j \leq \tau \leq T_{j+1} \\ \sum_{i=1}^{e} B(T_i)s_i & \text{if } \tau > T \end{cases}$$

Note the *accrual* term $s_{j+1}B(\tau)\frac{\tau - T_j}{T_{j+1} - T_j}$ which the buyer pays when the n-th default occurs in between spread payment dates. The value of the contract is then $\mathbb{E}(\widetilde{V})$.

As in [**Chen and Glasserman, 2006**] we separate the value of the contract into its deterministic and random component (V):

$$V = \widetilde{V} + \sum_{i=1}^{e} B(T_i)s_i = V_v - s_{j+1}B(\tau)\frac{\tau - T_j}{T_{j+1} - T_j}$$

Where we note that V will be zero unless n defaults occur. Hence calculating the value of the swap boils down to calculating the expectation $\mathbb{E}(V)$. The fair value of a N2D contract is then such that:

$$\mathbb{E}(V) = \sum_{i=1}^{e} B(T_i)s_i$$

2. Pricing in the Li Model

At the basis of credit derivative pricing is the modelling of default times and their correlation profile. In this paper we focus on the Li Model [**Li, 2000**].

In the Li model default times follow an exponential distribution $f(t) = he^{-ht}$ where the parameter h is usually called the *hazard rate*. Typically the hazard rate is assumed to be constant, at least in the short term, or deterministic. Often the term *credit curve* is used to refer to the deterministic hazard rate. Obtaining a credit curve to work with is the first step in pricing credit derivatives, three approaches stand out:

Merton approach: The Merton approach to credit risk modelling [**Merton, 1994**] aims at using fundamental information about a company to build its credit curve.

Historical approach: We estimate the credit curve based on historical default information and credit scores provided by rating agencies such as Moody and Standard and Poors.

Implied approach: The credit curve is constructed using the implied default probability inherent in the market prices of bonds, liquid CDS and other default sensitive instruments. See for instance [**Li, 1998**].

In the Li model, instead of modelling the multivariate distribution of the default times of the assets in our baskets we rather model the individual default times and use a *copula* [**Cherubini et al., 2004**] approach to derive their joint distribution. This approach has a number of advantages, the main ones being that we work with the marginals and are able to inject complex non linear correlation structures into the problem with minimal effort.

Definition 1 (Copula Function)**.** A function C is a copula with the following properties:

- $C : [0,1]^n \longrightarrow [0,1]$
- C has margins C_n such that $C_n(u) = C(1,\ldots,1,u,1,\ldots,1)$
- C is grounded and N-increasing:

$$\sum_{i_1=1}^{2}\cdots\sum_{i_n=1}^{2}(-1)^{i_1+\ldots+i_n}C(u_{1,i_1},\ldots,u_{n,i_n}) \geq 0$$

$$\forall(u_{1,1},\ldots,u_{n,1}) \text{ and } (u_{1,2},\ldots,u_{n,2}) \text{ such that } u_{n,1} \leq u_{n,2}$$

From this definition follows that is $F_1,\ldots,F_n$ are univariate distributions, than $C(F_1,\ldots,F_n)$ is a multivariate distribution whose marginals are indeed $F_1,\ldots,F_n$. Hence we can use copula functions to build joint distributions out of known marginals: we are injecting a dependency profile into the join distribution by our choice of copula function.

Definition 2 (Multivariate distribution using a copula)**.** Given m univariate distributions

$$F_1(x_1), F_2(x_2), \ldots, F_m(x_m)$$

we define the function:

$$F(x_1,x_2,\ldots,x_m) = C\big(F_1(x_1),F_2(x_2),\ldots,F_m(x_m)\big)$$

which is a multivariate distribution with marginals $F_1(x_1), F_2(x_2), \ldots, F_m(x_m)$

The above definition if the converse of the famous Sklar's Theorem which states that given a multivariate distribution $F(x_1,x_2,\ldots,x_m)$ with marginals $F_1(x_1), F_2(x_2), \ldots, F_m(x_m)$ there exist a copula function C such that

$$F(x_1,x_2,\ldots,x_m) = C\big(F_1(x_1),F_2(x_2),\ldots,F_m(x_m)\big)$$

In the Li model we use the *Gaussian* copula:

Definition 3 (Gaussian Copula)**.** Given m univariate distributions $F_1(x_1), F_2(x_2), \ldots, F_m(x_m)$ we define the function:

$$C\big(F_1(x_1),F_2(x_2),\ldots,F_m(x_m)\big) = \Phi_m\big(\Phi^{-1}(F_1(x_1)),\ldots,\Phi^{-1}(F_m(x_m))\big)$$

Where Φ_m is a multivariate gaussian CDF with correlation matrix ρ.

(1) Draw n values from a uniform distribution
(2) Transform the n values in a vector of normal variates Z
(3) Set $W = AZ$
(4) Set $u_i = \Phi(W_i)$
(5) Set $\tau_i = F^{-1}(u_i, h_i)$

Table 1. Gaussian Copula Algorithm

```
if numel(rho) == 1
    if (rho < -1 | 1 < rho)
        error('RHO is not a correlation matrix.');
    end
    u = normcdf(mvnrnd([0 0],[1 rho; rho 1],n));
else
    if ~iscor(rho)
        error('RHO is not a correlation matrix.');
    end
    p = size(rho,1);
    u = normcdf(mvnrnd(zeros(1,p),rho,n));
end
tau = inverse_cdf(u);
```

Table 2. Gaussian Copula Algorithm in Matlab

The Gaussian copula can be simulated using the algorithm in Table 1 [**Cherubini et al., 2004**]. In Table 2 we show a Matlab code snipped which implements the algorithm assuming that `rho` is a correlation matrix and that `inverse_cdf` is the inverse CDF of our target distribution (F^{-1}). The code calculates `n` sets of variates for the given copula hence we would not require a loop to calculate a sequence of random variates. This is a typical Matlab pattern: we work with vectors rather than loops [**Higham, 2002**].

Measuring correlation. Usually we measure correlation using Pearson's linear correlation ρ which in the bivariate case is defined as:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sqrt{\sigma_x \sigma_y}}$$

Linear correlation works well with normal variates but its insights degrade as we move away from normal distributions. This is particularly evident when using copulas. A better approach is to use a *rank correlation* coefficient [**Bouye et al., 2000**], the most used are *Spearman's rho* and *Kendall's tau*. For simplicity we limit our definitions to the bivariate case.

Definition 4 (Spearman's rho)**.** Given two random variables x, y with marginal cumulative distributions Ψ_x, Ψ_y and joint density ψ Spearman's rho is:

$$\varrho = 12 \int \int \Psi_x(x) \Psi_y(y) \psi(x, y) \mathrm{dxdy} - 3$$

Definition 5 (Kendall's tau)**.** Given two random variables x, y with marginal cumulative distributions ψ_x, ψ_y and joint density Ψ Kendall's tau is:

$$\tau = 4 \int \int \Psi(x, y) \psi(x, y) \mathrm{dxdy} - 1$$

```
function [Value, StandardError] = mc(runs, randomHandle, payoffHandle)
%MC A generic Monte Carlo algorithm.

% We feed in a payoff function and a way to obtain appropriate randon variables.
%
RunningSum = 0;
RunningSumOfSquares = 0;

blockSize = 1000;
runBlocks = runs/blockSize;

for i=1:runBlocks
    b = randomHandle(blockSize);
    RunningSum = RunningSum + sum(payoffHandle(b));
    RunningSumOfSquares = RunningSumOfSquares + sum((payoffHandle(b)).^2);
end

Value = RunningSum/runs;
Sigma = sqrt((RunningSumOfSquares/runs) - (RunningSum/runs)^2);
StandardError = Sigma/sqrt(runs);
```

Table 3. Generic Monte Carlo code

2.1. Monte Carlo Pricing. In this section we describe a basic Monte Carlo algorithm for N2D contracts based on our discussion so far.

In Table 3 we can see a generic Monte Carlo routine to calculate the price of the contract and the variance of the Monte Carlo estimator. The routine takes as input the number of simulation runs we wish to perform and two *function handles*: Matlab's way to functions as parameter. The first function handle `randonHandle` points to the function that the Monte Carlo routine will use to simulate default times, whilst the handle `payoffHandle` is the function which calculates the contract's payoff on each path of our simulation.
Note how we split the overall MC path simulation into sub-blocks of size `blockSize`, that makes sure that, no matter how many MC runs we wish to perform, we will not run out of memory. As to be expected loops are kept to a minimun: we work on vectors for each sub-block.

In Table 4 we can see the definition of the payoff function we shall use for the rest of the paper.

At the core of the Monte Carlo pricing algorithm is the simulation of the default times. For a gaussian copula with exponential marginals with hazard rates h (assume this is a vector of the appropriate size, hence hazard rates need not to be homogeneous), correlation matrix `rho` we have:

```
@(blockSize) expinv(copularnd('gaussian',rho,blockSize),repmat(mu, blockSize, 1));
```

Where mu $= \frac{1}{h}$.

2.2. Pricing with the Gaussian copula. In this section we refine the qualitative arguments about N2D prices we presented in Section 1.1 with some quantitative insights brought forward by our pricing model. For the purpose of this paper we shall use the test baskets defined in Appendix A with occasional variations. The values we show are in percentage of the swap face value. Also note that all our simulations are based on 50000 runs.

```
function [v, idxs] = payoff(b, r, recovery, T, defaults)
% Payoff function for the N2D contract
%
    temp = sum(b<=T, 2);    % how many defaults before expiry?
    temp2 = b(temp>=defaults,:); % selects the row with at least 'defaults' defaults
    idxs = find(temp>=defaults);
    [temp3, orderedIDX] = sort(temp2, 2); % sort them

    [i,j] = size(temp3);
    recovery = repmat(recovery, i, 1);
    for k = 1:i
        recovery(k,:) = recovery(k,orderedIDX(k,:));
    end
    recovery = recovery(:, defaults);

    nth_default_time = temp3(:,defaults); % Get the defaults(th) default time

    v = 100 * (1 - recovery) .* discountFactor(r, nth_default_time); %result in pct
end

%
% Discount factor at time t
%
function v = discountFactor(r, t)
    v = exp(-r * t);
end
```

Table 4. Payoff function for a N2D

In Section 1.1 we argued that F2D and higher order N2Ds behave differently with respect to correlation. We have now the tools to quantify the difference. In Figure 2 we show how correlation affects N2D prices. We are pricing the test basket `B0` with yearly interest rates of 5% and duration of 5 years. We can see how the prices changes as we increase the (uniform) correlation of the assets. As we expected, lower order swaps tend to react differently than higher order ones with respect to correlation. Indeed the behaviour of a F2D with respect to correlation is opposite to that of a 5th to default swap. The F2D value decreases as correlation increases whilst for a 5th to default swap the opposite is true: the value of the swap increases together with correlation.

In Section 1.1 we also noted that swap prices will raise as the size the baskets increases: defaults are more likely in bigger baskets. Figure 3 confirms our intuition. The swap we price is the same as above with uniform correlation $\sigma_{ij} = 0.5$ and growing basket size.

2.3. Pricing under alternative copulae. In this section we compare the pricing of N2D under different copula choices. It has been noted that T copulae result in prices which are a better match of observed market prices [**Hull and White, 2004**].

The T copula. So far we have considered only Gaussian copulae. However, it is well known that gaussian copulae fail to model satisfactorily tail dependencies [**Cherubini et al., 2004**, **Bouye et al., 2000**]. In this section we shall introduce an alternative copula which offers a more satisfactorily treatment of tail

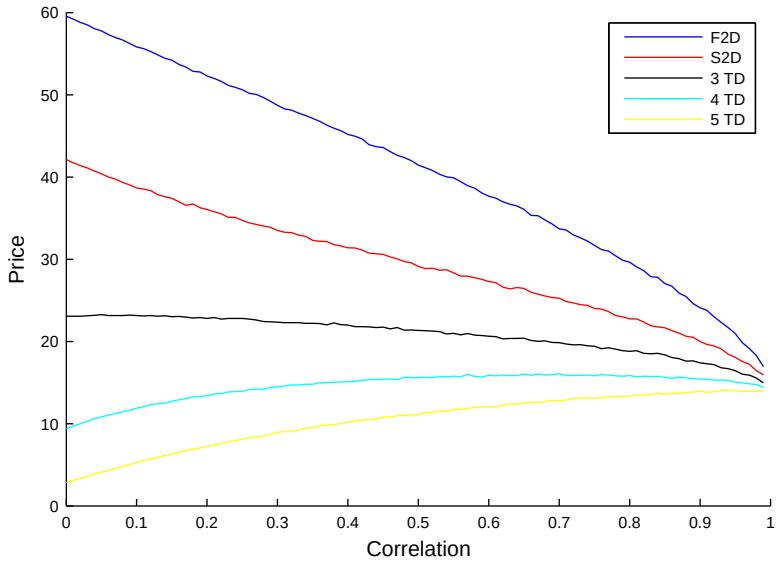
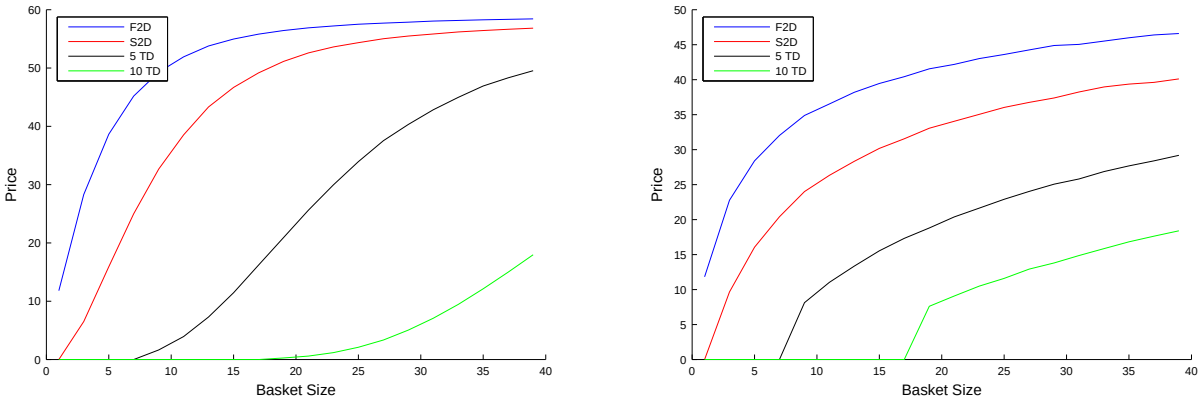


Figure 2. Comparing F2D and higher order swap prices with respect to correlation



(a) Correlation $\sigma_{ij} = 0.01$

(b) Correlation $\sigma_{ij} = 0.5$

Figure 3. Comparing swap prices with respect to basket size

dependencies: the T copula. We then adapt the techniques of the previous sections to the new copulae and given a quantitative evaluation of the price differentials between the various copulae.

Definition 6 (T copula)**.** Let $\mathbf{T}_{\Sigma,\nu}$ be a multivariate T distribution function with ν degrees of freedom and correlation matrix Σ. Let $\mathbf{T}_{\nu}^{-1}$ be the inverse of the univariate T distribution function. We define the T copula as:

$$C(u_1, u_2, \ldots, u_n; \rho, \nu) = \mathbf{T}_{\Sigma,\nu}(\mathbf{T}_{\nu}^{-1}(u_1), \ldots, \mathbf{T}_{\nu}^{-1}(u_n))$$

In Table 5 we can see an algorithm to simulate the T copula [**Cherubini et al., 2004**]. In Table 6 we can see a Matlab code snippet with its corresponding implementation, the code assumes that `rho` is a valid correlation matrix and `nu` is the degrees of freedom of the T distribution.

(1) Draw n values from a uniform distribution
(2) Transform the n values in a vector of normal variates Z
(3) Draw an independent χ^2_ν variate s
(4) Set $W = \sqrt{\frac{\nu}{s}} AZ$
(5) Set $u_i = \mathbf{T}_\nu(W_i)$
(6) Set $\tau_i = F^{-1}(u_i, h_i)$

Table 5. T Copula Algorithm

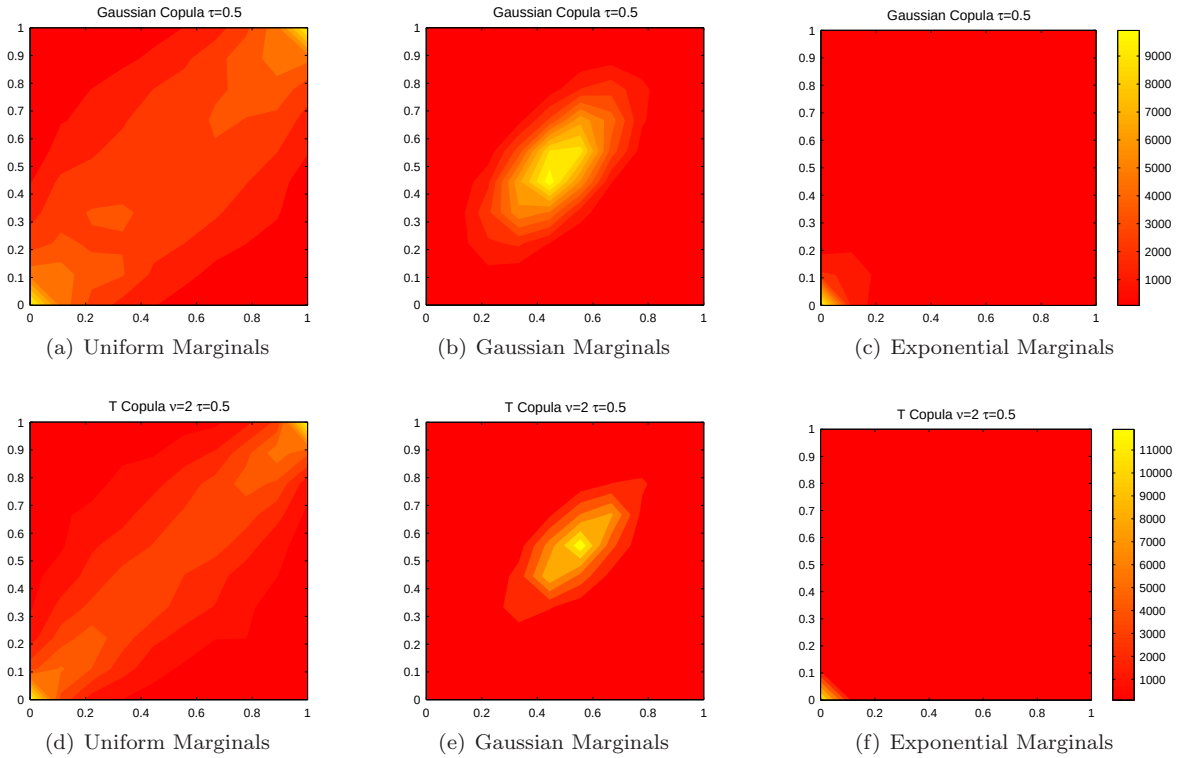
```
if ~(0 < nu)
    error('NU must be positive for the t copula.');
end
if numel(rho) == 1
    if (rho < -1 | 1 < rho)
        error('RHO is not a correlation matrix.');
    end
    u = tcdf(mvtrnd([1 rho; rho 1],nu,n),nu);
else
    if ~iscor(rho)
        error('RHO is not a correlation matrix.');
    end
    u = tcdf(mvtrnd(rho,nu,n),nu);
end
tau = inverse_cdf(u);
```

Table 6. T Copula Algorithm in Matlab

Using copulae it becomes easy to construct, given the marginals, a multivariate distribution with a certain correlation profile. The density plots Figure 4 give us an intuitive feeling of the properties of the two copulae. All the copulae have rank correlation $\tau = 0.5$. Figure 4(a) and 4(d) apply, respectively, a Gaussian and a T copula to uniform marginals. Similarly Figure 4(b) and 4(e) are based on gaussian ($N(0,1)$) marginals, whilst Figure 4(c) and 4(f) have exponential marginals (with intensity 1). This latter deserves our attention as in the Li model default times are assumed to be exponentially distributed; in Figure 5 we zoom in on the results of applying a gaussian copula to pair of exponential marginals.

The T copula has received remarkable attention in the credit derivatives literature and empirical evidence seems to indicate that it results in a good match between model prices and market prices [**Hull and White, 2004**].

In Figure 6 we can see how Gaussian and T copula differs when pricing our test baskets. We use the test basket `B0` (Figure 6(a)) and `B2` (Figure 6(b)) both with yearly interest rates of 5% and a duration of 5 years. The Figure shows how the two copula price different swaps as the (uniform) correlation varies; the solid lines are prices under a gaussian copula, whilst the dashed lines are obtained using a T copula with 2 degrees of freedom. As we can see, with respect to the gaussian copula, pricing with the T copula has the tendency to yield lower prices when price is inversely proportional to the correlation and higher prices when the opposite occurs.



(a) Uniform Marginals (b) Gaussian Marginals (c) Exponential Marginals

(d) Uniform Marginals (e) Gaussian Marginals (f) Exponential Marginals

Figure 4. Copula Densities with Various Marginals

As we know, the T distribution becomes closer to a gaussian as we increase its degrees of freedom, hence prices under a T copula will slowly match the prices obtained under a gaussian copula as we increase the degrees of freedom.

3. Efficiency concerns

The problem with a straightforward Monte Carlo approach to our pricing problem is that in most simulated paths we will see no defaults, hence our contract will have a constant value! Indeed the occurrence of a credit event should be the exception rather than the norm.

Example. Consider a basket of n independent obligors whose survival time follows an exponential distribution with constant hazard rate h. Then the density of each survival time is he^{-ht} and the probability of all of them defaulting by time T is given by:

$$\left(\int_0^T he^{-ht}\mathrm{dt}\right)^n = \left(1-e^{-hT}\right)^n \approx (hT)^n$$

which, in general, might be very small, even for small baskets. Hence only few paths of our simulation would include the event we are interested in.

The situation becomes a lot better if we are interested in a smaller number of defaults. For a first to default swap, we would need only 1 default per simulation path, hence the probability of having a relevant

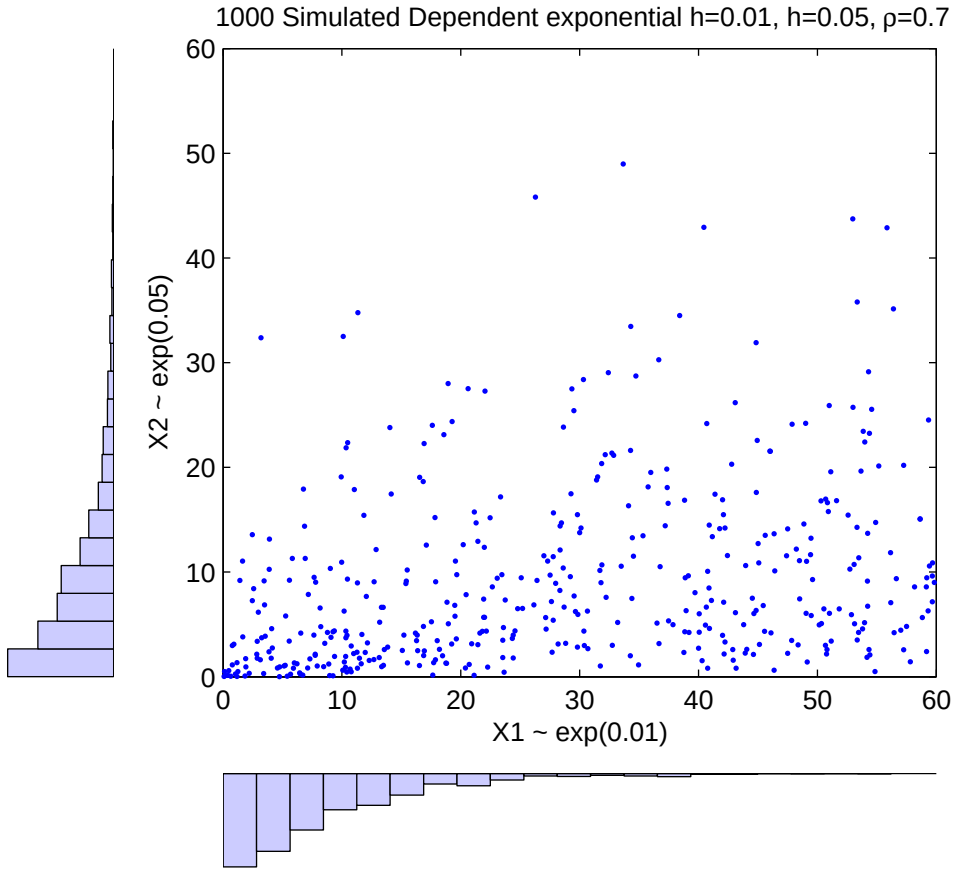
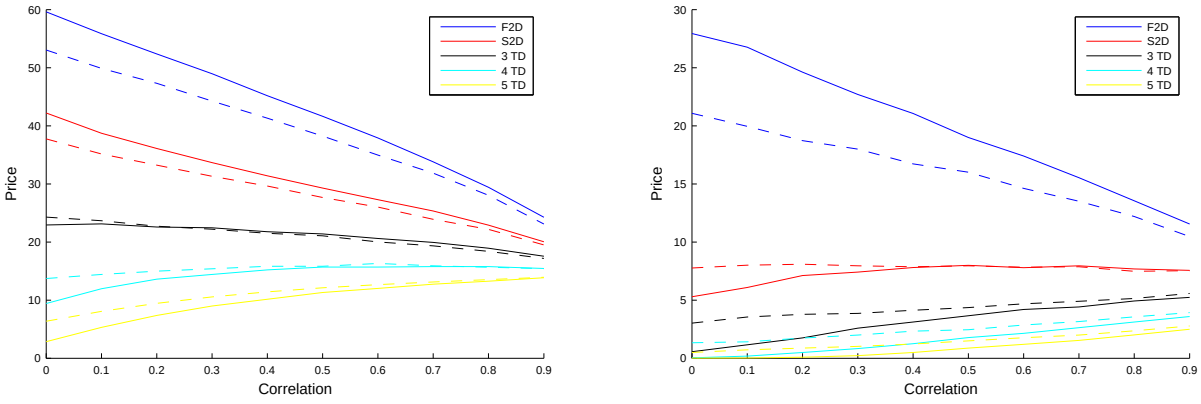


Figure 5. Gaussian Copula with exponential marginals



(a) Basket B0

(b) Basket B2

Figure 6. Comparing prices under Gaussian and T copulae

path becomes:

$$\int_0^T nhe^{-nht}dt = 1 - e^{-nhT} \approx nhT$$

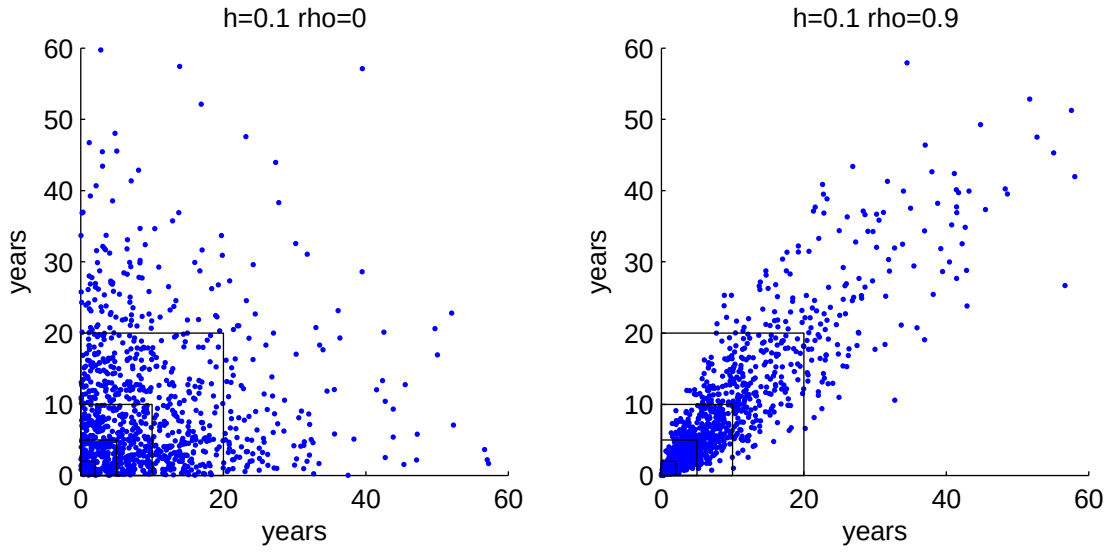


Figure 7. Highlighting the importance region

as it is easy to see that the hazard rate of the minimum default time is $h_{min} = h_1 + h_2 + \ldots + h_n = nh$. However, even in this case for small T we might encounter problems.

In Figure 7 we highlight the *important region* in the case of defaults distributed with mean $h = 0.1$. Only points in that region are relevant to our simulation. The longer lived the contracts the bigger the important region, however it is clear that a great deal of the simulated default times will not be relevant for most contracts. Indeed since outside the important region our contract will have constant value we are just wasting computation cycles whenever we draw survival times outside that area!

3.1. Importance Sampling. *Importance Sampling* [Glasserman, 2003] (IS) is a *variance reduction* technique which ensures that most of our random draws are relevant to the problem at hand: they fall into the important region. This technique is used in the field of *rare event* simulation; in the finance arena we see it applied, for instance, in pricing deep out of the money options and any other contract which has zero or constant value in a big subspace of our random distribution.

In technical terms, IS is a change of probability measure which takes us from the original probability measure to a new, biased, one which better suits our purposes. Assume we want to calculate $\mathbb{E}(f(x))$ where $f(x)$ has density $\psi(x)$:

$$\mathbb{E}(f(x)) = \int_{x \in A} f(x)\psi(x)\mathrm{dx}$$

Let $\varphi(x)$ be a biased distribution which focuses our random variates in the important region, we can the write:

$$\mathbb{E}\left(f(x)\right) = \int_{x \in A} f(x)\psi(x)\mathrm{dx} = \int_{x \in A} f(x)\varphi(x)\frac{\psi(x)}{\varphi(x)}\mathrm{dx} = \mathbb{E}^*\left(f(x)L\right)$$

where $L = \frac{\psi(x)}{\varphi(x)}$ is the *Radon-Nikodym derivative* or likelihood ratio of the change of measure we applied. The IS estimator is then:

$$m_{is} = \frac{1}{K}\sum_{i=1}^{K} f(x_i)L(x_i)$$

with $x_i \sim \varphi(x)$

Clearly the choice of the biased probability distribution $\varphi(x)$ is specific to the problem at hand, hence we cannot have a generic important sampling scheme.

Since IS is a variance reduction technique, often the variance ratio $\frac{\sigma^2_{mc}}{\sigma^2_{is}}$ is used as a measure of the effectiveness of the IS scheme used. It must be noted, however, that this metric does not take into account the computational overhead which if often introduced by an IS scheme: actual computation time tends to increase.

3.2. Efficient Pricing. To address the inefficiencies of a straightforward Monte Carlo approach to pricing, we shall use the IS techniques described in [**Joshi and Kainth, 2004**, **Chen and Glasserman, 2006**]. Our exposition follows the style of [**Chen and Glasserman, 2006**].

Given a N2D on the n-th default of a basket of size N, we want to tweak our default probabilities as to make sure that each simulated path contains at least n defaults. Let $Y_i = \mathcal{I}(\tau_i \leq T)$ be the indicator variables for defaults which occur within the lifetime of the contract. The IS probabilities we shall use are defined as:

$$q_1 = \frac{n}{N}, \qquad q_i = \begin{cases} \frac{n-\sum_{j=1}^{i-1} Y_j}{N-i+1} & \sum_{j=1}^{i-1} Y_j < n \\ p_i & \sum_{j=1}^{i-1} Y_j \geq n \end{cases}$$

Where we indicate with $\sum_{j=1}^{i-1} Y_j$ the number of assets which have defaulted by time T. Hence we tweak the probabilities only if we do not have enough defaults, once a path has enough defaults we go back to the original probabilities. It is easy to see that the probabilities above will always results in n defaults per path. Let $p_i = F_i(T|\tau_1, \ldots, \tau_{i-1}) = \mathbf{P}\left(Y_i = 1|\tau_i, \ldots, \tau_{i-1}\right)$ The JK IS algorithm can then be expressed as follows.

JK IS Sampling [**Joshi and Kainth, 2004**, **Chen and Glasserman, 2006**]

(1) Draw a vector U of N independent uniform variate over $(0, 1)$

(2) Let[1]:

$$V_i = \begin{cases} \frac{p_i U_i}{q_i} & \text{if } U_i \leq q_i \\ p_i + \frac{1-p_i}{1-q_i}(U_i - q_i) & \text{otherwise} \end{cases}$$

and

$$\tau_i = F_i^{-1}(V_i|\tau_1, \ldots \tau_{i-1})$$

[1]In [**Chen and Glasserman, 2006**] (March 2006 version) the definition is

$$V_i = \begin{cases} \frac{p_i U_i}{q_i} & \text{if } U_i \leq q_i \\ p_i + \frac{1-p_i}{1-q_i} U_i & \text{otherwise} \end{cases}$$

this appears to be a typo as it can be verified directly and by comparison with the original algorithm in [**Joshi and Kainth, 2004**]

(3) Calculate the likelihood ratio:

$$L_i = \begin{cases} \frac{p_i}{q_i} & \text{if } Y_i = 1 \\ \frac{1-p_i}{1-q_i} & \text{if } Y_i = 0 \end{cases}$$

Once we have a sample path $\tau_1, \ldots, \tau_N$ we calculate $V(\tau_1, \ldots, \tau_N)$ and adjust it multiplying by $L = L_1 \ldots L_N$. To calculate $\tau_i = F_i^{-1}(V_i|\tau_1, \ldots \tau_{i-1})$ we can use the following Lemma [**Chen and Glasserman, 2006**]:

Lemma 3.1. Let $\Sigma = AA'$ be the positive definite correlation matrix. Let $\mathbf{W} = \mathbf{AZ}$, and $\tau_i = F_i^{-1}(\Phi(W_i))$ then:

$$\mathbf{P}(\tau_i \leq t|Z_1, \ldots, Z_{i-1}) = \Phi\left(\frac{\Phi^{-1}(F_i(t)) - \sum_{j=1}^{i-1} A_{ij}Z_j}{A_{ii}}\right)$$

This Lemma gives us also a way to calculate the p_i by setting $t = T$:

$$p_i = \Phi\left(\frac{\Phi^{-1}(F_i(T)) - \sum_{j=1}^{i-1} A_{ij}Z_j}{C_{ii}}\right)$$

In Table 7 we have a Matlab implementation of the algorithm. Our implementation is partially *vectorised* as it is often the case for Matlab code [**Higham, 2002**].

In Figure 8 we can see a comparison of the variance between the JK method and the plain Monte Carlo one. The figure shows results similar to those presented in [**Chen and Glasserman, 2006**] and highlights that the JK method can perform worse than a plain Monte Carlo simulation. Our intuition would say that the JK method would perform better when defaults are unlikely and gradually become worse as defaults become more likely. Indeed when defaults become likely, a plain Monte Carlo method performs better than the JK method.

In [**Chen and Glasserman, 2006**] the authors note that a simple modification of the JK method yields a IS algorithm which performs at least as well as a plain MC algorithm. It suffices to change our tweaked probabilities to become:

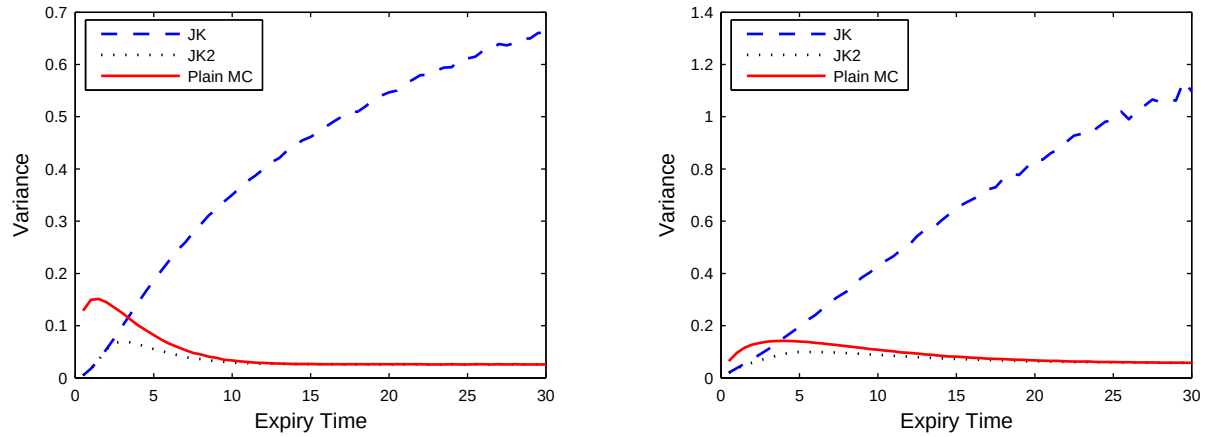
$$q_1 = \frac{n}{N}, \qquad q_i = \begin{cases} \min\left(\frac{n-\sum_{j=1}^{i-1} Y_j}{N-i+1}, p_i\right) & \sum_{j=1}^{i-1} Y_j < n \\ p_i & \sum_{j=1}^{i-1} Y_j \geq n \end{cases}$$

The variance of the improved JK algorithm (JK2) can be seen in Figure 8, as we can see the JK2 algorithm performs better than the original JK one, however it is still possible for the algorithm to underperform when compared with a simple MC scheme.

3.3. Importance Sampling with the T-copula. The original papers [**Joshi and Kainth, 2004**, **Chen and Glasserman, 2006**] are focused on the gaussian copula. In [**Joshi and Kainth, 2004**] we find a brief mention of the possibility of adapting the techniques of the previous section to other elliptical copulae. In this section we expand on that and attempt to give a full fledged IS algorithm under the T copula.

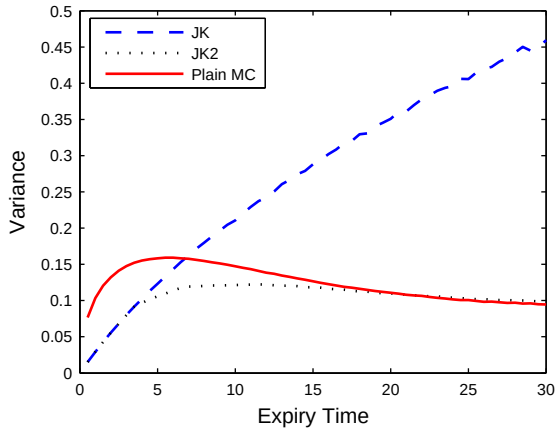
```
function [W, L] = gaussianJKImportanceVariates(rho, blockSize, mu, T, n2d)
% gaussianJKImportanceVariates Gaussian Importance sampling routine for default time simulations
%
% This function implements the importance sampling described in
%
%  M. Joshi and D. Kainth. Rapid and Accurate Development of Prices and
%  Greeks for nth to default credit swaps in the Li model.
%
% Parameters:
%              rho          Correlation Matrix
%              blockSize    The number of rows to be generated
%              mu           Exponential distribution parameter (1/h)
%              T            Expiry time
%              n2d          number of defaults
%
% Given a correlation matrix rho and blockSize it returns a
% blockSize x length(rho) matrix of appropriate variates
%
    if nargin < 5
       error('Not enough input arguments.');
    end

    n = length(rho);
    A = (chol(rho))'; % Cholesky factorization of rho (lower triangular AA' = rho)

    U = unifrnd(0,1,blockSize,n);

    V = zeros(blockSize,n);
    Z = zeros(blockSize,n);
    P = zeros(blockSize,n);
    L = ones(blockSize,1);        % per path IS adjustments
    defaultsSoFar = zeros(blockSize,1);
    ptilde = zeros(blockSize,1);

    extreme_T = norminv(expcdf(T, mu));

    for i=1:n
        P(:,i) = normcdf( (extreme_T(i) - (A(i,1:i-1)*Z(:,1:i-1)')) / A(i,i));

        q = (n2d - defaultsSoFar(:)) / (n + 1 - i);
        idx = defaultsSoFar >= n2d;
        ptilde(idx) = P(idx,i);
        ptilde(~idx) = q(~idx);

        idx2 = U(:,i) <= q(:);  % Note that for path with enough defaults q<=0!
        V(idx2, i) = (P(idx2,i) .* U(idx2,i)) ./ ptilde(idx2);
        L(idx2) = L(idx2) .* (P(idx2,i) ./ ptilde(idx2));

        V(~idx2, i) = P(~idx2,i) + ...
                      ( ((1 - P(~idx2,i)) ./ (1 - ptilde(~idx2))) .* (U(~idx2,i) - ptilde(~idx2)) );
        L(~idx2) = L(~idx2) .* ((1 - P(~idx2,i)) ./ (1 - ptilde(~idx2)));

        Z (:,i) = norminv(V(:,i));

        W(:,i) = expinv(normcdf(Z(:,1:i)*A(i,1:i)'), mu(i));
        idx3 = W(:,i) < T;
        defaultsSoFar(idx3) = defaultsSoFar(idx3) + 1;
    end
end
```

Table 7. JK Algorithm in Matlab

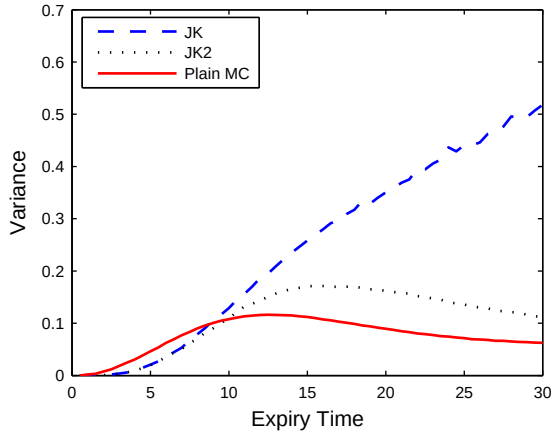


(a) Basket B0

(b) Basket B1



(c) Basket B2



(d) Basket B3

Figure 8. Variance of the MC and IS algorithm as function of swap maturity

Given the algorithm to simulate a T copula in Table 5, [**Joshi and Kainth, 2004**] remarks that at step **4** we are in the same situation of step **3** of the Gaussian copula simulation algorithm (see Table 1). Indeed we can repeat the reasoning that led to the gaussian importance sampling algorithm. At the core of the algorithm is the following Lemma which is the T-copula equivalent of Lemma 3.1.

Given N independent uniform variates $U_1, \ldots, U_N$, N independent normals $Z_1, \ldots, Z_N$ obtained by setting $Z_i = \Phi^{-1}(U_i)$ and $s \sim \chi^2_\nu$ we have the following lemma:

Lemma 3.2. Let $\Sigma = AA'$ be the positive definite correlation matrix. Let $\mathbf{W} = \sqrt{\frac{\nu}{s}}\mathbf{AZ}$, and $\tau_i = F_i^{-1}\left(T_\nu\left(W_i\right)\right)$ then:

$$\mathbf{P}\left(\tau_i \leq t | Z_1, \ldots, Z_{i-1}\right) = \Phi\left(\frac{\frac{T_\nu^{-1}(F_i(t))}{\sqrt{\nu/s}} - \sum_{j=1}^{i-1} A_{ij} Z_j}{A_{ii}}\right)$$

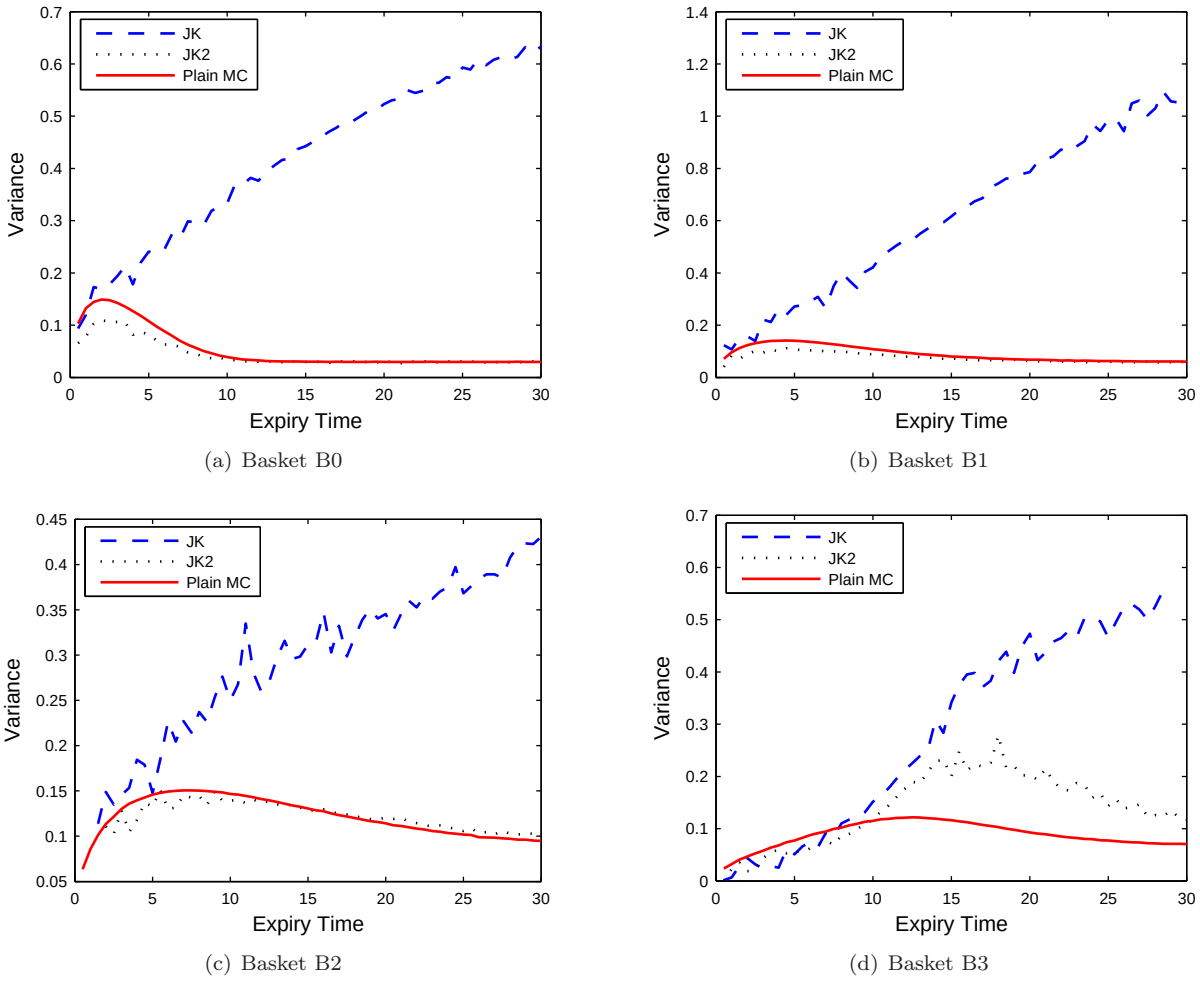
Proof. By definition:

$$\tau_i = F_i^{-1}\left(T_\nu\left(W_i\right)\right) = F_i^{-1}\left(T_\nu((AZ))_i\right)$$

Hence:

$$\begin{aligned}
\mathbf{P}\left(\tau_i \le t | Z_1, \ldots, Z_{i-1}\right) =& \mathbf{P}\left(F_i^{-1}\left(T_\nu(W_i)\right) \le t | Z_1, \ldots, Z_{i-1}\right) = \\
& \mathbf{P}\left(W_i \le T_\nu^{-1}\left(F_i(t)\right) | Z_1, \ldots, Z_{i-1}\right) = \\
& \mathbf{P}\left(\sqrt{\nu/s}\sum_{j=1}^{i} A_{ij} Z_j \le T_\nu^{-1}\left(F_i(t)\right) \middle| Z_1, \ldots, Z_{i-1}\right) = \\
& \mathbf{P}\left(\sum_{j=1}^{i} A_{ij} Z_j \le \frac{T_\nu^{-1}\left(F_i(t)\right)}{\sqrt{\nu/s}} \middle| Z_1, \ldots, Z_{i-1}\right) = \\
& \mathbf{P}\left(Z_i \le \frac{\frac{T_\nu^{-1}(F_i(t))}{\sqrt{\nu/s}} - \sum_{j=1}^{i-1} A_{ij} Z_j}{A_{ii}}\right)
\end{aligned}$$

From which the thesis follows. □



(a) Basket B0

(b) Basket B1

(c) Basket B2

(d) Basket B3

Figure 9. Variance of the MC and IS algorithm as function of swap maturity (T Copula)

Given the lemma we can now adapt the gaussian JK algorithm to the new copula. The corresponding Matlab code can be found in Table 8. The variance of IS algorithms under the T copula can be seen in Figure 9. We can see that under the T copula we the IS algorithms behave similarly as in the gaussian cases. However we do note a more erratic behaviour when the IS algorithms underperform.

4. Conclusion

We have implemented a Monte Carlo pricing engine for the n-th to default credit swaps based on the Li model. We have analysed the performance difference between a naive implementation of the pricing model and an optimised one based on the techniques presented in [**Joshi and Kainth, 2004**]. Our analysis reproduces the results of [**Chen and Glasserman, 2006**] although we did not stumble on swaps quite as extreme as the one they describe.

Finally we applied the technique of [**Joshi and Kainth, 2004**] to the T copula. In this instance we find similar variance reduction/increase as for the gaussian case, however, the IS algorithm does tend to show a more erratic behaviour in this context. An interesting follow-up to this work would be the implementation of the *conditional probability* algorithm of [**Chen and Glasserman, 2006**].

Acknowledgements

I would like to thank Aspect Capital Ltd. for sponsoring the second part of this MSc. Thanks also to Zhiyong Chen for providing prices for the baskets described in [**Chen and Glasserman, 2006**] which helped debugging our implementation. Thanks to Prof. Raymond Brummelhuis for proposing the subject of this dissertation and his supervision.

References

[jpm, 1999] (1999). *The J.P. Morgan Guide to Credit Derivatives*. J.P. Morgan and the RiskMetrics group.

[leh, 2003] (2003). *Guide to Exotic Credit Derivatives*. The Lehman Brothers.

[Bouye et al., 2000] Bouye, E., Durrleman, V., Nikeghbali, A., Riboulet, G., and Roncalli, T. (2000). Copulas for finance - a reading guide and some applications. Working Paper, Groupe de Recherche Operationnelle, Credit Lyonnais.

[Chen and Glasserman, 2006] Chen, Z. and Glasserman, P. (2006). Fast pricing of basket default swaps. submitted to Operations Research.

[Cherubini et al., 2004] Cherubini, G., Luciano, E., and Vecchiato, W. (2004). *Copula Methods in Finance*. John Wiley and Sons.

[Glasserman, 2003] Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering*. Springer.

[Higham, 2002] Higham, D. J. (2002). Nine ways to implement the binomial method for option valuation in matlab. *SIAM Rev.*, 44(4):661–677.

[Hull and White, 2004] Hull, J. and White, A. (2004). Valuation of a cdo and an n^{th} to default cds without monte carlo simulation. *Journal of Derivatives*, 12(2):8–23.

[Joshi and Kainth, 2004] Joshi, M. S. and Kainth, D. (2004). Rapid and accurate development of prices and greeks for nth to default credit swaps in the li model. *Quantitative Finance*, 4(3):266–275.

[Li, 1998] Li, D. X. (1998). Constructing a credit curve. *Credit Risk*, pages 40–44. A RISK special report.

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

[Merton, 1994] Merton, R. C. (1994). *Continuous-Time Finance*. Blackwell. Revised edition.

```
function [W, L] = TJKImportanceVariates(rho, nu, blockSize, mu, T, n2d)
% TJKImportanceVariates T Copula Importance sampling routine for default time simulations
%
% This function implements the importance sampling described in
%
%  M. Joshi and D. Kainth. Rapid and Accurate Development of Prices and
%  Greeks for nth to default credit swaps in the Li model.
%
% for a model using a T copula rather than a gaussian one.
%
% Parameters:
%              rho          Correlation Matrix
%              nu           Degrees of freedom
%              blockSize    The number of rows to be generated
%              mu           Exponential distribution parameter (1/h)
%              T            Expiry time
%              n2d          number of defaults
%
% Given a correlation matrix rho and blockSize it returns a
% blockSize x length(rho) matrix of appropriate variates
%
    if nargin < 6
       error('Not enough input arguments.');
    end

    n = length(rho);
    A = (chol(rho))'; % Cholesky factorization of rho (lower triangular AA' = rho)

    s = sqrt(nu ./ chi2rnd(nu));

    U = unifrnd(0,1,blockSize,n);

    V = zeros(blockSize,n);
    Z = zeros(blockSize,n);
    P = zeros(blockSize,n);
    L = ones(blockSize,1);        % per path IS adjustments
    defaultsSoFar = zeros(blockSize,1);
    ptilde = zeros(blockSize,1);

    extreme_T = tinv(expcdf(T, mu), nu) ./ s;

    for i=1:n
        P(:,i) = normcdf( (extreme_T(i) - (A(i,1:i-1)*Z(:,1:i-1)')) /  A(i,i));

        q = (n2d - defaultsSoFar(:)) / (n + 1 - i);
        idx = defaultsSoFar >= n2d;
        ptilde(idx) = P(idx,i);
        ptilde(~idx) = q(~idx);

        idx2 = U(:,i) <= q(:);  % Note that for path with enough defaults q<=0!
        V(idx2, i) = (P(idx2,i) .* U(idx2,i)) ./ ptilde(idx2);
        L(idx2) = L(idx2) .* (P(idx2,i) ./ ptilde(idx2));

        V(~idx2, i) = P(~idx2,i) + ...
                      ( ((1 - P(~idx2,i)) ./ (1 - ptilde(~idx2))) .* (U(~idx2,i) - ptilde(~idx2)) );
        L(~idx2) = L(~idx2) .* ((1 - P(~idx2,i)) ./ (1 - ptilde(~idx2)));

        Z (:,i) = norminv(V(:,i));

        W(:,i) = expinv(tcdf(s .* (Z(:,1:i)*A(i,1:i)'), nu), mu(i));
        idx3 = W(:,i) < T;
        defaultsSoFar(idx3) = defaultsSoFar(idx3) + 1;
    end
end
```

Table 8. JK Algorithm in Matlab for T Copula

Appendix A. Basket and Swap Definitions

Basket B0. This is a simple homogeneous basket:

N	10
Hazard rates	0.05
Recovery rates	0.3

The 10 assets are *independent*.

Swap S0. This is a F2D on B0.

Basket B1. Another simple homogeneous basket:

N	10
Hazard rates	0.05
Recovery rates	0.3

The asset correlation is uniform $\sigma_{ij} = 0.3$.

Swap S1. This is a S2D on B1.

Basket B2. This is basket I in [Chen and Glasserman, 2006].

N	10
Hazard rates	$\{0.03, 0.01, 0.02, 0.01, 0.005, 0.001, 0.002, 0.002, 0.017, 0.003\}$
Recovery rates	$\{0.3, 0.1, 0.2, 0.1, 0.3, 0.1, 0.2, 0.2, 0.1, 0.3\}$

The 10 assets are *independent*.

Swap S2. This is a F2D on B2 (the same swap as A1 in [Chen and Glasserman, 2006]).

Basket B3.

N	10
Hazard rates	$\{0.05, 0.01, 0.02, 0.02, 0.03, 0.1, 0.03, 0.09, 0.1, 0.05\}$
Recovery rates	$\{0.3, 0.1, 0.2, 0.1, 0.3, 0.1, 0.2, 0.2, 0.1, 0.3\}$

Their correlation matrix has been randomly generated using the Matlab command `gallery('randcorr', N)`:

$$\mathbf{\Sigma} = \begin{pmatrix} 1.0000 & -0.1960 & 0.0762 & 0.1940 & 0.1381 & -0.0463 & -0.0077 & -0.0070 & -0.0848 & 0.0437 \\ -0.1960 & 1.0000 & 0.0719 & 0.1279 & 0.0728 & 0.1299 & -0.0816 & 0.0736 & 0.1918 & 0.0466 \\ 0.0762 & 0.0719 & 1.0000 & -0.2391 & -0.1119 & -0.2278 & -0.1650 & 0.0743 & -0.0742 & -0.2371 \\ 0.1940 & 0.1279 & -0.2391 & 1.0000 & -0.1268 & -0.1145 & 0.0736 & -0.0242 & 0.0030 & -0.0852 \\ 0.1381 & 0.0728 & -0.1119 & -0.1268 & 1.0000 & 0.0250 & -0.0108 & -0.1825 & 0.0446 & -0.1387 \\ -0.0463 & 0.1299 & -0.2278 & -0.1145 & 0.0250 & 1.0000 & -0.0688 & 0.0503 & -0.1699 & -0.0387 \\ -0.0077 & -0.0816 & -0.1650 & 0.0736 & -0.0108 & -0.0688 & 1.0000 & 0.1445 & 0.0273 & -0.0683 \\ -0.0070 & 0.0736 & 0.0743 & -0.0242 & -0.1825 & 0.0503 & 0.1445 & 1.0000 & 0.0129 & 0.0762 \\ -0.0848 & 0.1918 & -0.0742 & 0.0030 & 0.0446 & -0.1699 & 0.0273 & 0.0129 & 1.0000 & -0.2342 \\ 0.0437 & 0.0466 & -0.2371 & -0.0852 & -0.1387 & -0.0387 & -0.0683 & 0.0762 & -0.2342 & 1.0000 \end{pmatrix}$$

Swap S3. This is a 5th to default swap on B2.

Appendix B. Matlab code

The Matlab project associated with this paper is available online at

http://www.brics.dk/~milicia/libraryGM.htm.

The algorithms presented in this paper are at the core of the system, however using the pricing engine is rather user-friendly and hides most of the details we have described.

The entry point of the pricer is the function `n2d`. The function can be used in three different manners:

- We can pass as input to the function a *swap* object.
- We can pass as input to the function a *basket* object and a number of parameter to complete the definition of the swap.
- Raw data. We can pass to the function all the input required to define a swap. This style of usage is particularly useful when performing parameter sweeps and scenario analysis on swaps.

Basket objects. A basket object is a collection of *assets* each with its own hazard and recovery rate. It also contains a correlation matrix.

Swap objects. A swap object is a basket and a series of parameter that fully specify the contract:

- Yearly interest rates
- Maturity in years
- Number of defaults

Usage

```
%    [Numerical, StandardError] = n2d(swap, varargin) Calculates the price of the given swap.
%
%    [Numerical, StandardError] = n2d(basket, r, T, defaults, varargin)
%    basket   The basket on which the N2D is defined
%
%    [Numerical, StandardError] = n2d(h, recovery, rho, r, T, defaults, varargin)
%
%    N        number of assets
%    h        hazard rates
%    recovery recovery rates as either a scalar (uniform recovery rates) or a
%             vector of size N
%    tau      correlation parameter
%    r        (yearly) interest rates
%    T        Years to expiry
%    defaults Number of defaults
%
% Optional parameters:
%
% type     (optional, default 'plain') The type of simulation to be performed. Currently we support
%          'plain'  Plain Monte Carlo
%          'JK'  Joshi Kainth Importance sampling algorithm
%          'JK2' The modified JK algorithm
%
% copula   (optional, default 'gaussian') The copula to use. We support 'gaussian' and 'T'
%          Note that the T copula needs as final argument nu (degrees of
%          freedom)
%
% runs     (optiona, default 50000) Number of MC runs to perform
%
```