# *CQF Final Project*

A DISSERTATION PRESENTED
BY
**VAMSHI KRISHNA JANDHYALA**

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
**CERTIFICATE OF QUANTITATIVE FINANCE**

FITCH 7CITY LEARNING
LONDON, UNITED KINGDOM
JANUARY 2013

Thesis advisor: Richard Diamond **Vamshi Krishna Jandhyala**

# *CQF Final Project*

## ABSTRACT


### PRICING BASKET CREDIT DEFAULT SWAPS BY COPULA

We discuss the implementation of Gaussian and Student's t copula models for pricing $k-th$ to default basket credit default swap contracts. Calibration procedures are also provided to construct a credit curve from CDS premiums for a reference entity and to optimize parameters of the copula based on historical pricing data for the reference entities of a contract.We discuss the sensitivity of the spread to several key factors, including the credit quality of the reference entities, the default correlation among the reference entities, the recovery rate, and the discount factors.

### IMPLEMENTATION OF HJM MODEL

We discuss an implementation of the Heath-Jarrow-Morton model for pricing zero-coupon bonds and caps by Monte Carlo simulation. Principal component analysis is used to estimate volatilities of the model from historical time series data for forward rates. Models with as many as 10 factors are supported.

# Contents

# PartI

# Pricing Basket Credit Default Swaps by Copula

# 1 Pricing Basket CDS

## 1.1 K-TH TO DEFAULT BASKET CDS

A BDS is a default swap in which the credit event is the default of some combination of the credits in a specified basket of credits (typically from 3 to 5 names). In the particular case of a first-to-default basket(1st to Def), it is the first credit in a basket of reference obligors whose default triggers a payment to the protection buyer. As in the case of a (single name) default swap, this payment may be cash settled. More commonly, it will involve physical delivery of the defaulted asset in return for a payment of the par amount in cash. In return for protection against the 1st to Def, the protection buyer pays a basket spread to the protection seller as a set of regular accruing cash flows. As with a default swap,

these payments terminate following the first credit event. Similarly other credit products may be defined such as a second-to-default basket which triggers a credit event after two or more obligors have defaulted, and so on from the nth-to-default basket until the last-to-default basket. Basket Default Swaps are essentially default correlation products. Hence, the main aspect for pricing is to model the joint de- fault dependency. The modeling of dependent defaults is difficult because there is very little historical data available about joint defaults and because there are usually no reliable quotes in the market. Therefore, the models cannot be calibrated, neither to defaults nor to prices. Copula methods are emerging as the favored pricing approach. Their appeal is principally due to the simplicity in simulation and to their theoretical framework.

## 1.2 Overview of Basket CDS pricing implementation

We use the following notation and terminology in describing our pricing algorithm:

- $N$ is the number of reference names.
- $N_p$ is the notional principle, which we will normalize to 1.00.
- $T$ is the maturity of the basket default swap and we assume there are $M$ periods, $t_1, \ldots, t_M$ with $t_M = T$.
- $k$ is the seniority of the basket default swap; that is, the number of defaults required to trigger a default payment by the protection seller.
- $\tau_n$ is the time of default of the $n$th reference entity, for $n = 1, \ldots, N$.

- $\tau_{(k)}$ is the time of the k-th default.
- $s$ is the fair spread of the contract, to be paid $\frac{1}{\delta}$ times per annum until $T$ or $\tau k$.
- $Z(t, T)$ is the risk free zero coupon bond price as a discount factor.
- $R$ is the constant recovery rate and $L = 1 - R$.
- $F_n$ is the distribution function of $\tau_n$, for $n = 1, \ldots, N$.
- $F_{(k)}$ is the distribution function of $\tau_{(k)}$.

Our implementation of k-th to default basket credit default swap pricing uses Monte Carlo simulation of a copula model and has the following main steps:

1. For each reference name, bootstrap implied default probabilities from quoted CDS and convert them to hazard rates.
2. Estimate the appropriate inputs for sampling from a copula
   (a) Estimate the covariance matrix for the Gaussian Copula
      i. Transform the price data to returns data.
      ii. Transform the returns data to uniform variates using empirical marginal distributions.
      iii. Estimate the correlation matrix by maximizing the log-likelihood function of the Gaussian copula density
   (b) Estimate the covariance matrix and degrees of freedom of StudentT Copula
      i. Transform the price data to returns data.

ii. Transform the returns data to uniform variates using empirical marginal distributions.

iii. Estimate the correlation matrix using Kendall's tau.

iv. Estimate the degrees of freedom by maximizing the log-likelihood function of the Student's t copula density with correlation matrix over a grid.

3. Find a Cholesky factorization , $A$, of $\Sigma$, the covariance matrix so that $\Sigma = A \cdot A^T$ .

4. For each simulation, repeat the following routine:

   (a) Sample a vector of correlated uniform random variables from the copula:

      i. Sample from a Multivariate Gaussian Copula

         A. Draw an N-dimensional vector $z$ of independent standard normal variates.

         B. Set $x = A \cdot z$.

         C. Transform $\mathbf{x}$ to a vector of uniform variates, $\mathbf{u}$, by setting $u_n = \Phi(x_n)$, for $n = 1, \ldots, N$

      i. Sample from a Multivariate Student's t Copula

         A. Draw an N-dimensional vector $z$ of independent standard normal variates.

         B. Draw an independent $X_\nu^2$ random variate $s$.

         C. Set $\mathbf{y} = \mathbf{A} \cdot \mathbf{z}$.

         D. Set $x = \sqrt{\frac{\nu}{s}}\mathbf{z}$

         E. Transform $\mathbf{x}$ to a vector of uniform variates, $\mathbf{u}$, by setting $u_n = t_\nu(x_n)$, for $n = 1, \ldots, N$.

   (b) Use hazard rates of each reference name to convert the corresponding uniform variable into exact default time.

(c) Sort the default times and choose the k-th one, $\tau_{(k)}$.

(d) The spread of k-th to default basket credit default swap is computed by equating the expected value of the discounted premium leg with the expected value of the discounted default leg under the risk neutral probability measure.

The premium leg is

$$PL = s \times N_p \times \delta \times \sum_{m=1}^{M} Z(0, T_m) \times (1 - F_{(k)}(T_m)) \quad (1.1)$$

Assuming default payment is paid at the end the default period, the default leg is

$$DL = (1-R) \times N_p \times \sum_{m=1}^{M} Z(0, T_m) \times (F_{(k)}(T_m) - F_{(k)}(T_{m-1})) \quad (1.2)$$

It follows that the fair spread is

$$s = \frac{(1-R)\sum_{m=1}^{M} Z(0, T_m)(F_{(k)}(T_m) - F_{(k)}(T_{m-1}))}{\delta \sum_{m=1}^{M} Z(0, T_m)(1 - F_{(k)}(T_m))} \quad (1.3)$$

We may adjust the default leg by substracting the accrued premium, which, if we assume default can only occur at the middle of each period, is

$$AP = \frac{1}{2} \times s \times N_p \times \delta \times \sum_{m=1}^{M} Z(0, T_m) \times (F_{(k)}(T_m) - F_{(k)}(T_{m-1})) \quad (1.4)$$

5. Calculate the average the spread.

# 2 Copulas

In this chapter we will define the Gaussian and Student's $t$ copulas and discuss the methods for calibrating them.

## 2.1 Elementary Properties

**Definition 1.** *A $d$-dimensional copula $C : [0,1]^d \to [0,1]$ is a function which is a cumulative distribution function with uniform marginals. In the following the notation $C(\mathbf{u}) = C(u_1, \ldots, u_d)$ will always be used for a copula. The condition that $C$ is a distribution function immediately leads to the following properties:*

- *As cdfs are always increasing, $C(u_1, \ldots, u_d)$ is increasing in each component $u_i$.*

- *The marginal in component* $i$ *is obtained by setting* $u_j = 1$ *for all* $j \neq i$ *and as it must be uniformly distributed,*

$$C(1, \ldots, 1, u_i, 1, \ldots, 1) = u_i \tag{2.1}$$

- *Clearly, for* $a_i \leqslant b_i$ *the probability* $\mathbb{P}(u_1 \in [a_1, b_1], \ldots, u_d \in [a_d, b_d])$ *must be nonnegative, which leads to the so-called* rectangle inequality

$$\sum_{i_1=1}^{2} \cdots \sum_{i_d=1}^{2} (-1)^{i_1+\cdots+i_d} C(u_{1,i_1}, \ldots, u_{d,i_d}) \geqslant 0 \tag{2.2}$$

*where* $u_{j,1} = a_j$ *and* $u_{j,2} = b_j$.

On the other side, every function which satisfies these properties is a copula. Furthermore, if we have a d-dimensional copula then $C(u_1, \ldots, u_{d-1})$ is again a copula and so are all k-dimensional marginals with $2 \leqslant k < d$.

**Theorem 1.** *Sklar Consider a* $d$*-dimensional cdf* $F$ *with marginals* $F_1, \ldots, F_d$*. There exists a copula* $C$*, such that*

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) \tag{2.3}$$

*for all* $x_i$ *in* $[-\infty, \infty], i = 1, \ldots, d$*. If* $F_i$ *is continuous for all* $i = 1, \ldots, d$ *then* $C$ *is unique.*

On the other hand, consider a copula $C$ and univariate cdfs $F_1, \ldots, F_d$. Then $F$ is defined on 2.3 is a multivariate cdf with marginals $F_1, \ldots, F_d$.

**Corollary 1.** *Let* $F$ *be a* $d$*-dimensional cdf with continuous marginals* $F_1, \ldots, F_d$ *and define* $C : [0, 1]^d \to [0, 1]$ *by*

$$C(u_1, \ldots, u_d) = F(F^{-1}(u_1), \ldots, F^{-1}(u_d)) \tag{2.4}$$

*Then* $C$ *is a copula and for all* $(x_1, \ldots, x_d) \in \mathbb{R}^d$

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) \tag{2.5}$$

If $F$ has a density function $f$ and the copula is sufficiently differentiabls, then Sklar's theorem allows us to write

$$c(\mathbf{u}) = \frac{\partial^d C(u_1, \ldots, u_d)}{\partial u_1 \cdots \partial u_d} = \frac{f(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d))}{f_1(F_1^{-1}(u_1)), \ldots, f_d(F_d^{-1}(u_d))} \tag{2.6}$$

## 2.2 Gaussian Copula

To define a Gaussian copula, we need a symmetric, positive definite matrix, $\Sigma$, with unit diagonal entries. Then the *multivariate Gaussian copula* with covariance matrix $\Sigma$ is defined by

$$C(u_1, \ldots, u_N; \Sigma) = \mathcal{N}(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_N); 0, \Sigma) \tag{2.7}$$

where $\mathcal{N}(x_1, \ldots, x_N; 0, \Sigma)$ denotes the multivariate Normal cumulative distribution for mean 0 and covariance matrix $\Sigma$ and $\Phi$ is the standard normal cumulative distribution. The density for this copula is obtained from 2.6:

$$c(\Phi(t_1), \ldots, \Phi(t_N); \Sigma) = \frac{\frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}} e^{-\frac{1}{2}t^T \Sigma^{-1} t}}{\prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t_n^2}} \tag{2.8}$$

By setting $\zeta = (\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_N))^T$, we see that

$$c(u_1, \ldots, u_N; \Sigma) = \frac{1}{\sqrt{\Sigma}} e^{-\frac{1}{2}\zeta^T(\Sigma^{-1} - I)\zeta} \tag{2.9}$$

### 2.2.1 CALIBRATING GAUSSIAN COPULA

Calibrating a Gaussian copula to market data is a simple application of maximum likelihood estimation. Suppose $X_t = (X_1^{(t)}, \ldots, X_N^{(t)})$ for $t = 1, \ldots, T+1$ is a time series of adjusted price data for the reference entities of a basket credit default swap contract. We first transform the price data to returns data by setting

$$Y_n^{(t)} = \log \frac{X_n^{(t+1)}}{X_n^{(t)}} \tag{2.10}$$

for $n = 1, \ldots, N$ and $t = 1, \ldots, T$. Using 2.9, for $\Sigma$ a symmetric positive definite NxN matrix we obtain the log-likelihood of the returns data as

$$l(\Sigma) = -\frac{T}{2} \ln|\Sigma| - \frac{1}{2} \sum_{t=1}^{T} \zeta_t^T (\Sigma^{-1} - I) \zeta_t \tag{2.11}$$

As is well known, this function is maximized by taking $\Sigma = \hat{\Sigma}$, where

$$\hat{\Sigma} = \frac{1}{T} \sum_{t=1}^{T} \zeta_t \zeta_t^T \tag{2.12}$$

## 2.3 STUDENT'S T COPULA

The *multivariate Student's t copula* with covariance matrix $\Sigma$ and $\nu$ degrees of freedom is defined by

$$C(u_1, \ldots, u_N; \Sigma, \nu) = \mathcal{T}(t_\nu^{-1}(u_1), \ldots, t_\nu^{-1}(u_N); 0, \Sigma, \nu) \tag{2.13}$$

where $\mathcal{T}(x_1, \ldots, x_N; \Sigma, \nu)$ denotes the Student's t cumulative distribution for covariance matrix $\Sigma$ and $\nu$ degrees of freedom and $t_\nu$ is the cumulative distribution for a univariate Student's t with $\nu$

degrees of freedom. The density for this copula is obtained from 2.6:

$$c(u_1,\dots,u_N;\Sigma,\nu) = \frac{1}{\sqrt{\Sigma}}\frac{\Gamma(\frac{\nu+N}{2})}{\Gamma(\frac{\nu}{2})}\left(\frac{\Gamma(\frac{\nu}{2})}{\Gamma(\frac{\nu+1}{2})}\right)^N \frac{(1+\frac{\zeta^T\Sigma^{-1}\zeta}{\nu})^{-\frac{\nu+N}{2}}}{\prod_{n=1}^N(1+\frac{\zeta_n^2}{\nu})^{-\frac{\nu+1}{2}}} \tag{2.14}$$

where $\zeta = (t_\nu^{-1}(u_1),\dots,t_\nu^{-1}(u_N))^T$.

### 2.3.1 Calibrating Student's t Copula

Calibrating a Student's t copula is considerably more difficult because it requires simultaneaous optimization of the correlation matrix and the degrees of freedom. There are a number of possible approaches, including maximum likelihood inference functions for margins and canonical maximum likelihood (CML). The CML method approximates the marginals using the corresponding empirical distributions to avoid simultaneaous optimization. Besides being more tractable, CML makes no assumptions on the distributional form of the marginals. For this reason we restrict our attention to the CML method. The empirical marginal distributions, denoted by $\hat{F}_n$, are determined by the returns data using the formula

$$\hat{F}_n(y) = \frac{1}{T}\sum_{t=1}^{T} 1_{\{Y_n^t \leqslant y\}} \tag{2.15}$$

for $n = 1,\dots,N$. Using the empirical marginal distributions, we can transform the returns data into uniform variates by setting

$$U_t = (U_1^{(t)},\dots,U_N^{(t)}) = (\hat{F}_1(Y_1^{(t)}),\dots,\hat{F}_N(Y_N^{(t)})) \tag{2.16}$$

fot $t = 1,\dots,T$ To further simplify the optimization, we follow [6] recommendation of estimating the correlation matrix of the Student's t copula with Kendall's tau.

Given a time series of adjusted price data $X_t = (X_1^{(t)}, \ldots, X_N^{(t)}$ for $t = 1, \ldots, T+1$, our calibration procedure is made up of the following steps:

1. Transform the price data to returns data, $\{Y_t\}_t$, using 2.10;
2. Transform the returns data to uniform variates, $\{U_t\}_t$, using 2.16;
3. Estimate the correlation matrix, $\hat{\Sigma}$, using $\hat{\Sigma}_{nm} = \sin(\frac{\pi}{2}\tau(U_n, U_m))$;
4. Estimate the degrees of freedom, $\nu$, by maximizing the log-likelihood function of the Student's t copula density 2.14 with correlation matrix $\hat{\Sigma}$ over a grid.

# 3

# Building a Hazard Rate Term Structure

A key component in our approach to pricing basket credit default swaps is estimating the instantaneous default probabilities of each reference entity in the underlying portfolio. In this section we describe a method of construct- ing a term structure of instantaneous default probabilities by bootstraping from credit default swap (CDS) spread data.

In brief, a CDS is used to transfer the credit risk of a reference entity from one party to another. In a standard CDS contract one party purchases credit protection from another party, to cover the loss of the face value of an asset following a credit event. To pay for this protection, the protection buyer makes a regular stream of

payments, known as the premium leg, to the protection seller. The size of these premium payments is calculated from a quoted default swap spread which is paid on the face value of the protection. These payments are made until a credit event occurs or until maturity, whichever occurs first. If a credit event does occur before the maturity date of the contract, there is a payment by the protection seller, known as the protection leg. This payment equals the difference between par and the price of the cheapest to deliver (CTD) asset of the reference entity on the face value of the protection and compensates the protection buyer for the loss. It can be made in cash or physically settled format.

## 3.1 Pricing a CDS

To price a CDS we use a simplified method known as the JP Morgan Approach. In this approach there are $N$ periods, indexed by $n = 1, \ldots, N$ and each period is of length $\Delta t$, expressed in years. The end of period maturi- ties are $T_n = n\Delta t$. The risk-free forward interest rate that can be locked in at time $0$ for investing over the period from $T_{n-1}$ to $T_n$ is denoted by $r_n = r(T_{n-1}, T_n)$. The discount factors can then be written as functions othe forward rates:

$$D_n = D(0, T_n) = exp(-\sum_{k=1}^{n} r_k \Delta t)$$

For a given obligor, we assume that the hazard rate is constant over forward period $n$ with value $\lambda_n$. The survival probability of the obligor at the end of period maturities is then given by

$$P_n = \mathbb{P}(\tau > T_n) = e^{-\sum_{k=1}^{n} \lambda_k \Delta t}$$

We denote the N-period CDS spread as $S_N$, stated as an annualized

percentage of the nominal value of the contract, which, without loss of gen- erality, we can take to be 1.00. We assume that defaults occur only at the end of the period so the premiums will be paid until the end of the pe- riod. Since the premium payments are made as long as the reference entity survives, the expected present value of the premium leg ($PL_N$) is

$$PL_n = S_N \sum_{n=1}^{N} D(0, T_n) P_n \Delta t$$

The expected loss payment in period n is based on the probability of default in period n, conditional on no default in a prior period. This is given by the probability of surviving until period $n-1$ and then defaulting in period n. It follows that the expected present value of the default leg ($DL_N$) is

$$DL_N = (1 - R) \sum_{n=1}^{N} D(0, T_n)(P_{n-1} - P_n)$$

The fair pricing of the N-period CDS, i.e. the fair quote of the spread $S_N$, must be such that the expected present value of payments made by buyer and seller are equal, i.e. $PL_N = DL_N$. From the above discussion, we see that

$$S_N = \frac{(1 - R) \sum_{n=1}^{N} D(0, T_n)(P_{n-1} - P_n)}{\sum_{n=1}^{N} D(0, T_n) P_n \Delta t} \tag{3.1}$$

## 3.2 Bootstrapping Hazard Rates

If we know the discount factors $D_n$ and credit default swap fair spreads for the maturities $T_1, \ldots, T_N$, we can use 3.1 to b̈ootstrapẗhe

hazard rate values. Precisely,

$$P_1 = \frac{L}{L + S_1 \Delta t} \tag{3.2}$$

where $L = 1 - R$; and

$$P_{n+1} = \frac{LX_n - \Delta t S_{n+1} Y_n}{d_{n+1}(L + \Delta t S_{n+1})} + \frac{P_n L}{L + \Delta t S_{n+1}}$$

where

$$X_n = \sum_{k=1}^{n} D_k[P_{k-1} - P_k] \text{ and } Y_n = \sum_{k=1}^{n} D_k P_k$$

Using the procedure just described we obtain survival probabilities at the maturity times. Now we would like to find a corresponding stepwise hazard function; that is, we would like values $\lambda_1, \ldots, \lambda_m$ so that $P(\tau > T_n) = e^{\sum_{i=1}^{n} \lambda_i (T_i - T_{i-1})}$ (where we set $T_0 = 0$) for $n = 1, \ldots, m$. A procedure for finding these values is given below. In that code, T is an array of maturity times and P is an array of survival probabilities at the maturity times. The algorithm below returns an array of constants defining the hazard function.

**Algorithm 1** HazardRates

```
1:  procedure HAZARDRATES(T, P)
2:      t ← 0
3:      p ← 1
4:      for i ← 1, m do
5:          δ ← T_i − t
6:          λ_i ← T_i − t
7:          p ← P_i
8:          t ← T_i
9:      end for
10:     return λ
11: end procedure
```

# 4

# Sensitivity Analysis

## 4.1 Sensitivity Analysis

The key factors on which the price of a kth to default credit default swap depends are

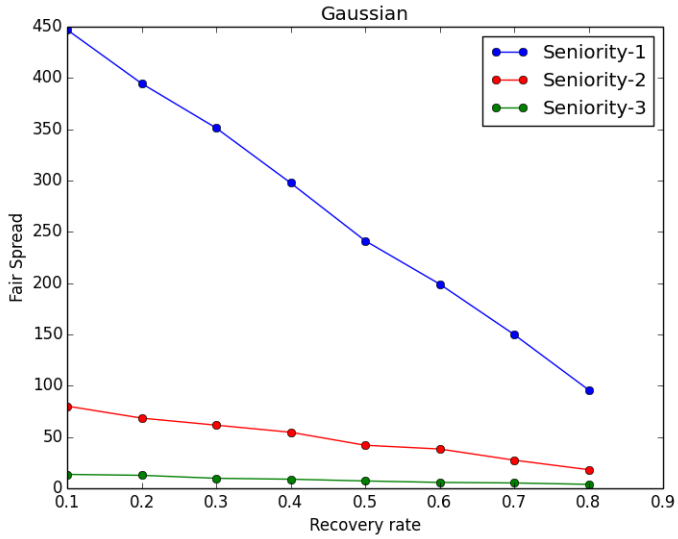
- the recovery rate
- the credit quality of the reference entities
- the default correlation among the reference entities
- the discount factors

In this section we investigate the influence of each of these factors on the simulated fair spread of a basket credit default swap. To simplify this sensitivity analysis, we will assume that the short rate

is constant, the underlying credit default swap curve is flat and the same for all reference entities, and that correlation among the reference entities is pairwise constant.

### 4.1.1 Recovery Rate

Higher recovery rates decrease the fair spread of a basket credit default swap. This is consistent across seniority. Clearly higher recovery rates decrease the expected loss so these results are easily explained. Here is the graph of fairspread vs recovery rate and the python code for generating the graph




```
from bcd.entity import ENTITIES, Entity
from bcd.calibration import GaussianCalibrator
from bcd.discounter import SimpleDiscounter
from bcd.bcdpricer import BCDPricer
from numpy import array, arange
from math import ceil
```

```
from datetime import datetime
import matplotlib.pyplot as plt

def getNoOfPeriods(delta, maturityDate, effectiveDate):
     return int(ceil((maturityDate - effectiveDate).days / (365 * delta)))

entities = ["AAL.L", "AGK.L", "ATST.L", "BA.L", "BAY.L"]
delta = 0.5
includeAccruedPremium = True
seniorities = [1, 2, 3]
recoveryRates = arange(0.1, 0.9, 0.1)
noOfSimulations = 10000
mdate = datetime.strptime("20/09/2015", "%d/%m/%Y")
edate = datetime.strptime("20/06/2010", "%d/%m/%Y")
discounter = SimpleDiscounter([0.5, 1.0, 2.0, 3.0, 4.0, 5.0], \
[0.9932727, 0.9858018, 0.9627129, 0.9285788, 0.8891939, 0.8474275])
noOfNames = len(entities)
noOfPeriods = getNoOfPeriods(delta, mdate, edate)

marginals = []
ent_objs = []
for t in entities:
    e = Entity(ENTITIES[t],t)
    e.calibrate()
    ent_objs.append(e)
    marginals.append(e.survivalDistribution)

calib = GaussianCalibrator()
pd = array([e.priceData() for e in ent_objs])
cop = calib.calibrate(pd)

fsas = []
for k in seniorities:
```

```
    fss = []
    for rr in recoveryRates:
        pricer = BCDPricer(noOfNames, k, delta, noOfPeriods, \
        rr, discounter, cop, marginals)
        price, sims = pricer.price(noOfSimulations, includeAccruedPremium)
        fss.append(price*10000)
    fsas.append(fss)

plt.plot(recoveryRates, fsas[0], 'bo-', label="Seniority-1")
plt.plot(recoveryRates, fsas[1], 'ro-', label="Seniority-2")
plt.plot(recoveryRates, fsas[2], 'go-', label="Seniority-3")
plt.title("Gaussian")
plt.xlabel("Recovery rate")
plt.ylabel("Fair Spread")
plt.legend()
plt.show()
```

### 4.1.2 Credit Quality

In addition to the general simplifying assumptions mentioned above, if we also assume a constant recovery rate, a fixed pairwise default correlation among reference entities and a constant spot interest rate then it can be shown that the basket credit default premiums increase with increasing premiums for the underlying CDS curve.

### 4.1.3 Default Correlation

The premium for 1st to default basket credit default swap decreases as the pairwise default correlation increases. For 2nd or 3rd to default basket credit default swaps however, premiums appear to increase as the pairwise default correlation increases though the

effect is quite small.

### 4.1.4 Discount Factors

The influence of discount factors can be explored by considering constant shifts in the yield curve, i.e by letting the short rate vary. Assuming a constant recovery rate,a fixed pairwise default correlation among reference entities, it can be shown that the fair spread value of a basket credit default swap does not vary much with the discount factors.

# PartII

# Implementation of HJM Model by Monte Carlo Simulation

# 1

# The Heath-Jarrow-Morton Framework

The Heath-Jarrow-Morton (HJM) framework is a general framework within which specific no-arbitrage models can be implemented. In a HJM model the whole yield curve is modelled in terms of forward rates because the no-arbitrage condition has a particularly simple form in this approach and the initial forward-rate curve is a part of the input.

## 1.1 Overview of the HJM implementation

1. Relate the bond prices to the instantaneous forward rates.
2. Model the bond prices using a single or multi factor Stochastic

Differential Equation(SDE).

3. Obtain a model(SDE) for the forward rate curve using the bond price model and the relation between bond prices and forward rates.
4. Derive the risk neutral drift in terms of volatilities using no arbitrage conditions.
5. Use Musiela parametrization to simplify the volatility structure used for implementation.
6. Discretize the forward rate SDE.
7. Perform PCA to calibrate the discrete volatility using historic forward rate time series data.
8. Perform Monte Carlo simulation to calcuate prices of zero coupon bonds and interest rate derivatives like caps, floors etc.

## 1.2 Forward Rate Equation

The key concept of the HJM Model is that we model the evolution of the whole forward rate curve, not just the short end. Let $Z(t,T)$ denote the price at time $t$ of a zero-coupon bond maturing at time $T$ and having face value 1. Recall that the forward rate $f(t,T)$ represents the instantaneous continuously compounded rate contracted at time $t$ for riskless borrowing or lending at time $T > t$.

If $f(t,T)$ is known for all values $0 \leqslant t \leqslant T$, then

$$Z(t,T) = e^{-\int_t^T f(t,\nu)d\nu}, 0 \leqslant t \leqslant T \tag{1.1}$$

From the above we also have

$$f(t,T) = -\frac{\partial}{\partial T} \log Z(t,T). \tag{1.2}$$

### 1.2.1 Risk-Free rate of Interest and the Short Rate

The instantaneous risk-free rate of interest is related to the instantaneous forward rate of interest in the following manner

$$r(t) = f(t,t) \tag{1.3}$$

$r(t)$ can be interpreted as the rate of interest on a bank account:this can be changed on a daily basis by the bank with no control on the part of the investor or bank account holder. $r(t)$ is referred to as the short rate.

## 1.3 One factor HJM model

We take the initial forward-rate curve $f(0,T)$ as our starting point.For a fixed maturity, $T$, $f(t,T)$ is an Itô process satisfying

$$df(t,T) = \mu(t,T)dt + \sigma(t,T)dW(t) \tag{1.4}$$

for each $T > t$, where $\mu(t,T)$ and $\sigma(t,T)$ may depend upon $f(t,T)$, or the whole forward-rate curve at time $t$, or, even more generally, upon $\mathcal{F}_t = \sigma(W(s) : s \leqslant t)$.

In a one-factor model, we define, for all maturities $T$, Itô processes that are dependent upon the same one-dimensional source of uncertainty $W(t)$. It follows that changes over the whole of the forward-rate curve are perfectly but non-linearly correlated.

### 1.3.1 Drift under no arbitrage conditions

Under no-arbitrage conditions, the drift and volatility are related as follows

$$\mu(t,T) = \sigma(t,T)\int_t^T \sigma(t,\nu)d\nu \tag{1.5}$$

# 2

# Implementation of the HJM model

## 2.1 Calibrating the HJM Model using PCA

The diffusion process is the same in the dynamics of forward rates under the real world measure as it is in the dynamics under the risk-free measure, we can use historical data to estimate the diffusion, $\sigma(t, T)$, in the HJM model.

Assume that $\sigma(t, T)$ is of the form

$$\sigma(t, T) = \tilde{\sigma}(T - t) \tag{2.1}$$

for some deterministic function $\tilde{\sigma}(\tau)$. We will choose $\tilde{\sigma}(\tau)$ to

match historical data. The forward rate evolves according to the SDE

$$df(t,T) = \mu(t,T)dt + \tilde{\sigma}(T-t)dW(t); 0 \leqslant t \leqslant T \tag{2.2}$$

Suppose we have observed this forward rate at times $t_1 < t_2 < \cdots < t_J < 0$ in the past and the forward rate we observed at those times was for the relative maturities $\tau_1 < \tau_2 < \cdots < \tau_K$; that is, we have observed $f(t_j, tj + \tau_k)$ for $j = 1, \ldots, J$ and $k = 1, \ldots, K$. Suppose further that for some small positive $\delta$ we have also observed $f(tj + \delta; tj + \tau_k)$, where $\delta$ is sufficiently small that $tj + \delta < t_{j+1}$ for $j = 1, \ldots, J-1$ and $t_{J+\delta} < 0$. According to our model,

$$f(t_j + \delta, t_j + \tau_k) - f(t_j, t_j + \tau_k) \approx \delta\mu(t_j, t_j + \tau_k) + \tilde{\sigma}(\tau_k)(W(t_j + \delta) - W(t_j))$$

Let

$$D_{j,k} \approx \sqrt{\delta}\mu(t_j, t_j + \tau_k) + \tilde{\sigma}(\tau_k)\frac{W(t_j + \delta) - W(t_j)}{\sqrt{\delta}} \tag{2.3}$$

Since the first term in this equation contains $\sqrt{\delta}$, it is small relative to the second term. So, for

$$X_j = \frac{W(t_j + \delta) - W(t_j)}{\sqrt{\delta}}, j = 1, \ldots, J \tag{2.4}$$

which is a standard normal random variable.we have

$$D_{j,k} = \tilde{\sigma}(\tau_k)X_j \tag{2.5}$$

But the $X_i, \ldots, X_J$ are not only standard normal random variables, they are also independent. This means we can regard $D_{1,k}, \ldots, D_{J,k}$ as independent observations taken at times $t_1, \ldots, t_J$ on forward rates, all with the same relative maturity $\tau_k$. The empirical

covariance is

$$C_{k_1,k_2} = \frac{1}{J}\sum_{j=1}^{J} D_{j,k_1} D_{j,k_2}$$

The theoretical covariance, computed from the right-hand side of 2.5, is

$$\mathbb{E}[\tilde{\sigma}(\tau_{k_2})X_j^2] = \tilde{\sigma}(\tau_{k_1})\tilde{\sigma}(\tau_{k_2})$$

We would like to find $\tilde{\sigma}(\tau_1), \ldots, \tilde{\sigma}(\tau_K)$ so that

$$C_{k_1,k_2} = \tilde{\sigma}(\tau_{k_1})\tilde{\sigma}(\tau_{k_2}), k_1, k_2 = 1, 2, \ldots, K \tag{2.6}$$

To determine a best choice of $\tilde{\sigma}(\tau_1), \ldots, \tilde{\sigma}(\tau_K)$, we use principal component analysis.Let

$$C = \lambda_1 e_1 e_1^T + \lambda_1 e_2 e_2^T + \cdots + \lambda_K e_K e_K^T,$$

where $\lambda_1 \geqslant \lambda_2 \cdots \lambda_K \geqslant 0$ are the eigenvalues of C and the column vectors $e_1, e_2, \ldots, e_K$ are the orthonormal eigenvectors. As noted above, we would like to find $\tilde{\sigma}(\tau_1), \ldots, \tilde{\sigma}(\tau_K)$ so that

$$C = \begin{bmatrix} \tilde{\sigma}(\tau_1) \\ \tilde{\sigma}(\tau_2) \\ \vdots \\ \tilde{\sigma}(\tau_K) \end{bmatrix} \begin{bmatrix} \tilde{\sigma}(\tau_1) & \tilde{\sigma}(\tau_2) & \ldots \tilde{\sigma}(\tau_K) \end{bmatrix}$$

but this may not be possible to do exactly. The best approximation is

$$\begin{bmatrix} \tilde{\sigma}(\tau_1) \\ \tilde{\sigma}(\tau_2) \\ \vdots \\ \tilde{\sigma}(\tau_K) \end{bmatrix} = \sqrt{\lambda_1} e_1$$

## 2.2 THE PRICING ALGORITHM

The implementation of the HJM model uses Monte Carlo to simulate . However, except for very simple cases of $\sigma$, exact simulation of is not feasible and one must resort to discrete approximation. We will follow [1] Section 3.6.2 and 3.6.3) quite closely.

Discrete approximation of forward rates will require discretization of both arguments of $f(t, T)$.To simplify notation we will fix a time grid $0 = t_o < t_1 < t_i < t_{i+1} < \cdots < t_M$ for the first argument and then for each fixed time $t_i$ we will use the subgrid $t_i < t_{i+1} < \cdots < t_M$ for the second argument. At each step, we need only the vector of current rates. $t_M = T^*$ is our time horizon, so $t_M$ is the maturity of the longest-maturity bond represented in the model. This means that the last forward rate relevant to the model applies to the interval $[t_{M-1}, t_M]$, the forward rate with maturity argument $t_{M-1}$. We use $\hat{f}(t_i, t_{i+1})$ to denote the discretized forward rate for maturity $t_j$ as of time $t_i$, $j \geqslant i$. Thus, our initial vector of forward rates consists of the $M$ components $\hat{f}(0;0), \hat{f}(0, t_1), \ldots, \hat{f}(0, t_{M-1})$.

### 2.2.1 INITIAL FORWARD RATE CURVE

We use $\hat{Z}(t_i, t_j)$ to denote the discretized zero-coupon bond price so that

$$\hat{Z}(t_i, t_j) = exp\left(-\sum_{l=i}^{j-1} \hat{f}(t_i, t_l)(t_{l+1} - t_l)\right) \tag{2.7}$$

To minimize discretization error, we would like the initial values of the discretized zero-coupon bonds $\hat{Z}(0, t_j)$ to match the exact values $Z(0, t_j)$ for all maturities $t_j$ on the grid. From 1.1 and 2.7 it

follows that this holds if

$$\sum_{l=1}^{j-1} \hat{f}(0,t_l)(t_{l+1}-t_l) = \int_0^{t_j} f(0,v)dv$$

that is if

$$\hat{f}(0,t_l) = \frac{1}{t_{l+1}-t_l}\int_{t_l}^{t_{l+1}} f(0,v)dv, \tag{2.8}$$

for all $l = 0, 1, \ldots, M-1$. So we should initialize $\hat{f}(0,t_l)$ to the average level of the forward curve $f(0,T)$ on the interval $[t_l, t_{l+1}]$.

### 2.2.2 USING DISCRETIZED SDE FOR TIME STEPPING

As the simulation evolves, the number of relevant rates decreases: at time $t_i$ we are only interested in the rates $\hat{f}(t_i,t_i), \hat{f}(t_i;t_{i+1}), \ldots, \hat{f}(t_i,t_{M-1})$. We represent these $M-i$ rates remaining at time ti as the vector $(f_1;\ldots;f_{M-i})$. Thus we are indexing forward rates by relative maturity, as done in the Musiela parametrization.

Similar notational conventions will be used for $\hat{\mu}(t_i,t_j)$ and $\hat{\sigma}_k(t_i,t_j), k = 1,\ldots,d$ we use $m_j$ for $\hat{\mu}$ and $s_j(k)$ for $\hat{\sigma}_k$. Thus the simulation step from $t_{i-1}$ to $t_i$ becomes

$$f_j \leftarrow f_{j+1} + m_j[t_i - t_{i-1}] + \sum_{k=1}^{d} s_j(k)\sqrt{t_i - t_{i-1}}Z_{ik}, j = 1,\ldots,M-i,$$

where

$$m_j = \hat{\mu}(t_{i-1}, t_{i+j-1}), s_j(k) = \hat{\sigma}_k(t_{i-1}, t_{i+j-1})$$

Our implementation of the HJM simulation is broken into to main parts: one calculating the discrete drift parameter at a fixed time step and the other looping over time steps and updating the forward curve at each step.

Calculating the discrete drift parameter amounts to evaluating

$$\hat{\mu}(t_{i-1}, t_{i+j-1}) = \frac{1}{2h_j}\left[\sum_{k=1}^{d}\left(\sum_{l=i}^{j}\hat{\sigma}_k(t_{i-1}, t_l)h_{l+1}\right)^2 - \sum_{k=1}^{d}\left(\sum_{l=i}^{j-1}\hat{\sigma}_k(t_{i-1}, t_l)h_{l+1}\right)^2\right]$$

2 achieves this in a way that avoids duplicate computation. In the notation of the algorithm, the drift parameter is evaluated as

$$\frac{1}{2(t_{j+1} - t_j)}[B_{next} - B_{prev}]$$

and each $A_{next}(k)$ records the quantity

$$\sum_{l=i}^{j}\hat{\sigma}_k(t_{i-1}, t_l)h_{l+1}$$

The inputs are the volatilities
$s = \{s_j(k) : j = 1, \ldots, M - i, k = 1, \ldots, d\}$, the intervals
$h = \{h_l : l = 1, \ldots, M\}$ (where $h_l = t_l - t_{l-1}$), and the step index $i$.

3 implements a single replication of the pricing procedure in an HJM simulation, which is repeated many times to estimate a product price. This algorithm calls 2 to calculate the discrete drift for all remaining maturities at each time step. The key step in the algorithm used to value interest rate derivatives is

$$P \Leftarrow \text{cashflow at } t_i$$

where, of course, the particular instrument determining the cashflow at $t_i$ must be provided as input. The remaining inputs are the initial curve $f = \{f_l : l = 1, \ldots, M\}$ and the intervals
$h = \{h_l : l = 1, \ldots, M\}$.

**Algorithm 2** UpdateDrift(s,h,i)

**procedure** UPDATEDRIFT($s, h, i$)
  $A_{prev} = 0, k = 1, \ldots, d$
  **for** $j = 1$ to $M - i$ **do**
    $B_{next} = 0$
    **for** $k = 1$ to $d$ **do**
      $A_{next}(k) = A_{prev}(k) + s_j(k) \times h_{i+j}$
      $B_{next} = B_{next} + A_{next}(k) \times A_{next} \times A_{next}(k)$
      $A_{prev}(k) = A_{next}(k)$
    **end for**
    $m_j = (B_{next} - B_{prev})/(2h_{i+j})$
    $B_{prev} = B_{next}$
  **end for**
  **return** $(m_i, \ldots, m_{M-i})$
**end procedure**

**Algorithm 3** PresentValue

**procedure** PRESENTVALUE($f, h, product$)
  $D = 1, P = 0, C = 0$
  **for** $i = 1$ to $M - 1$ **do**
    $D = D \times e^{-f_1 \times h_i}$
    **for** $j = 1$ to $M - i$ **do**
      **for** $k = 1$ to $d$ **do**
        $s_j(k) = \hat{\sigma}_k(t_{i-1}, t_{i+j-1})$
      **end for**
      $(m_1, \ldots, m_{M-i}) = UpdateDrift(s, h, i)$
    **end for**
    generate $Z_1, \ldots, Z_d \sim \mathcal{N}(0, 1)$
    **for** $j = 1$ to $M - i$ **do**
      $S = 0$
      **for** $k = 1$ to $d$ **do**
        $S = S + s_j(k) \times Z_k$
      **end for**
      $f_j = f_{j+1} + m_j \times h_i + S \times \sqrt{h_i}$
    **end for**
    $P =$ cashflow at $t_i$ (depending on the instrument)
    $C = C + D \times P$
  **end for**
  **return** $(C)$
**end procedure**

# 3

# Numerical Results

In this chapter we describe the results of applying our implementation of the HJM model to pricing some simple fixed-income products e.g. caps.

## 3.1 Zero-Coupon Bonds

Zero coupon bond prices can be computed from the initial forward curve, using the formula:

$$Z(0,t_j) \approx \hat{Z}(0,t_j) = e^{-\sum_{l=0}^{j-1} \hat{f}(0,t_l)(t_{l+1}-t_l)}$$

However, this provides one way to test the implementation. This means that if we use our simulation to price a bond, then the results

should converge to the value to which the model was initially calibrated. The table below compares simulated and analytic prices for several zero-coupon bonds with face value of 1:00.

| Maturity | ZCB Price Analytic | ZCB Price Simulated |
|---|---|---|
| 1 Month | 0.999558430854 | 0.999133691333 |
| 1 Quarter | 0.998692522162 | 0.998234730251 |
| 1 Year | 0.994042814814 | 0.993407538138 |
| 2 year | 0.982783264698 | 0.981436508441 |
| 5 Year | 0.902187117285 | 0.897855729498 |
| 10 Year | 0.703925090987 | 0.689221048009 |

The python code used for obtaining the above values is given below

```
from hjm.hjmpricer import *

principal = 1.0
product = "ZCB"
maturities = [0.084, 0.25, 1.0, 2.0, 5.0, 10.0]
int_rate = 0.4
freq = 0
factors = 3
nsims = 10000

print "Analytic ZCB Price", "|", "Sim ZCB Price"
for m in maturities:
    hjmp = HJMPricer(principal, product, m, int_rate, freq, factors, nsims)
    (simprice, sims), pca = hjmp.getHjmPrice()
    print hjmp.getZCBPrice(), "|", simprice
```

## 3.2 CAPS

A caplet is an interest rate derivative providing protection against an increase in an interest rate for a single period. A *cap* is a portfolio of caplets covering multiple periods. The caplet is like a call option on the short rate, having a payoff of the form $(r(T) - R)^+$ for maturity $T$ and fixed rate $R$. The underlying rate in a caplet applies over and interval and is based on discrete compounding. To simplify the implementation, we will assume the interval has the form $[t_i, t_{i+1}]$. At $t_i$ the continuously compounded rate for this interval is $\hat{f}(t_i, t_i)$; the corresponding discretely compounded rate $\hat{F}$ satisfies

$$\frac{1}{1 + \hat{F}(t_i)[t_{i+1} - t_i]} = e^{-\hat{f}(t_i,t_i)[t_{i+1}-t_i]} \tag{3.1}$$

so that,

$$\hat{F}(t_i) = \frac{1}{t_{i+1} - t_i}\left(e^{-\hat{f}(t_i,t_i)[t_{i+1}-t_i]} - 1\right) \tag{3.2}$$

The payoff of the caplet would then be $\hat{F}(t_i) - R)^+$ (times the principal). The payment is made at the end of the interval, $t_{i+1}$, and so must be discounted back to $t_i$, which leads to

$$P = e^{-f_1 h_{i+1}}\left(\frac{1}{h_{i+1}}(e^{f_1 h_{i+1}} - 1) - R\right)^+ \tag{3.3}$$

Some cap prices computed using our implementation are given below. These all have tenor one quarter and interest rate 0.4% (Number of simulations = 10000).

The python code used for obtaining the above values is given below

```
from hjm.hjmpricer import *


principal = 1.0
```

| Maturity | Simulated Cap Price |
|---|---|
| 1 Quarter | 0.00130178679059 |
| 6 Month | 0.00310862528672 |
| 1 Year | 0.00888973584619 |
| 5 Year | 0.348849093993 |
| 10 Year | 1.18296755777 |

```
product = "CAP"
maturities = [0.25, 0.5, 1.0, 5.0, 10.0]
int_rate = 0.4
freq = 0.25
factors = 3
nsims = 10000

print "Sim CAP Price"
for m in maturities:
    hjmp = HJMPricer(principal, product, m, int_rate, freq, factors, nsims)
    (simprice, sims), pca = hjmp.getHjmPrice()
    print simprice
```

# Appendices

# A

# Software Implementation

## A.1 Implentation Details

The implementation of the BCD and HJM models is done using Python 2.7 making use of numeric libraries like numpy and scipy. To improve the performance of the code, Cython was used.

### A.1.1 Learnings

Python was used as it is a highly expressive and readable language with a very extensive set of libraries. The high expressiveness enables us to write very compact code which reveals the structure of the numerical procedure clearly. It also makes it easy to validate the correctness of the implementation. Unfortunately optimizing the Python code for performance was not as easy as expected. Monte

Carlo simulations can be very efficiently executed in parallel but writing parallel programs in Python is not straightforward. Here are a few alternative ways of implementing the numerical procedures to make them more peformant:

1. Implement the procedures in a language like Julia which is highly performant and provides powerful parallel primitives.
2. Implement the code which needs to be optimized in C/C++ and use Boost Python or SWIG to invoke it from python code

Optimizing Python by implementing certain parts in Cython can be tricky as benefits of generating and executing C code can be offset by the back and forth type conversions between C and Python code.

## A.2 USING THE SOFTWARE

Copy all the contents(a folder named "cqf_vamshi_Jan13") of the CD onto the local machine. All the data required by the application is stored in a directory called "data" in the root folder. All the python packages that are required for the project are part of standard free python distributions like Anaconda or Canopy express. In addition to one of the python distributions mentioned above, a C compiler is required to compile the cython extension modules to C. All the dependencies can be downloaded and the cython extension modules can be compiled by running the following commands .

```
cd cqf_vamshi_Jan13
pip install -r requirements.txt
```

## A.3 RUNNING SAMPLE TEST CASES

Sample test cases can be executed by first modifying the **testhjm.yaml** and **testbcd.yaml** files and then running the coresponding **testhjm.py** and **testbcd.py** files. The yaml files are human readable and editable in a text editor. Instructions provided in the yaml files should be carefully followed and the indentation in the files should not be changed. The commands for executing the test cases are as follows

```
python testhjm.py
python testbcd.py
```

# References


[1] Glasserman, P., Monte Carlo Methods in Financial Engineering, Springer, New York, 2003.

[2] Shreve, Steven E., Stochastic Calculus for Finance II: Continuous-Time Models, Springer, New York, 2004.

[3] Wilmott, Paul, Paul Wilmott on Quantitative Finance, Wiley, New York, 2000.

[4] Nelsen, R.B., An introduction to Copulas, Springer, New York, 2006.

[5] Lindskog F., McNeil, A. and Shmock, U., Kendalls tau for elliptical distributions. Working paper, Risklab ETH Zurich, 2001.

[6] Mashal, R. and Zeevi, A., Beyond Correlation: Extreme Comovements Between Financial Assets. Working paper, Columbia Graduate School of Business, 2002.