Pricing Credit Risk Derivatives with R

Giuseppe Bruno

Bank of Italy, Economics and Statistics, Rome, Italy

Copyright ©2016 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

Abstract This paper shows the implementation of some pricing techniques for multiname credit derivatives. Among these financial instruments we consider Basket Default Swaps (BDS) and Collateralised Debt Obligations (CDO). The pricing methodologies put forward are based on Monte Carlo simulations. These techniques are written in the **R** programming language. This software framework provides an ample set of functions for credit risk modelling. Two main issues are put forward in this paper: the first one is the employment of Quasi Random Number (QRN) for reducing the variance of price estimate in the Monte Carlo experiment; the second one is the code parallelization of some pricing techniques using **R** end-user techniques.

Keywords Collateralized Debt Obligation, Monte Carlo methods, One Factor Copula Models.

1 Introduction

Since the end of the eighties new financial instruments have arisen with the purpose of allowing the exchange of credit risk. These instruments are generically dubbed credit derivatives. Among the most important instruments traded in this market there are: Credit Defaults Swaps (CDS), Basket Default Swaps (BDS) and Collateralised Debt Obligations (CDO) (for example see Gibson[5], Acharya[1] and Bluhm[2]).

Credit Default Swaps (CDS) are the simplest risk exchanging instruments. These are single name credit derivative contractual agreements yielding a payoff function of the credit quality of a third party called the obligor¹. Analysis of default time of the obligor has been tackled also with Neural Networks Models (see for example A. Slavova[10]). Other financial instruments of this kind are the multiname credit derivatives such as Basket Default Swaps (BDS) and Collateralised Debt obligations (CDO). Here the term multiname refers to the pooling of credits from different borrowers. Fathi and Nader[4] which employ Gaussian copula and Monte Carlo simulation for pricing BDS and CDO provide Matlab code for their examples. In this paper we will analyze the software implementation of some pricing algorithms

for these multiname credit derivatives. Many useful examples of implementation issues can be found in Fouque[7]. In this work, the software tool employed for these computations is the statistical package *R*. This is an open software product quite spread among the statistical community but much less known in financial economist circles.

This work will show the feasibility and the performance capabilities of the *R* software in dealing with these pricing tasks. Furthermore we will also put forward some methods for effectively reducing the pricing estimation variance when employing Monte Carlo methods. Among the available techniques for variance reduction in Monte Carlo simulation, the replacement of standard pseudo random number with the quasi random number sequences has proved a robust methodology. Löffler and Posch[8] propose their use for credit risk modeling in the Visual Basic framework. In the software framework provided by *R* we have implemented a Monte Carlo pricing procedure taking advantage of the Quasi Random Number (QRN) generators featured in the package **randtoolbox**. As far as our knowledge goes this is the first time the QRN methodology is applied for pricing multiname credit derivative instruments. As a final point we propose a preliminary path for a software framework taking advantage of the multi-core/processor architecture of the computing platforms. The rest of the paper is organized in the following way. In section 2 we present the basic technical concepts on the credit derivatives instruments. In the following section we provide the modelling framework for pricing multiname credit derivatives. Section 4 puts forward the methodologies for reducing the pricing estimation variance. Section 5 shows the preliminary results achieved through the code parallelization within the *R* framework. Finally section 6 provides some concluding remarks.

2 Credit derivatives products and their role in the banking balance sheet

In this paper we will take into account the problem of implementing the pricing of multiname credit derivatives such as Basket Default Swaps (**BDS**) and Collateralized Debt Obligation (**CDO**). Both of these credit derivatives are based

¹There is just one asset and therefore a single name obligor

on an insurance-like contract with two payment legs. The **BDS** are made up by a pool of credits referring to different names (obligors) which is split in two by a threshold value of number of acceptable defaults. The losses below the threshold are borne by **BDS** issuer (the protection buyer). The default exceeding the threshold will trigger the contingent leg and the end of the contract. The **CDO** instruments consist of a portfolio of defaultable assets such as loans, credit and bonds whose aggregate credit risk is first split into a given number of tranches of increasing seniority. The defaults occurring during the instrument lifetime will generate losses hitting first the most junior tranche, called the equity tranche, successively the losses will start eating up the notional of the more senior tranches. Each tranche will be responsible for the losses within a given percentage interval defined by its attachment and detachment points. These tranches are then separately sold to investors with different risk profiles. Either of these instruments are contractual agreements between a protection purchaser, the issuing organization, and a protection seller, which is the investor. These contractual agreements bind the two parties for a fixed time period ranging from 1 to usually no more than 10 years.

These financial instruments can be compared to a standard insurance contract where we can pin down two separate payment flows:

1. the periodic premium leg (borne by the protection buyer);
2. the default leg (borne by the protection seller).

The protection buyer owns a pool of defaultable assets for which he wants to buy protection against a credit event (i.e. a default). The premium leg constitutes the periodic insurance premium while the default leg conveys the ensuing payments following a default. The structure of the agreement can be better understood by referring to the following figure:

Figure 1. Credit derivative payments legs.

The credit risk is shifting away from the protection buyer to the protection seller on the right section of the picture. The payments flowing through the premium leg are settled on a predefined schedule, usually there are quarterly payments in arrears. Assuming a pool of assets with an initial notional amount of $M = \sum_{j=1}^N P_j$ composed of N obligors, where each asset has the nominal value P_j and a set of regularly spaced payment dates, the amount of each installment will be given by

$$PL_i = s \cdot \overline{PL_i} = s \cdot e^{-r(T-T_i)} \sum_j^N P_j \cdot \mathbf{1}_{\{\tau_j > T_i\}} \quad (1)$$

where: T is the expiration time for the instrument, T_i is the current time, $e^{-r(T-T_i)}$ is the discount factor with an assumed constant interest rate (r), s is the fair spread required to compensate for the burden of the credit risk, τ_j is the default time for obligor j , $\mathbf{1}_{\{\tau_j > T_i\}}$ is the event indicator function which takes on values 1 or 0 according to whether the event $\{\tau_j > T_i\}$ occurs or not. This event indicator signals

the occurrence of the default for obligor j . On the default leg there is a flow of payments going in the opposite direction. We will assume these payments occur necessarily on the first premium payment date following the default event. The amount foreseen for the default leg is given by

$$DL_i = e^{-r(T-T_i)} \sum_j^N P_j \cdot (1 - \delta_j) \cdot \mathbf{1}_{\{\tau_j \leq T_i\}} \quad (2)$$

where: δ_j is the recovery rate for the default of the asset whose notional amount is P_j . Therefore the default leg conveys the discounted sum of the reimbursement for all the assets defaulted before T_i . The task of pricing this kind of financial instruments consists in the determination of the fair premium required for insuring a zero value of the contract at its inception. Assuming $t_0 = 0$ we will have

$$V(t_0) = -s \cdot \mathbb{E}_{t_0} [\overline{PL_i}] + \mathbb{E}_{t_0} [DL_i = 0] \Rightarrow s = \frac{\sum_j^N (1 - \delta_j) \cdot P_j \cdot \mathbb{E}_{t_0} [\mathbf{1}_{\{\tau_j \leq T_i\}}]}{\sum_j^N P_j \cdot \mathbb{E}_{t_0} [\mathbf{1}_{\{\tau_j > T_i\}}]} \quad (3)$$

In equation 2 we have taken the standpoint of the protection buyer in giving the negative sign in the premium leg and the positive sign to the default leg. For a **BDS** we have either no payment or just one payment on the default leg. Assuming the default payment is triggered by the n_h default. If this default happens at time τ^h then the spread rate can be computed in a similar way according to the following

$$V(t_0) = -s \cdot \mathbb{E}_{t_0} [\overline{PL_i}] + \mathbb{E}_{t_0} [DL_i] = 0 \quad (4)$$

by solving (4) with respect to s we get

$$s = \frac{(1 - \delta) \cdot \mathbb{E}_{t_0} [\mathbf{1}_{\{\tau^h \leq T\}}]}{\sum_j^N P_j \cdot \mathbb{E}_{t_0} [\mathbf{1}_{\{\tau^h > T_i\}}] + (\tau^h - T_{i-1}) \mathbb{E}_{t_0} [\mathbf{1}_{\{T_{i-1} < \tau^h < T_i\}}]} \quad (5)$$

where we have assumed a common recovery rate δ among the obligors and a premium paid only on the surviving assets.

3 Pricing Multiname credit derivatives

In this section we examine the computational issues involved in pricing multiname credit derivatives belonging to the category of **BDS** and **CDO**.

We assume a portfolio of N names of defaultable assets. The **BDS** is a natural extension to the single name **CDS**. The basic idea is that of purchasing protection on a basket of different names. The two parties agrees on the event that will trigger the contingent payment. This is the default of order n_h where $n_h \in \{1, 2, \dots, \hat{N}\}$ with $\hat{N} < N$ default in the basket. For the **CDO** instruments this portfolio is sliced in J tranches of progressively increasing seniority. These tranches are defined by their two boundaries values, normalized to the total portfolio value, called the attachment and the detachment points: K_{L_j} and K_{U_j} . The first attachment and the last detachment points will always be $K_{L_1} = 0$ and $K_{U_J} = 1$.

The problem of pricing multiname credit derivatives consists of determining the break-even premium s_j for each tranche $j \in \{1, 2, \dots, J\}$ composing the whole capital structure. This premium is the spread value s_j referred in chapter 2 which makes zero the present value of the balance between

the two payments flowing of fig. 1. Therefore the crucial element required for pricing these instruments is the distribution function of the total portfolio losses defined as

$$L(t) = \sum_{j=1}^N P_j \cdot (1 - \delta_j) \cdot \mathbf{1}_{\{\tau_j < t\}}. \quad (6)$$

For the following explanation it is useful to introduce also the total percentage loss given by $Z_t = \frac{L(t)}{\sum_{j=1}^N P_j}$. The premium or fixed leg of the contract represents the payments received by the tranche holders (insurance sellers), whereas the default or floating leg constitutes the payments made by the tranche holders to the protection buyers in case a default has occurred. Let us assume a financial instrument of initial notional amount M whose contractual premium payment dates belong to the set $\mathcal{T} = \{t_1, t_2, \dots, t_K\}$ where $\Delta = t_k - t_{k-1} \quad \forall t_k \in \mathcal{T}$, i.e. equally spaced intervals.

At each payment date $t_k \in \mathcal{T}$, the premium to pay depends on the stock of survived assets of the tranche: $\Gamma_{j,k} = (K_{U_j} - K_{L_j} - Z_{j,k}) \cdot M$, that is the difference between the initial tranche percentage size and the current percentage tranche loss $Z_{j,k} = \lceil \min \{Z_t, K_{U_j}\} - \min \{Z_t, K_{L_j}\} \rceil$. The protection seller of tranche j receives a premium whose amount is $s_j \cdot \Delta \cdot \Gamma_{j,k}$. Therefore the present discounted value of the premium leg for tranche j will be:

$$V_{PL}^j = s_j \cdot \overline{V_{PL}^j} = s_j \cdot \sum_{k=1}^K e^{-(t_k - t_0)} \cdot \Delta \cdot \mathbb{E}[\Gamma_{j,k}] \quad (7)$$

that is the sum over the life span of the instrument of the average tranche value net of losses. On the opposite side of the contractual agreement, in each date t_k the protection buyer will receive a compensation proportional to the loss occurred in the interval $[t_{k-1}, t_k]$ which is $[(Z_{j,k} - Z_{j,k-1}) \cdot M]$. Thus, the present discounted value of the default leg for the same tranche j will be:

$$V_{DL}^j = \sum_{k=1}^K e^{-(t_k - t_0)} \cdot \mathbb{E}[(Z_{j,k} - Z_{j,k-1}) \cdot M] \quad (8)$$

The break-even spread s_j in equation 7 can be computed as the value which ensures a present value of the tranche equal to zero at contract inception. The value of the tranche over time will be given by the following equation:

$$V^j(t) = -s_j \mathbb{E}_t \left[\overline{V_{PL}^j} \right] + \mathbb{E}_t \left[V_{DL}^j \right] \quad (9)$$

by setting to zero the left handside of equation 9 we can solve for the spread value which is given by $s_{j,Par} = \mathbb{E}_t \left[\overline{V_{DL}^j} \right] / \mathbb{E}_t \left[\overline{V_{PL}^j} \right]$. By employing the R software framework we have developed a program employing the single factor copula function for pricing a CDO composed of a set of 125 obligors with an homogeneous default intensity and recovery rate². The structure of this CDO reflects closely that one of the iTraxx and CDX family of indices³. In our empirical application, we have employed a CDO with the following parameters:

1. annual default intensity $\lambda = .03$
2. payment period (normalized to a year) $\Delta = .25$

3. recovery rate $R = .4$

and the portfolio losses are distributed in six tranches as described in the following table:

Table 1. Tranche structure of the priced CDO

Tranche number	Tranche name	Attachment points (%)	
		attachment	detachment
1	Equity	0	.03
2	Mezz 1	.03	.07
3	Mezz 2	.07	.1
4	Mezz 3	.1	.15
5	Senior	.15	.3
6	Sup Senior	.3	1

The estimation outcome is summarized in the following table:

Table 2. Price of the different tranches in base points

ρ	Equity	Mezz 1	Mezz 2	Mezz 3	Senior	Sup Senior
.08	869.08	240.49	87.07	24.78	1.39	0.0
.18	631.60	196.09	88.54	38.08	6.14	0.02
.28	487.17	168.27	86.65	45.14	11.14	0.15
.38	382.57	146.33	81.78	47.96	15.97	0.45
.48	307.82	127.39	76.05	49.23	20.45	0.89
.58	250.16	112.05	70.97	49.15	24.02	1.50
.68	202.37	96.69	65.22	47.64	26.50	2.37
.78	161.20	82.04	58.97	46.39	27.94	3.35
.88	123.80	69.28	53.72	44.94	28.94	4.51
.98	88.12	58.46	48.42	41.08	28.67	5.81

Table 2 shows the prices of a six tranche CDO over a grid of ten increasing values for default correlation ρ among every couple of obligors. The results have been obtained with the adoption of a t copula with 10 degrees of freedom. In table 2 we see a regular behavior of the tranche price as a function of the tranche seniority and the default correlation. The equity and tranche mezz 1 show a monotone decreasing behavior with the default correlation. This is consequence of the default clustering caused by higher correlation. The two higher mezzanine and the senior tranches show an hump-shaped behavior. The super senior tranche presents a spread monotonically increasing with the default correlation. The growth of the default correlation increases the likelihood of a higher number of clustered defaults hitting even the super senior tranche. The R programs provide routines for BDS and CDO pricing sampling from different kind of copulas such as elliptical (e.g. Gaussian and t) and archimedean (e.g. Clayton, Gumbel and Frank).

4 Variance reduction techniques for CDO pricing

The various CDO pricing algorithms available in literature are concerned with the computation of credit derivatives prices and their sensitivities to variation of the different model parameters. Closed form analytic methods provide deterministic values for the prices. When Monte Carlo Methods are employed the achieved prices are averages of given probability distributions. In these cases it is of paramount importance to integrate the average measures with an index of

²Some code snippet are shown in Appendix 7.1. The whole R program is available upon request.

³iTraxx indices cover European and Asian credit exposures, whereas CDX indices cover the North American credit risk market.

dispersion. In this work we employed the R features to compute averages and standard deviation of the Monte Carlo experiments. Different algorithms yield different standard deviations. One of the main goals of the analysis of Monte Carlo methods is to achieve the optimal trade-off between numerical accuracy and computational time. In other words, given a particular numerical accuracy we would like to minimize the number of Monte Carlo replications. Variance reduction techniques aim at making the most efficient use for each Monte Carlo replication. For the computation of **CDO** prices, besides the basic approach, we have used two families of variance reduction algorithms:

1. plain Monte Carlo experiments;
2. Monte Carlo experiments with antithetic variates;
3. Monte Carlo experiments with Quasi Random Numbers (QRN).

The use of antithetic variates is based on the principle of drawing the samples with a given negative correlation to reduce the total sample variance⁴. In our empirical applications, for the Gaussian copula model, we have generated the default times assuming an exponential distribution function for the survival time: $S_j(t) = e^{-h_j t}$. In this case the default times can be simulated in a straightforward way by

$$\tau_j^\alpha = -\frac{1}{h_j} \ln[\Phi(x^\alpha)] \quad (11)$$

where: α is the replication count, h_j is the asset j default intensity⁵ and the sample x is drawn from a standard normal distribution function whose cumulant is indicated as $\Phi(\cdot)$. When we adopt the antithetic variates only $\frac{n}{2}$ of the required sample are drawn from standard normal sample: $Z_1, Z_2, \dots, Z_{n/2}$ the remaining $\frac{n}{2}$ values are taken as their opposite: $Z_{n/2+1} = -Z_1, Z_{n/2+2} = -Z_2, \dots, Z_n = -Z_{n/2}$. In this way we take advantage of the negative correlation present in the sample for decreasing the variance of the average. Aside from the difference in the random number sequences, the simulation code remains exactly the same. Quasi-Monte Carlo methods can be defined as the deterministic counterpart of the Monte Carlo simulation techniques. While the latter use random or pseudo-random, the former employ some kind of deterministic sequences called Low Discrepancy Sequences (henceforth LDS). One of the most critical issues with Monte Carlo methods is the generation of good random points. In practical applications, software routines can only generate pseudo-random sequences, which, especially in higher dimensions, can lead to clumping of points. The accuracy of integration for methods based on pseudo-random points can be substantially improved by using more uniformly distributed sequences. The employment of LDS sequences has proved an effective avenue for generating high dimensional quasi random number suitable for Monte Carlo experiments. Different techniques for generating these sequences have been proposed in the literature. To the best

⁴The variance of the average of two random variables, X_1 and X_2 is given by:

$$Var[\frac{1}{2}(X_1 + X_2)] = \frac{1}{4}[Var(X_1) + Var(X_2)] + \frac{1}{2} \cdot Cov(X_1, X_2) \quad (10)$$

therefore the variance is reduced when $Cov(X_1, X_2) \leq 0$

⁵The default intensity is assumed constant.

of our knowledge these QRN numbers have never been employed for carrying out Monte Carlo experiment for **CDO** pricing.

In our empirical application we have employed the package **randtoolbox** (see Chalabi[3] 2013 for details). This package has allowed us to carry out experiment with the Halton (Halton[6]) and the Sobol sequences (see Sobol[9]). In our experiments we have compared the **BDS** and **CDO** pricing and their standard deviation with classical pseudo-random and antithetic variates samples.

In the following table we show a comparison among four different Monte Carlo techniques for pricing a **BDS** of 100 names up to the third default in the portfolio:

Table 3. Pricing Basket Default Swap with classical and QRN Monte Carlo

Nrep	Nruns	Pseudo-Random	Antithetic var	Halton	Sobol
10	$5 \cdot 10^3$	1640.4 (38.6)	1654.4 (10.7)	1675.3 (14.3)	1685.6 (6.9)
10^2	$5 \cdot 10^3$	1647.3 (40.8)	1650.9 (11.7)	1684.0 (9.5)	1682.1 (3.3)
10^3	$5 \cdot 10^3$	1648.4 (40.8)	1654.3 (5.7)	1683.7 (9.7)	1686.0 (6.8)
10	10^4	1648.9 (20.6)	1648.4 (9.2)	1682.0 (7.2)	1681.7 (5.7)
10^2	10^4	1654.7 (20.2)	1650.3 (3.3)	1683.8 (6.1)	1680.9 (4.9)
10^3	10^4	1638.6 (38.7)	1650.2 (0.6)	1683.9 (6.4)	1685.1 (3.8)

The relevant section for the R code is shown in appendix 7.1. The results presented in table 3 show a clear advantage in the use of the Halton and the Sobol QRN sequences for reducing the estimate standard deviation. Our empirical application shows also a clear dominance of the antithetic variates over the classical Monte Carlo methods. Table 4 shows the elapsed time required for the different algorithms⁶.

Table 4. Execution time (in sec.) for pricing a BDS with classical and QRN Monte Carlo

Nrep	Nruns	Pseudo-Random	Antithetic var	Halton	Sobol
10	$5 \cdot 10^3$	16.3	15.7	20.2	20.8
10^2	$5 \cdot 10^3$	180.7	168.8	223.3	217.9
10^3	$5 \cdot 10^3$	1736	1683	2232	2179
10	10^4	39.8	35.3	48.5	53.1
10^2	10^4	385	361	497	473
10^3	10^4	3575	3482	4784	4712

In employing the QRN sequences, we have only experienced an increase of about 20% in the computing times for the Halton and Sobol sequences, Monte Carlo experiments with the antithetic variates require a computing time about 8% smaller than a standard pseudo-random number procedure.

⁶We ran all the computation on a HP BL685 G7 Blade server with a 2.2 GHz, eight core four socket AMD Processor 6132 HE. Memory type DDR3 1333 MHz. Cache size 512 KB.

5 The parallelization experiences with R

The R programming environment provides many packages addressing the issues of increasing the computing performances taking advantage of the parallelism present in the CPUs. In this paper we have tested three different parallel back-ends provided by R under the UNIX/LINUX Operating Systems⁷. These three different back-ends are equipped with an interface for the R **foreach** package. The following table summarizes the parallel frameworks employed:

Table 5. Parallel backend and **foreach** interfaces

Parallel backend	interface	Kind of parallelization
multicore	doMC	fork system call based
SNOW	doSNOW	socket/pipe based
parallel	doParallel	merge of multicore & SNOW

The **foreach** package provides a looping construct to execute R code repeatedly. The main advantage in employing the **foreach** package is that it allows both serial and parallel execution. A **foreach** loop can execute the operations included in the body of a loop on multiple processors/cores of a computing platform, or on multiple nodes of a cluster.

The **multicore** package is aimed towards a shared memory platform. It dispatches parallel tasks on the different cores of a single computing node. The **SNOW** package is geared for a distributed memory platform⁸. Finally the **parallel** package provides a hybrid parallel back-end that is a combination of **multicore** and **SNOW**. By default, the **parallel** package adopts the multicore functionality on UNIX-like systems and **SNOW** functionality on Windows platforms. In our empirical application we have estimated the price of a *CDO* and a *CDO*² (*CDO* squared) instrument within the same program. The computing time results are summarized in the following table:

Table 6. CPU time Comparison among the parallel backend available with **foreach** package

Nreplic	# cores	Copula kind	Scalar code	Byte-compiled	multicore	SNOW	parallel
10 ³	30	Gaussian	39.7	39.2	3.7	71.6	61.9
		t	7.8	7.7	2.7	67.9	61.6
		Clayton	4.3	4.3	2.4	69.2	60.8
5 · 10 ³	30	Gaussian	201.3	201.5	17.6	91.1	80.4
		t	41.3	41.0	10.2	87.0	75.3
		Clayton	21.9	22.4	9.5	86.7	72.7
10 ⁴	30	Gaussian	399.6	397.	35.4	116.1	105.0
		t	76.5	75.1	20.3	110.6	93.6
		Clayton	42.9	43.7	17.9	109.7	93.6
2 · 10 ⁴	30	Gaussian	774.	798.4	167.9	126.3	157.3
		t	160.4	168.6	41.4	159.8	133.1
		Clayton	83.3	83.8	36.9	158.9	126.3

A code snippet describing the main sections employed in the code parallelization is shown in Appendix 7.2. The performance results present some confirmations and leave some questions. First of all, on a single computing machine the use of **multicore** outpaces both the **SNOW** and the **parallel** packages. This is a consequence of the fact that **multicore** employs the faster pipes instead of the sockets for communication among tasks, furthermore the time required to fork a process is smaller than that required for the initialization of

new processes required by the other two back-ends. In some instances the wide difference should be deeply investigated at the compiler level.

6 Conclusions

In this paper we have considered some methods for pricing multivariate credit risk derivatives instruments such as **BDS** and **CDO**. We have shown the feasibility and the performance of these algorithms when they are written in the open source package R. We have shown some R procedures implementing different variance reduction techniques for pricing **CDO**. In particular, we have put forward the QRN technique as a strategy to effectively decrease the variance of the price estimate of the credit risk derivatives. In the preliminary empirical analysis we have verified the superiority of the Sobol QRN sequence in achieving the lowest sampling variance in many examples. Also the antithetic variates Monte Carlo simulation has proved very effective achieving, in some cases, the smallest variance. Besides this algorithmic proposal, we have shown some of the possible avenues for writing a parallel version of these algorithms by employing three of the most widespread parallel back-ends provided in R. Our implementation has shown that by employing the **multicore** package we reached the shortest execution time. We have also tried the features of the **compiler** package which provides a bytecode translation of the original R code. In these pricing procedures the use of the bytecode yielded an improvement of less the 20%. A relevant avenue for further analysis should include some sort of code adaption for testing the performance on computing platforms equipped with numerical boards such as a Graphical Processing Unit (GPU) or Many Integrated Core (MIC).

7 Appendix-Code snippets for the empirical examples

7.1 Pricing comparison among different Random number generation methods

Here it is shown the relevant code section for the multiple loop over runs and replications with two different QRN sequences (Halton and Sobol), the antithetic variates and the classical pseudo random number generation.

```
for (runs in seq(5000,10000,by=5000))
{
  for (nreps in c(10,100,1000)) {
    z1sob <- replicate (nreps,
      Gausqsob(N, k, runs, rho, T, r,
        R, h, delta, nsubs, sobol ))
    etiso <- proc.time()[3] - tin[3]

    # We must initialize the Halton
    sequence to avoid mistakes in the
    following section
    halton(runs,dim=N,normal=T)
    tin <- proc.time()
    z1hal <- replicate (nreps, Gaussqrn
      (N, k, runs, rho, T, r, R, h,
        delta, nsubs, halton ))
```

⁷Windows operating systems do not provide the fork system call and this reduces the parallelization opportunities.

⁸The name stands for Simple Network Of Workstations.

```

etiha <- proc.time()[3] - tin[3]

z1pse <- replicate(nreps, Gaussn2Mc
  (N, k, runs, rho, T, r, R, h,
   delta, nsubs))
z1ant <- replicate(nreps, Gausanti(
  N, k, runs, rho, T, r, R, h,
  delta, nsubs))

sink(file="outsimu.log", type="
  output", split=T)
remat <- cbind(mean(z1sob), mean(
  z1hal), mean(z1ant), mean(z1pse))
sdmat <- cbind(sd(z1sob), sd(z1hal)
  , sd(z1ant), sd(z1pse))
resmat <- rbind(remat, sdmat)
}}

```

The outermost *loop* of the code controls the length of each simulation run. The innermost *for* loops controls the number of time each Monte Carlo experiment is replicated. Each replication is started with a different seed or a different activation number for the QRN sequences.

7.2 Parallelizing the code for CDO pricing within a single factor copula framework

```

p_d <- .05
copula <- 'Clayton'
para <- .2^2
Rr_r <- .5
CD0_attp <- c(0, .05, .15, 1)
CD2_attp <- c(0.05, .15)
runs <- 10000
##### snip #####
library(copula)
library(doMC)
registerDoMC(cores=30)
##### snip #####
gen_cdo <- function(copula, para, a_pool,
  n_pool, p_d, rr) {
  default <- colSums(t(coprnd(copula,
    para, a_pool, n_pool) <= p_d))
  n_pool - (1-rr)*default
}
rsl <- foreach(icount(runs), .combine=
  cbind) %dopar% {
  gen_cdo(copula, para, a_pool, n_pool,
    p_d, rr)
}

```

At the beginning we find a parameter initialization section. Afterwards we have the loading of the required libraries and the initialization of the parallel back-end. Eventually we have the definition of the computing function and the parallel

foreach loop with its multithreads firing statement **dopar**.

Acknowledgements

The author wishes to thank Roberto Stok for his helpful suggestions and unceasing cooperation. The views expressed are the authors only and do not imply those of the Bank of Italy.

REFERENCES

- [1] V. V. Acharya, T. C. Johnson. Insider trading in credit derivatives. *Journal of Financial Economics* 4, 110 – 141, 2007.
- [2] C. Bluhm. CDO modelling: techniques, examples and applications. *Hypo Vereinsbank (1)*, 2003.
- [3] Y. Chalabi. Randtoolbox. <http://cran.r-project.org/web/packages/randtoolbox> 2013.
- [4] A. Fathi, N. Nader. Copula based simulation procedures for pricing basket credit derivatives, Munich Personal RePEc Archive I, 1 – 30, 2007.
- [5] M.S. Gibson. Understanding the Risk of Synthetic CDOs. FRB Working Paper, 2004.
- [6] J. H. Halton. On the efficiency of certain quasi random sequences of points in evaluating multidimensional integrals. *Numerische Mathematik* 2, 84 – 90 , 1960.
- [7] J. P. Fouque, T. F. and K. Solna. *Econometrics and Risk Management*. Stanford: Emeralds JAI, 2008.
- [8] Loeffler, G. and P. N. Posch (2010). *Credit risk modeling using Excel and VBA* (2 ed.). Wiley.
- [9] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals, *U.S.S.R Computational Mathematics and Mathematical Physics* 7(4), 86 – 112 , 1967.
- [10] A. Slavova. Cellular Neural Networks Model of Risk Management Risk Management, IEEE proceedings of 11th International Workshop on Cellular Neural Networks and their application (4), 181 – 185 , 2008.