CQF: Final Project

Joseph Dwonczyk

April 9, 2018

Part I

CVA Valuation

Chapter 1

Introduction to CVA

title In this section, I calculate the credit valuation adjustment (CVA) taken by a Counterparty A when they enter a vanilla interest rate swap with a Counterparty B in which Counterparty A pays fixed payments and receives floating payments. In this project, I will use a European bank as an example Counterparty B as there is much more data available on the probabilty of default of major European banks.

Chapter 2

Data Collection

For this part of the project I require the following data:

- Discount Curves for discounting simulated future cash flows in the Interest Rate swap:
  - ⋆ LIBOR
  - ⋆ SONIA
- Default Rates for Counterparty B:
  - ⋆ Credit Spreads for Tenors out to 5 years to cover the lifetime of the swap.

To fetch this data, I have used the Bloomberg terminal of a colleague.
For fetching the Discount rates, I fetch the spot rates for LIBOR and SONIA and bootstrap the implied Discount rates as follows. I first grab the discount factors for the observed tenors:

$$DF_i(0,\tau) = \Big(\frac{1}{1+i}\Big)^t$$

I next interpolate the discount factors so that we can look at the exposure of Counterparty A to Counterparty B over smaller windows in the future. We will use months (1/12 years) as the timestep and review later if there is any evidence that more timesteps are required. I use a log-linear interpolation as the discount factor is the inverse of the rate that we are observing in the market. i.e.

$$logDF(0,\tau) = \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i} logDF(0,\tau_{i+1}) + \frac{\tau_{i+1} - \tau}{\tau_{i+1} - \tau_i} logDF(0,\tau_i)$$

For fetching the default probabilities of Counterparty B, I pull the Credit Default Swap Spreads for the Tenors of 1 out to 5 years inclusive. I then make the assumption that the market prices of these values are sufficiently liquid to infer the Markets view on **the price of the probability of default of Counterparty B**. I then use these values as a proxy for the probability of default of Counterparty B until the respective tenor. For example, the 5 year CDS can be used to derive a proxy for the probability that Counterparty B defaults between now and 5 years time. I then again interpolate the default probabilities to monthly periods using a linear interpolation for the same reason as above.

Chapter 3

Forward-Rate Simulation in the Heath-Jarrow-Morton Family

I this project, I have decided to model the forward rates in the HJM family of models as follows:

This can be seen by starting with the relationship between the price of a Bond at time t today, $Z(t,T)$ and the instantaneous forward rate $f(t,T)$ out to time T as seed from time t:

$$Z(t,T) = e - \int_t^T f(t,s)ds = -\frac{\partial}{\partial T} logZ(t,T)$$

Under the HJM framwork we then use the Bond Pricing Equation to give us:

$$df(t,T) = \frac{\partial}{\partial T}\Big[\frac{1}{2}\sigma^2(t,T) - \mu(t,T)\Big]dt - \frac{\partial}{\partial T}\sigma(t,T)dW_t \tag{3.1}$$

where μ defines the drift function for the bond price and σ the volatility of the bond price. We next consider the BPE under the risk neutral measure which would give us the same as above but with $\mu^{\mathbf{Q}}(t,T) = r(t)$ and $dW_t^{\mathbf{Q}}$. I notice that as $r(t)$ is not T-dependent, the partial derivative wrt T is by definition 0. We apply the change of measure under the result of Girsanov's thm.
We are then able to setup a Hedged portfolio using a bond of different maturity. In doing so we discover that we are able to separate the Hedged portfolio into 2 different parts; the first which depends on the first bonds maturity and the second likewise. Which tells us at the Market price of risk, $\lambda(t)$, must be independent of maturity. Thus after some simplification, we have:

$$df(t,T) = \sigma(t,T)\frac{\partial}{\partial T}\sigma(t,T)dt - \frac{\partial}{\partial T}\sigma(t,T)dW_t^{\mathbf{Q}} \tag{3.2}$$

With this in mind, I fetch historical Libor spot rates for the last 3 years so that there are sufficient data points. I fetch this data for each tenor: 1 month, 6 months, 12 months out to 50 years. I then bootstap the forward rates from the spot rates for each historical date. From the historical forward rates, I can now conduct a Principal Component Analysis on the different rates out to 50 years to conclude which rates are historically the most significant. I will then simulate these significant rates in order to simulate the outcome of the vanilla interest rate swap. The Monte carlo simulation would be too slow if we chose to evolve all of the rates as it would become a very high dimensional problem with each forward rate adding a new dimension.
To conduct the principal component analysis of the forward rates, I first calculate the matrix of absolute differences (day-by-day) in forward rates in order to then calculate the covariance matrix for the forward rates. I then calculate the eigenvalues and respective eigenvalues for the covariance matrix as below:

	0	1	2	3	4	5
0	1.0	2.5479834212e-06	3.26647738456e-06	2.29331392091e-05	-3.60862585462e-06	-1.3
1	2.5479834212e-06	1.0	-5.44517950652e-08	-5.91168281189e-05	2.26947689793e-05	2.83
2	3.26647738456e-06	-5.44517950652e-08	1.0	0.000171578302069	-3.25578910033e-05	-3.2
3	2.29331392091e-05	-5.91168281189e-05	0.000171578302069	1.0	-0.000873518055955	-0.0
4	-3.60862585462e-06	2.26947689793e-05	-3.25578910033e-05	-0.000873518055955	1.0	0.00
5	-1.32026977123e-06	2.83177431241e-05	-3.24196198103e-05	-0.000945510801984	0.000246162870129	1.0
6	1.00818266744e-05	4.97124617692e-05	-2.45824600446e-05	-0.00103519482214	0.000356696623934	0.00
7	2.2148730266e-05	-2.82866240003e-05	0.000219743415952	0.00235641517881	-0.000625882887944	-0.0
8	1.19383372506e-05	4.61319589783e-05	5.03510705643e-05	-0.000651597802177	0.000391703865431	0.00
9	2.52316935023e-05	4.29989571905e-05	8.66693856812e-05	7.97753631638e-05	8.60652306779e-05	0.00
10	1.05765572324e-05	2.35182991259e-05	7.11575552309e-05	2.81091444219e-05	8.54576009618e-05	0.00
11	1.01385544238e-05	1.8492492678e-05	7.06599913062e-05	4.29719738392e-05	8.96471920099e-05	8.69
12	7.18211921494e-06	1.47635282991e-05	6.18264553988e-05	6.51366373902e-07	7.5455438304e-05	8.99
13	3.18094384313e-06	1.12097899388e-05	5.37290897304e-05	-9.3569807032e-05	2.97292693313e-05	0.00
14	4.29203016265e-06	6.24840158264e-06	4.10532780364e-05	-2.3762719505e-05	3.53115398284e-05	4.75
15	6.46842241308e-06	1.12710111731e-05	5.72466167378e-05	2.64717658404e-05	4.96030842502e-05	7.06
16	4.75281485517e-06	9.70351293336e-06	5.91613501244e-05	3.08850143083e-07	5.22401026608e-05	7.41
17	2.01341647183e-05	2.86751897321e-05	6.76206239852e-05	2.91627788768e-05	6.33553978083e-05	0.00
18	9.49095865681e-06	1.76255734581e-05	8.2907630706e-05	2.98240353677e-05	7.22569418738e-05	0.00
19	3.74920468535e-07	3.37183040588e-06	4.75075935483e-05	-1.44767380782e-05	4.78854110934e-05	6.65
20	-9.74278493378e-06	-1.00756519112e-05	3.79972528656e-05	-4.73653426559e-05	3.87478508083e-05	4.97
21	-7.97986981484e-07	8.45714937594e-07	3.26054765205e-05	-3.2473694431e-06	2.72643417549e-05	4.14
22	-1.45246688963e-05	-1.45166579959e-05	1.79831374843e-05	-7.21096279405e-05	3.75032887559e-05	2.09
23	7.52077449756e-06	1.85297948185e-05	4.04583332064e-05	-4.39152919981e-06	6.00574923196e-05	8.35
24	9.16448315967e-06	1.27739276181e-05	3.29017680324e-05	3.2967333407e-06	4.56628924553e-05	4.50
25	6.34434048042e-06	1.03143285393e-05	3.2287019978e-05	-1.0304934076e-05	4.12952212691e-05	5.11

3.1 Eigen-Values

	0	1	2	3	4	5	6
Eigen values	1.00756256558	1.00545869247	1.00374441381	1.00178751973	1.0006608268	1.00033096681	1

3.2 Eigen-vectors

	0	1	2	3	4	5
0	-0.00531884181795	-0.00293574987334	0.00772743976615	0.00924002346893	0.0049586285041	-0.0206
1	-0.0113099981815	-0.00386526369613	-0.0210052728466	0.0189827928331	0.000577319079393	-0.0393
2	-0.0229776032602	-0.0287726367495	0.0687886905288	-0.0208278341729	-0.0946648919415	-0.1346
3	0.0288686707545	-0.0258830431473	0.608596900875	-0.0719457459258	-0.0211774345217	-0.0056
4	-0.0498471523378	-0.00359445651822	-0.292863576596	0.0316851634439	-0.0446904593942	-0.0664
5	-0.0525070835336	-0.0208195367295	-0.31612903875	0.0592707030281	-0.0388549794042	-0.0851
6	-0.240290758075	-0.00372587386789	-0.289096883055	0.186837727662	0.0689097502534	-0.0459
7	-0.154601661729	-0.00608836690884	0.569306766613	-0.00627378074096	-0.0177832187791	-0.0817
8	-0.533205909577	0.266469320033	-0.0801093955313	-0.206371781104	0.00817851176261	0.0100
9	-0.422074949057	-0.0154042868625	0.080428674857	0.259243688597	-0.0533729361132	0.0174
10	-0.318902578204	-0.0581448755073	0.0441665471977	0.129099835096	-0.216549385336	0.1418
11	-0.35275591155	0.0158328719008	0.0146575874586	-0.145787032985	-0.175767991229	0.1394
12	-0.18618569801	-0.181663690213	-0.0041869715873	-0.00435259478668	-0.274464336464	0.1803
13	0.177765878759	-0.412385499447	-0.0112003310842	0.266058891128	-0.0529463691326	-0.0328
14	0.138240365746	-0.398799875232	-0.00697585558785	0.289030647198	-0.255634703155	0.1617
15	-0.119584866632	-0.292165008204	-0.0041401757372	-0.0672450600463	-0.135348181171	0.1268
16	-0.105365128045	-0.292588361832	-0.0218458658275	-0.174701037989	-0.019944681336	0.0284
17	-0.210469525282	-0.358049968647	0.00807609454818	0.0194408294408	0.317042076438	-0.0553
18	-0.182060377986	-0.384858110208	0.00872890987678	-0.0743299079339	0.240075999256	-0.1104
19	-0.0390731823116	-0.242489326322	-0.0411446800716	-0.429196794059	0.340202516117	-0.2704
20	0.0528929602622	-0.169863876968	-0.066255982726	-0.415212279798	-0.131158251883	-0.1333
21	-0.00358305715477	-0.130544657367	-0.0363208471398	-0.199782034029	-0.314342137846	0.1639
22	0.129759225772	0.0411646466894	-0.0605259627949	-0.349051659832	-0.466650631706	-0.1104
23	-0.102079246582	0.0640113168913	0.0628879801828	0.292693387253	-0.0889654412286	-0.5101
24	-0.0508117029584	-0.0530453880121	-0.00289382966621	0.0847753785137	-0.142124559008	-0.4551
25	-0.0246845656428	-0.0203811886661	-0.00867655746509	0.0159636604847	-0.321417769269	-0.4796

From this, I take the three most significant rates which have the largest eigen values. I only choose the most significant 3 as this adds 3 dimensions to the simulation and accounts for the short and long ends of the rates curve.
For the Monte Carlo simulation in the next section, for each iteration I will project the obeserved forward rates of the 3 most significant forward rates out to the expiry of the 5 year interest rate swap. I will project the forward rates using a time step of 1 day using equation 3.1 above **under the risk neutral measure for pricing**. For this , I require the risk free drift and volatility of the bond which we have from equation 3.2 where:

$$\sigma(t,T)\frac{\partial}{\partial T}\sigma(t,T)dt = \frac{\partial}{\partial T}\Big[\frac{1}{2}\sigma^2(t,T) - r(t)\Big]dt$$

So I fit the volatility function for each rate from the observed historical data using the output of the principal component analysis. Define e_i to be the eigen value for the i^{th} rate and likewise v_i the i^{th} vector. Then define:

$$\hat{\sigma}_i(t,T) = \sqrt{e_i}\dot{v}_i$$

I then provide a function that interpolates the volatility vector above, $\hat{\sigma}_i(t,T)$ using a cubic spline to have a value for each required tenor when we come to carry out the simulation.

Chapter 4

Monte Carlo Simulation of Interest Rate Swap

Now that the forward rate methodology is setup, all that is left to do is to run the Monte Carlo simulation. For this we require a **random** number generator, however in reality I will use low discrepancy numbers as a proxy for a random number generator. The advantage of low discrepancy numbers over their alternative, pseudo-random numbers, is that they are spread out accross the plane ([0,1] for example) in less iterations that pseudo-random numbers and hence one requires less simulations to converge a Monte Carlo simulation.
To create the low discrepancy numbers, we are going to use Sobol numbers. I will describe the calculation of Sobol numbers later in the project.
For each iteration of the Monte Carlo simulation, I will perform the following:

1. Simulate the forward rates out to the maturity of the 5 year swap as detailed in the previous section.
2. Calculate the swap payments throughout the lifetime of the swap for every payment date.
3. On swap payment dates, calculate the exposure of counterparty A to counterparty B.
4. Check the variance of the final price of the swap over all iterations **so far** so that we can halt the simulation when the variance of the estimate of the price of the swap is within a predefined tolerance.

I will now briefly describe the discounted swap payments and define the exposure calculation.
For pricing, on each payment date, I calculate the entire Mark to Market of the swap by discounting all of the remaining cashflows to that payment date. At this date I use the forward rates that have been projected to that date to calculate the MTM. The discounted floating payment are calculated as follows:

$$P_{fl_i} = DF_i * \text{Notional} * r_{fl_i} * \frac{1}{2}$$

where DF is the discount factor to present from payment date i. The discounted flixed payments are as follows:

$$P_{fix_i} = DF_i * \text{Notional} * r_{fix_i} * \frac{1}{2}$$

So the MTM_i for the remains of the swap looking out from the i^{th} payment date is the discounted sum of the floating payments less the discounted sum of the present payments.
The Exposure, E_i to counterparty A for the remains of the swap looking out from the i^{th} payment date defined as:

$$E_i = max(PV_i, 0) = (PV_i)^+$$

From the Monte Carlo simulation, I therefore calculate a vector of exposures accross different payment dates for each iteration which reveals a distribution of exposures for each payment date at the end of the simulation. I observe the following results:
In order to calculate the Counterparty Valuation Adjustment (CVA) that counterparty A makes due to the credit risk of entering into a swap with a counterparty B, I will use a combination of the exposures, probability of default of counteparty B and recovery rate of counterparty B to computed an expected loss from counterparty B defaulting. I first assume that defaults occur halfway in between consecutive payment dates.
This results in a CVA for each payment date of:

	0	**1**	**2**	**3**	**4**	**5**
Payment Dates CVA	1.16668304288	2.33170440067	3.41051832239	4.21423780432	4.61071283942	4.58784

Part II

Credit Portfolio Fair Spread and Sensitivity Analysis

Chapter 5

Introduction to CDS

In this section, I will estimate the fair spread for a portfolio of Credit Default Swaps on 5 reference names (Basket CDS) for each of the **k^{th} to default instruments**.

Definition 5.1. A **Credit Default Swap (CDS)** is a financial derivative which essentially offers the buyer insurance against the underlying instrument defaulting whom is specified in the contract. For this protection, the buyer makes regular payments to the seller (specified in the contract); in the event that the underlying defaults, the buyer receives a payout and discontinues premium payments to the seller, (i.e. the swap expires). Note however, in some cases, the insurance is only against a change in the the credit of the underlying instrument as opposed to a default.

CDS's are a way for buyers to hedge their counterparty credit risk, however, the hedge is not perfect seeing as the buyer then receives further counterparty risk to the seller of the CDS. This is an important point because issuers of CDS are not required to hold reserves for payouts by regulation and thus, the leveraged effect of an underlying instrument defaulting can have huge consequences if one counterparty who happens to be an issuer of CDS is also unable to payout and hence goes into default, in addition not paying can put the buyer of the CDS in their own financial risk (counterparty credit risk).

In general, CDS for most instruments are over the counter (OTC), however, a small number of underlying instruments have actively traded CDS contracts of differing Tenors from 6 months out to 10 years in some cases. These underlying instruments tend to be large banks. For that reason, I will use 5 European / American banks as the reference names in the Basket because data on the available credit spreads is more widely available. This will allow me to estimate the default probabilities for these banks collectively in order to simulate outcomes of the Basket CDS and hence estimate a fair spread for the portfolio from a Monte Carlo simulation.

Definition 5.2. The **Basket CDS** is a credit derivative which combines the credit risk of a *group or basket* of underlying reference names. It is a contract that provided a payoff when any of the underlying reference names default (or trigger the contracts default clause) in return for payment of a premium rate (spread) until this default occurs.

The contract will also specify the number of defaults after which payoffs are generated. For this reason, there are several types of basket default swaps. The popular ones are as follows:

- **first-to-default** :- Premium payments are made until the first underlying reference name triggers the contracts default clause a upon which a Compensation payoff is received by the buyer of the CDS.
- **k-th-to-default** :- Premium payments are made until k underlying reference names have triggered the contracts default clause. A Compensation payoff is then received by the buyer of the CDS.
- **n-out-of-m-to-default** :- The seller pays the difference of the notional amount and the recovered value for each of the first defaults until the n^{th} default occurs when the buyer also stops paying the CDS premiums.

- **all-to-default** :- The buyer pays the CDS premium until all of the underlying reference names have defaulted when the buyer receives a compensation payment from the seller of any lost notional amount of the basket CDS.

In this project, we will consider the k^{th} to default basket CDS.

Definition 5.3. The joint distribution of n uniform random variables $(U_1, ..., U_n)$ is defined to be the **Copula** function; where Σ_ρ is the correlation matrix that defines the dependency structure of the random variables.

$$C(u_1, u_2, ..., u_n; \Sigma_\rho)$$

The idea here is that one can simply transform a random variable by its own CDF to obtain the copula function. A copula is useful because it allows one to understand the dependence structure among multiple random variables in a joint distribution without ever knowing the underlying joint distribution. Furthermore, for such random variables with known marginal distributions, we are able to derive the joint distribution function from the copula function. So in practice the process is as follows:

1. Generate a vector of correlated Gaussian random variables.
2. Transform the random variables to Uniform[0,1] random variables using the Gaussian CDF.
3. Transform the U[0,1] variates to their respective marginals using their respective inverse cumulative marginal probability functions.

$$C(u_1, u_2, ..., u_n; \Sigma_\rho) := \mathbf{P}(U_1 \leq u_1, ..., U_n \leq u_n)$$

$$= \mathbf{P}(F_1(X_1) \leq u_1, ..., F_n(X_n) \leq u_n)$$

$$= \mathbf{P}(X_1 \leq F_1^{-1}(u_1), ..., X_n \leq F_n^{-1}(u_n))$$

$$= \mathbf{P}(X_1 \leq x_1, ..., X_n \leq x_n) =: F(x_1, ..., x_n)$$

Theorem 5.4. ***Sklar's theorem*** *states that any joint distribution can be expressed as a copula function and if the joint distribution is continuous, then the copula function is unique.*

Sklar's theorem allows the derivation of the copula density function from a multivariate PDF.
With this if we assume that the default distribution is continuous, i.e. a reference name can default in any arbitrarily small time period if it has not already defaulted, then we have a unique copula function to describe the joint distribution of the defaults. So we will look to parametrise 2 elliptical copulas to describe and simulate the defaults:

- Gaussian copula
- Student's t copula

The Gaussian copula is easier to parametrise and makes sense when we assume that the underlying default probabilities have Gaussian marginal. However, it is also likely that in reality, the tails of the default probability marginals are larger than Gaussian, here the student's t distribution incorporates this tail dependence.

5.1 Data Collection

- Discount Curves for discounting simulated future cash flows in the CDS:
  - ⋆ SONIA (market convention for discounting)
- Default Rates for all 5 reference names:
  - ⋆ Credit Spreads for Tenors out to 10 years to sufficiently cover the term structure of hazard rates for most simulated default times. *I assume that changes in the hazard rate are negligible at the long end past 10 years.*
  - ⋆ Historical Credit Spreads for the most liquid tenor of 5 years to parametrise copula functions.

5.2 Assumptions

- I am going to assume that the buyer of the basket CDS makes premium payments **quarterly** as this seems to be typical in the market.
- I am going to initially set the **Recovery** rate of a defaulted reference name to 40% as this is again typical in the market. I will then alter this rate and analyse the effects.

5.3 Pricing

5.3.1 Vanilla CDS

The present value of the sum of the *premium* payments that the buyer of the CDS makes to the seller are calculated as follows:

$$\Pi = \sum_{i=1}^{n} \pi \cdot N \cdot (\tau_i - \tau_{i-1}) \cdot DF(\tau_i) \cdot S(\tau_i)$$

where:

- n is the number of potential premium payments if the underlying doesn't default.
- π is the premium rate (spread) that is paid by the buyer for this credit protection.
- N is the notional of the contract.
- τ_i is the date or time of the i^{th} premium payment.
- $DF(\tau)$ is the discount factor from τ to present.
- $S(\tau)$ is the probability that the contract's underling reference name has *survived* to the time τ.

On the other hand, the present value of the *compensation* payment that the seller of the CDS makes to the buyer is calculated as follows:

$$C = \sum_{i=1}^{n} N \cdot (1 - R) \cdot DF(\tau_i) \cdot PD(\tau_i)$$

where:

- R is the recovery rate of the contract's underlying reference name.
- $PD(\tau)$ is the probability that the contract's underlying reference name has *defaulted* by the time τ.

Finally the Mark-to-market of the contract from the buyer's point of view is simply the difference:

$$MTM = C - \Pi$$

In the world of pricing, the MTM is set to 0 in order to calculate a *fair* value for π that results in a zero MTM for the instrument.

$$\hat{\pi} = \frac{\sum_{i=1}^{n} N \cdot (1 - R) \cdot DF(\tau_i) \cdot PD(\tau_i)}{\sum_{i=1}^{n} N \cdot (\tau_i - \tau_{i-1}) \cdot DF(\tau_i) \cdot S(\tau_i)}$$

5.3.2 Basket CDS

For the k^{th} to default basket CDS, one calculates the instrument's payments very similarly to the Vanilla CDS above. However, the derivation of a fair value for π, the premium rate of the basket CDS, is more complex as one isn't able to bootstrap default probabilities for the k^{th} to default reference name as it is *not* known which reference name this will be.

The computation of the 2 legs, given that a premium rate π is known, is analogous if the following terms are redefined where I define L to be the number of underlying reference names in the basket CDS.

- $S(\tau)$ is the probability that **strictly greater than** $(L-k)$ of the contract's underling reference names have *survived* to the time τ.
- $PD(\tau)$ is the probability that **strictly less than** (k) of the contract's underlying reference names have *defaulted* by the time τ.

So as above, to calculate a *fair* value for π that results in a zero MTM for the basket instrument, I simulate the defaults in the basket and then price the compute the cashflows for the instrument with the default times of the first k reference names known. I compute the cashflows as follows:

$$\Pi = \sum_{i=1}^{n} \pi \cdot N \cdot (\tau_i - \tau_{i-1}) \cdot DF(\tau_i) \cdot \mathbf{1}_{\{\tau_i < \tau^{\mathbf{D}}\}}$$

$$C = \sum_{i=1}^{n} N \cdot (1-R) \cdot DF(\tau_i) \cdot \mathbf{1}_{\{\{\tau_i \geq \tau^{\mathbf{D}}\} \cap \{\tau_{i-1} < \tau^{\mathbf{D}}\}\}}$$

where $\tau^{\mathbf{D}}$ is the exact time of default for the k^{th} reference name.
Therefore, I will compute the fair spread by simulation as follows:

$$\hat{\pi} = \frac{\sum_{i=1}^{n} N \cdot (1-R) \cdot DF(\tau_i) \cdot \mathbf{1}_{\{\{\tau_i \geq \tau^{\mathbf{D}}\} \cap \{\tau_{i-1} < \tau^{\mathbf{D}}\}\}}}{\sum_{i=1}^{n} N \cdot (\tau_i - \tau_{i-1}) \cdot DF(\tau_i) \cdot \mathbf{1}_{\{\tau_i < \tau^{\mathbf{D}}\}}}$$

5.4 Monte Carlo Procedure

In order to carry out a Monte Carlo simulation, I will carry out the following procedure:

1. I first need to be able to simulate defaults of the 5 reference names in the basket CDS. The joint default distribution of these reference names is not known and so I must infer the joint default distribution empirically. I will therefore fit a copula to the default data to proxy the joint default distribution of reference names.
2. At the outset of the project, we are not sure what shape best describes the joint default distribution of the reference names. So I have chosen the Gaussian copula and the Student's T copula which are both continuous copula because they are easier to implement and there are no specific reasons why they should not fit the data. To parametrise the copulas to the underlying default data, I must:
   (a) Pull a History of CDS spreads of equal tenor (the most liquid) for each reference name.
   (b) Calculate a correlation matrix from the time series of CDS spreads.
3. The copula then allows me to simulate 5 Uniform Random Variable which can then be used to calculate the exact default time of the 5 reference names by using their term structure of hazard rates.
4. Having simulated 5 default times, 1 for each reference name, I am able to simulate the cash flows in the basket CDS which I then discount to the present using the discount curve data.

5. The Monte Carlo Algorithm for each iteration is as follows:

   (a) Use the Copula (T or Gauss) to simulate and calculate the default time of each reference name.

   (b) Evaluate the basket CDS cash flows for this simulation and discount them to present.

   (c) Store the sum of the Premium leg cash flows (the cash flows made by the buyer) for this iteration separately from the sum of the Compensation leg cash flows (any compensation payments made by the seller due to default of an underlying reference name).

   (d) Store a running average of the sum of Premium leg cash flows and Compensation leg cash flows separately, updating them both at each iteration.

   (e) Stop the iterations when both running averages' variances converge within an arbitrarily chosen tolerance of 10E-20. At which point, calculate an estimate of the fair spread for the basket CDS by subtracting the average of the sum of the Compensation leg cash flows from the average of the sum of the Premium leg cash flows.

Chapter 6

Market Implied Discount Factors

In order to estimate the market's view of the value of capital in the future, I will use the SONIA swap rate as it is the currently accepted market convention to proxy investing money at a rate with zero risk. SONIA swaps are also a liquid and traded instrument meaning that there is plenty of historical data with differing swap maturities to bootstrap discount factors from.

$$DF_{\tau_n} = \left(\frac{1}{1 + i_{\tau_n}}\right)^t$$

where i_{τ_n} is the market swap rate for a SONIA swap with a maturity of τ_n.

I then use a log-linear interpolation for calculating discount factors between the tenors provided in the market. I use a log-linear interpolation as the discount factor is the inverse of the rate that we are observing in the market. i.e.

$$logDF_\tau = \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i} logDF_{\tau_{i+1}} + \frac{\tau_{i+1} - \tau}{\tau_{i+1} - \tau_i} logDF_{\tau_i}$$

From the data, I calculate the following Discount curve:

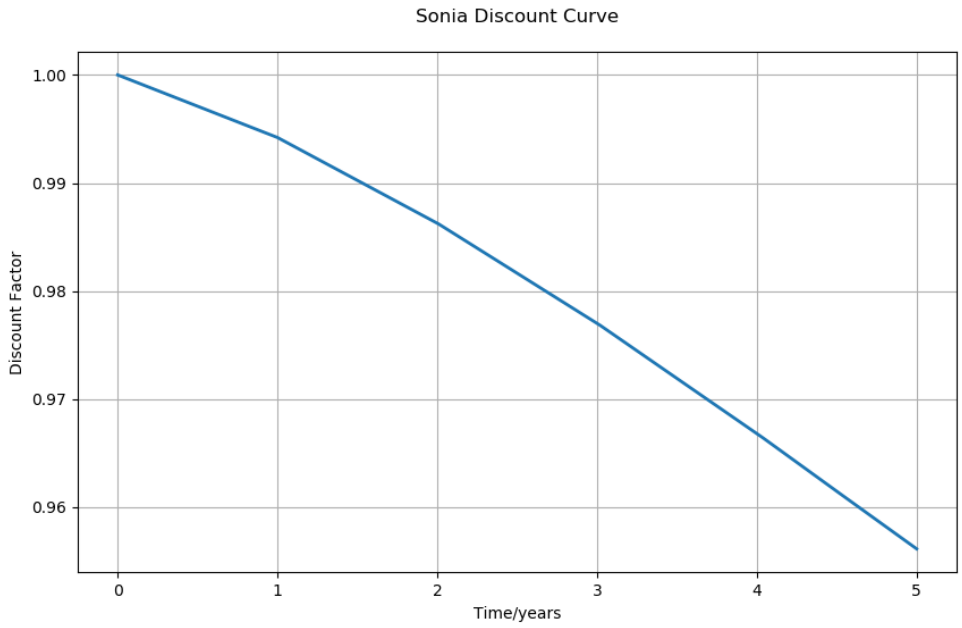

Figure 6.1: SONIA Discount Curve

Chapter 7

Implied Probability of Default

In order to estimate the default probabilities for each of the probabilities and hence the term structure of hazard rates, we bootstrap survival probabilities from the Credit Spreads data as follows:

$$\mathbf{S}_n := \frac{\sum_{i=1}^{n-1} DF_{\tau_i} \cdot \Big[(1-R)\mathbf{S}_{i-1} - \mathbf{S}_i(1-R+\psi_n(\tau_i - \tau_{i-1}))\Big]}{DF_{\tau_n}(1-R+\psi_n(\tau_n - \tau_{n-1}))} + \frac{\mathbf{S}_{n-1}(1-R)}{1-R+\psi_n(\tau_n - \tau_{n-1})}$$

where:

- N is the number of credit spreads of differing tenor that I have available to me from the market.
- $n \in 1, ..., N$
- R is the Recovery rate which is defined in the market to be at 40% for these credit spreads.
- τ_n is defined as the n^{th} tenor where $\tau_{n-1} \leq \tau_n \forall n \in 2, ..., N$.
- DF_{τ_n} is a factor that discounts in pounds sterling from τ_n to present.
- ψ_n is defined as the n^{th} credit spread.
- $\mathbf{S}_n$ as the n^{th} survival probability implied by ψ_n.
- $\mathbf{S}_0 := 1$
- $\mathbf{S}_1 := \frac{\mathbf{S}_0(1-R)}{1-R+\psi_1(\tau_1 - \tau_0)}$

With equivalent hazard rates, λ_n, as follows:

$$\lambda_n = -\frac{log(\mathbf{S}_n/\mathbf{S}_{n-1})}{\tau_n - \tau_{n-1}}$$

for $n \in 1, ..., N$ where λ_n is defined as the hazard rate in the time interval $[\tau_{n-1}, \tau_n)$.

From the market credit spreads, I calculate the following implied term structure of hazard rates:

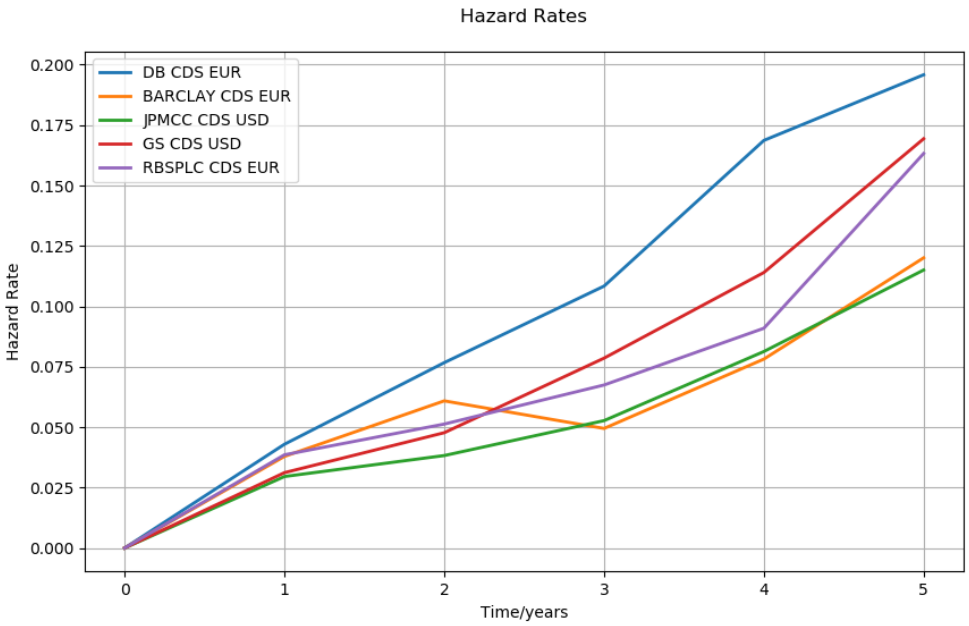

Figure 7.1: Term structure of hazard rates for each reference name

Chapter 8

Copula Parametrisation

I now estimate the correlations between the default probability of each of the reference names. As the default probability correlation between 2 reference names is not dependent on the tenor, but instead on the relationship between the 2 reference names (i.e. positions with one-another), we use the most liquid tenor at 5 years of historical credit spreads for deriving the probability of default correlations as we can compare this data for each reference name side-by-side. It is however worth noting that Credit spreads depend on other external market influences than just the default probability of a company as they are a traded instrument; therefore, credit spreads are not necessarily a perfect indicator for the default probability correlation structure between counterparties. But due to data constraints and the fact that default probabilities themselves are not directly 'tradable' (not observable); I will use credit spreads as other methods of deriving correlations are outside of the scope of this project.

To calculate the correlation matrix, I begin by calculating log returns from the historical credit spreads for each reference name. I calculate the **weekly** returns to begin with as I believe that it the weekly returns will have significantly less noise than daily returns. Considering that there needs to be sufficient data to calculate meaningful correlations that don't change wildly based on sample size, I choose arbitrarily to use at least 100 data points to calculate correlations from; this results in historical spreads from at least 2 years back.

$$rln_{ji} = Log(\psi_{ji}/\psi_{ji-l})UPDATETOAVERAGE$$

where:

- i indexes each day in the historical data. Meaning that ji gives results in the calculation of one return every j business days. I begin by setting $j = 1$.
- l is the lag in numbers of days. I use a lag of 5 to result in weekly returns.
- rln is the log return.

8.1 Gaussian Copula

For the Gaussian copula, I calculate the **Linear, (Pearson)** correlation, $\mathbf{C}^{lin}$, between log returns and use this to parametrise the copula. Using linear correlation as a measure of co-dependance is only acceptable when the underlying variables are Gaussian. Therefore we will use the linear correlation to parametrise the Gaussian copula. Furthermore, for linear correlations the following hold:

- A correlation of 0 is equivalent to independence for Gaussian distributions.
- Given elliptical marginal distributions and their correlation, it is possible to construct a joint distribution.

I calculate the log of the credit spreads historical *weekly* returns for each reference name as below:

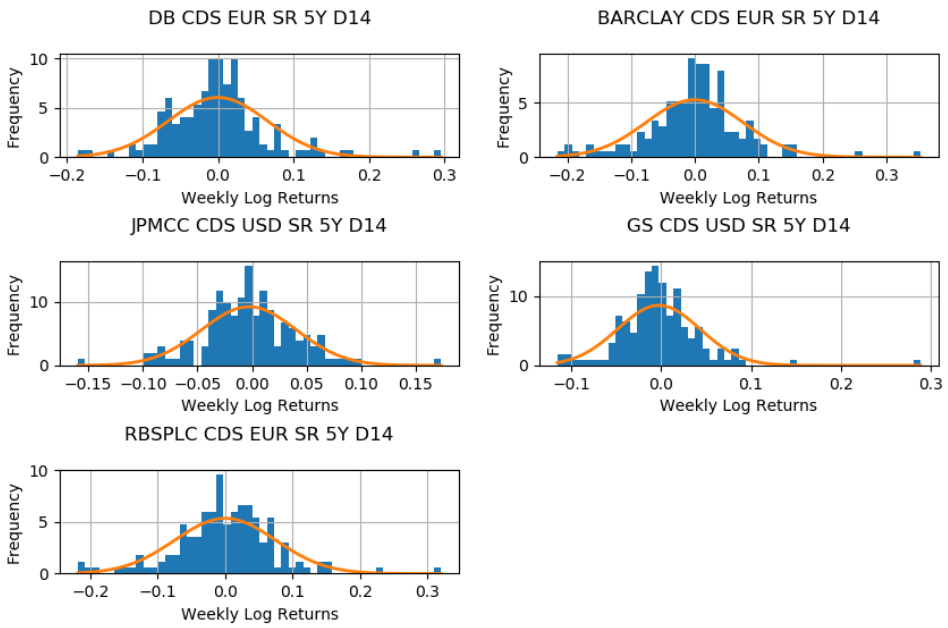

Figure 8.1: Weekly Log Returns

In doing so, I am able to calculate a Pearson correlation matrix:

	0	**1**	**2**	**3**	**4**
0	1.0	0.18649886011	0.101372525924	0.115160494021	0.18647762541
1	0.18649886011	1.0	0.114773569696	0.109328816372	0.22762339474
2	0.101372525924	0.114773569696	1.0	0.216432755271	0.119391581963
3	0.115160494021	0.109328816372	0.216432755271	1.0	0.11993594199
4	0.18647762541	0.22762339474	0.119391581963	0.11993594199	1.0

8.2 Student's T Copula

On the other hand, for the Student's t copula, the underlying variables are not Gaussian and so using the linear correlation would be unsuitable here. Instead I use the **Rank** correlation because this gives scale invariant estimates of the correlation which are more suitable for non-Gaussian random variables. Intuitively, the rank correlation measures the dependency between a big movement in one reference name's credit spread in comparison to a movement of equivalent ranking in another reference name's credit spread. I calculate **Spearman's Rho**, which is a rank correlation, by *ranking* the observations in order and then scaling them to $[0, 1]$ to output scale invariant estimates of the correlation. An alternative Rank correlation that can also be used is **Kendall's Tau**. As stated above, this is important for the student's t copula because the student's t distribution does not satisfy the constraints of the linear correlation above.

To calculate the Rank correlation, I first compute the absolute differences from the historical credit spreads for each reference name. I then calculate the Empirical CDF of these differences and use it to transform the data (*absolute differences*). Absolute differences are given by:

$$\text{diff}_{ji} = |\psi_{ji} - \psi_{ji-l}| UPDATETOAVERAGE$$

where diff is the absolute differences for historical day (ji).

I calculate the weekly Absolute differences taking the average of each week and subtracting week-on-week; the resulting differences have the following distributions:

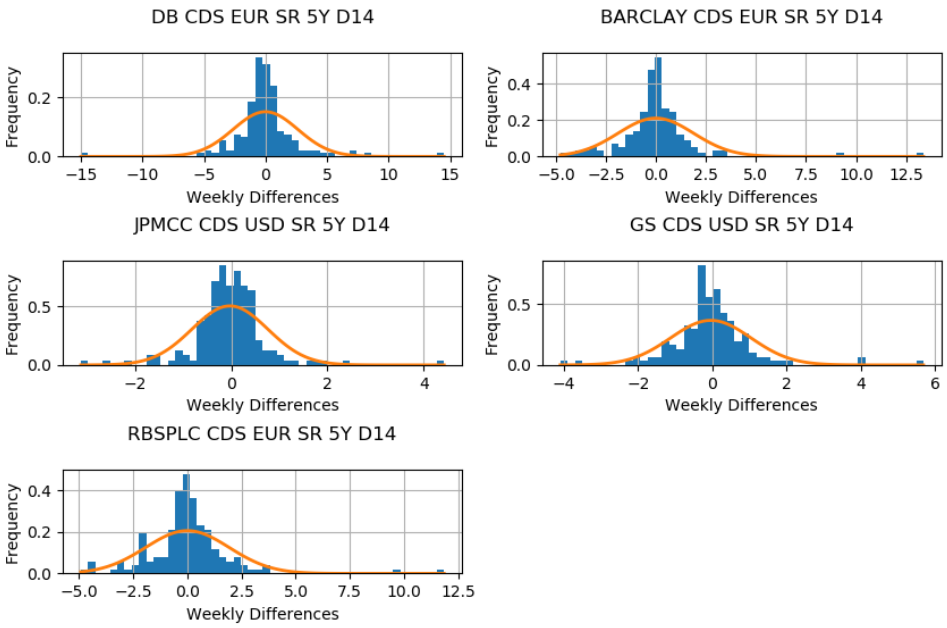

Figure 8.2: Weekly Absolute Differences

The Empirical Cumulative Distribution function (ECDF) is given by:

$$\hat{F}_n(t) := \frac{1}{n} \sum_{i=1}^{n} 1_{x_i \leq t}$$

where n is the sample size and i indexes the sample.

We see from the below histograms, that the transformed data is approximately uniform as expected:

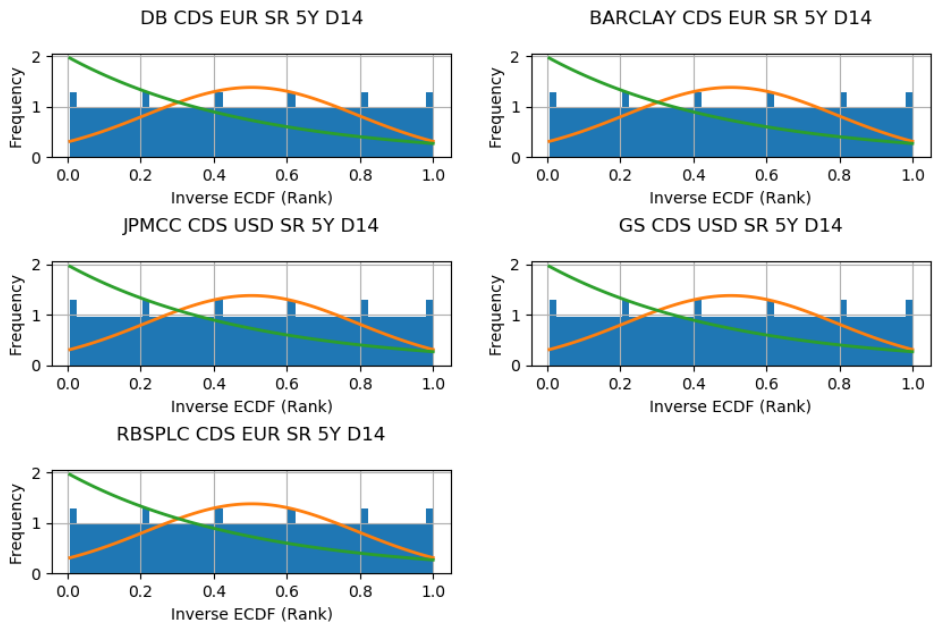

Figure 8.3: Distribution of Differences by their ECDF

The Rank correlation, $\mathbf{C}^{rank}$, is then calculated from the transformed differences.

8.3 Random number generation

To sample from the parametrised copulas, one needs to be able to sample random variables from U[0,1]. Computationally, the following random number generation techniques are commonly used:

- Pseudo-random numbers.
- Low discrepancy numbers.

Pseudo random numbers use number theory to generate *a sequence of random numbers* on an n-dimensional plane where the only restriction on the elements of the sequence is that they are constrained to the plane. Each sequence is uniquely defined by a unique seed (number) that is arbitrarily chosen outside of the scope of the generator.

Low discrepancy numbers are a subset of pseudo-random numbers. They have the additional feature that each new element in the sequence is generated to be maximise the modulus of the difference between the new element and all other elements in the sequence. This ensures that Low discrepancy numbers spread over the plane much quicker than Pseudo-random numbers which can cluster. It is a useful feature for Monte Carlo simulation as one can achieve convergence of the estimate in far fewer simulations.

In this project, we will use **Sobol numbers** which are a type of low discrepancy numbers.

8.4 Sobol numbers

I briefly outline the generation of Sobol numbers. The algorithm has not been tweaked from the literature as it is very sensitive and number theory is outside of the scope of this project.

1. Start with integers on the interval $[1, 2^b - 1]$ where $b = 32$ representing the 32 bits of the x86 language that I use to generate the sequence.

2. Define x_{nk} to be the n^{th} draw of the k^{th} dimension of the sequence. Then $y_{nk} := \frac{x_{nk}}{2^b}$ where $y_{nk} \in (0, 1]$

3. I next define the *direction integers* which are the underlying basis of the number generation. Define v_{kl} to be the l^{th} direction integer for dimension k. These direction integers are constrained to the following rules:

   - Only the l left-most bits of the bitfield for v_{kl} can be non-zero.
   - The l^{th} bit must be set.

4. Define the *primitive polynomial* $p_k(z) := \sum_{j=0}^{g_k} a_{kj} z^{g_k - j}$ where g_k is the dimension of the polynomial, $p_k(z)$.

5. For a given dimension k with primitive polynomial $p_k(z)$, we have the following:

   - For $l \in \{1, ..., g_k\}$, v_{kl} are **initialised** freely. I discuss this initialisation process below.
   - For $l \geq g_k$:
   $$v_{kl} = \frac{v_{k(l-g_k)}}{2^{g_k}} \oplus_2 \sum_{j=1}^{g_k} {}^{\oplus_2} a_{kj} v_{k(l-j)}$$

6. Then as in other number generators, for each draw, n, one requires a generating integer $\gamma(n)$. It is common and simple to choose $\gamma(n) = n$; however I discuss below that it is far more efficient to choose $\gamma(n) = G(n)$ where $G(n)$ is the **Gray Code**. I explain the Gray Code below. Thus my implementation of the generator uses the Gray Code.

7. Then finally:
$$x_{nk} := \sum_{j=1}^{d} {}^{\oplus_2} v_{kj} \infty_{\{j^{th} bit of \gamma(n) isset\}} \tag{8.1}$$

8.4.1 Gray code integer encoding

Gray Code is an encoding of integers for efficiency in computational iterators / algorithms. The idea is that for each integer n, $\exists$ as bitwise representation of n such that the bitwise representation of $(n+1)$ differs by only **one** bit. I define this representation to be $G(n)$. Furthermore, the one bit that differs is the right-most zero bit of n.

The Gray Code is *not* unique. In my implementation, I use a commonly version of the encoding: $G(n) := n \oplus_2 [n/2]$. Then, because $G(n)$ differs from $G(n+1)$ by one bit, only *one* of the operations in equation 8.1 is necessary. Therefore:

$$x_{nk} = x_{(n-1)k} \oplus_2 v_{jk}$$

8.4.2 Initialisation

I remind the reader of the initialisation of the direction integers which is a problem with many solutions. The behaviour and properties of the generated sequence is heavily dependent on the choice of initialisation of the direction integers. In order to review the quality of an initialisation, I introduce the following number theory definition.

Definition 8.1. Property A (dispersion) states that a low discrepancy is said to have property A if for any binary segment (not an arbitrary subset) of the d-dimensional sequence of length 2^d, there is exactly one draw in each of the 2^d hypercubes that result from subdividing the unit hypercube along each of its unit length extensions into half.

The simplest initialisation is **Unit initialisation** which is easy to implement but leads to poorly generated sequences. Unit initialisations sets:

$$v_{kl} = 2^{b-l}$$

Instead, one can use **Pseudo-random initialisation** which uses a separate pseudo-random number generator to generate $u^*_{kl} \sim U[0,1]$. One then defines $w_{kl} = int(u^*_{kl} \cdot 2^{l-1})$ where $int(x)$ is a function that rounds x to the nearest integer and w_{kl} is only used when it satisfies $w_{kl}\%2 = 1$. Then:

$$v_{kl} = w_{kl} \cdot 2^{b-l}$$

I use the pseudo-random initialisation in my implementation as the behaviour of the generator is much better than unit initialisation.

8.5 Extreme Value Theory

For Elliptical Copula, it can often be the case that the data is not well represented in the tails of the copula. So one can employ Extreme Value Theory (EVT) to address this issue.

Chapter 9

Simulation & Observations

9.1 Estimation of Fair Spread by simulation

9.1.1 Reference name co-dependencies

Having computed default probabilities and discount factors, I first parametrise the copulas. I observe the following co-dependencies between the reference names from observing the copulas:

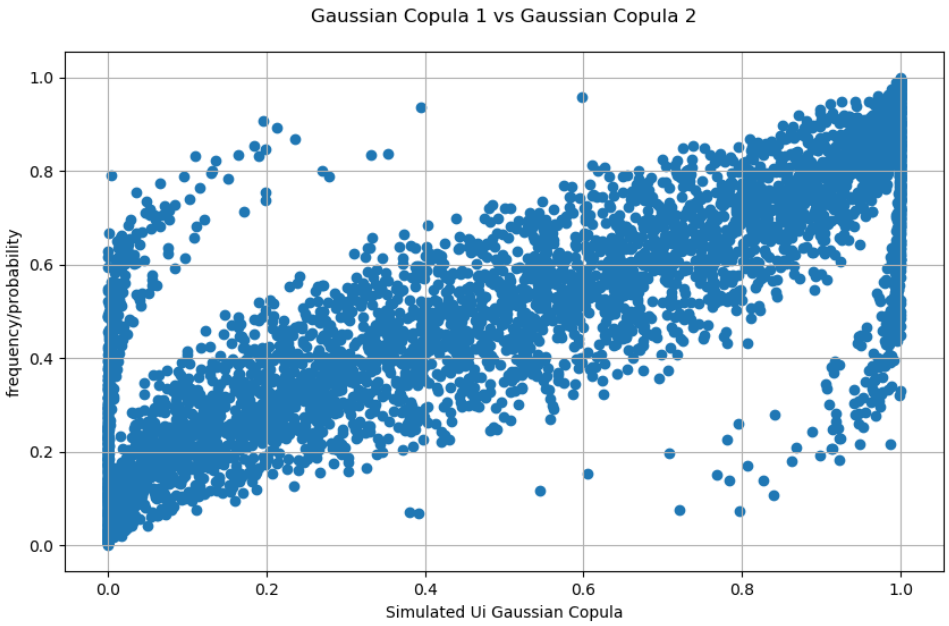

Figure 9.1: Joint Distribution of 1st and 2nd reference names in the Gaussian Copula

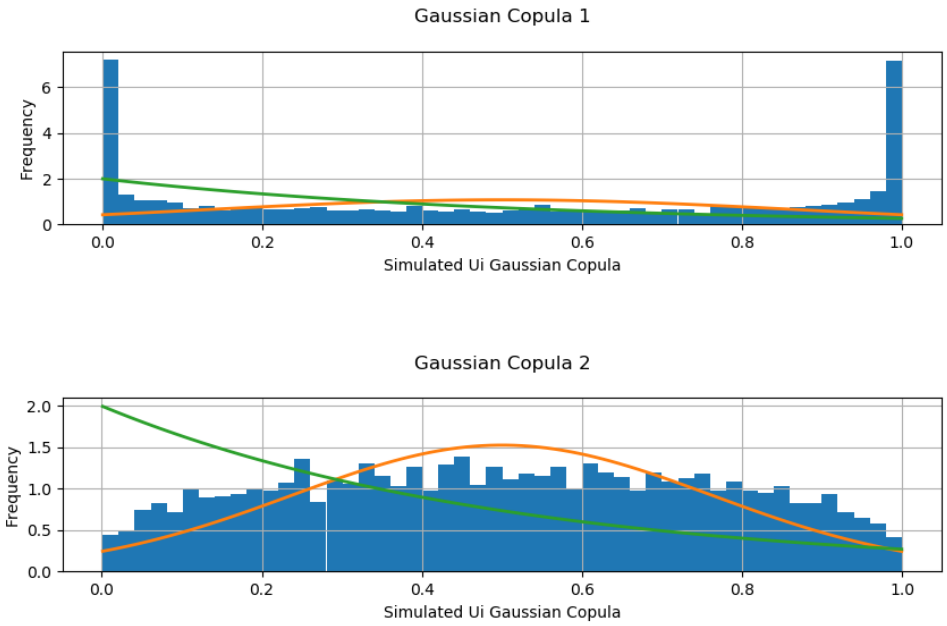

Figure 9.2: Marginal Distributions of 1st and 2nd reference name in the Gaussian Copula

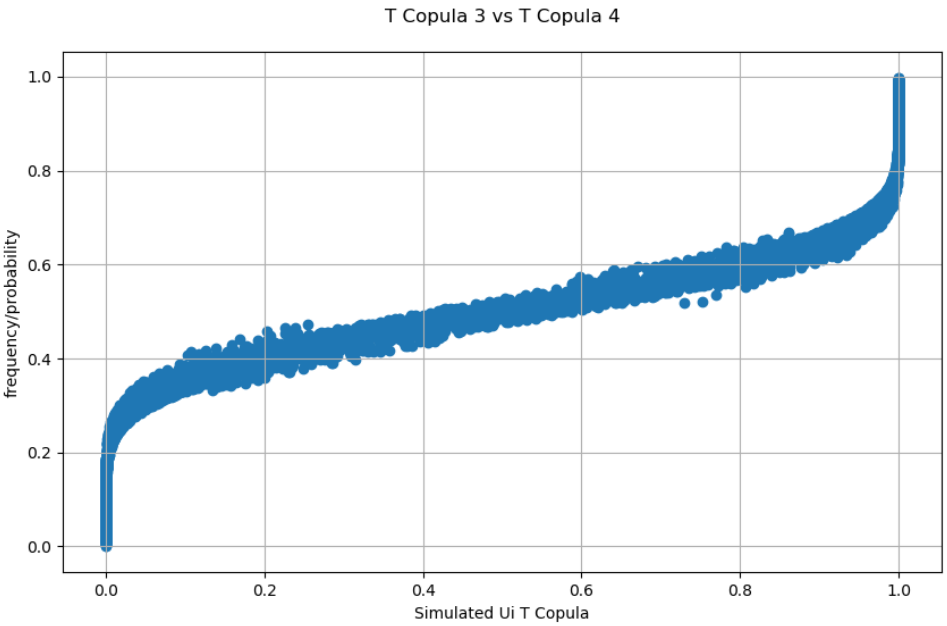

Figure 9.3: Joint Distribution of 3rd and 4th reference names in the Student's T Copula

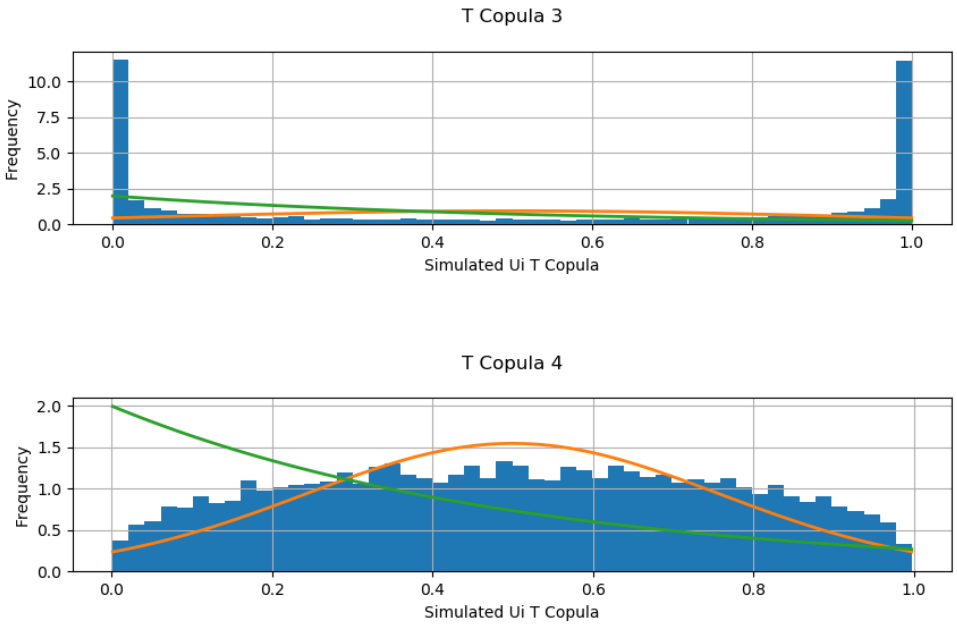

Figure 9.4: Marginal Distributions of 3rd and 4th reference name in the Student's T Copula

9.1.2 Monte-Carlo Observations

I now run the simulation where the running averages and running variances are displayed below. This illustrates the benefit of using Low discrepancy numbers because one observes that the conversion of the Monte Carlo estimate occurs in far fewer iterations when the simulation contains a relatively small number of dimensions. *Note that the Monte Carlo simulations were stopped before the maximum number of iterations of 5000 if they converged, this is the reason for the running averages falling suddenly to 0.*

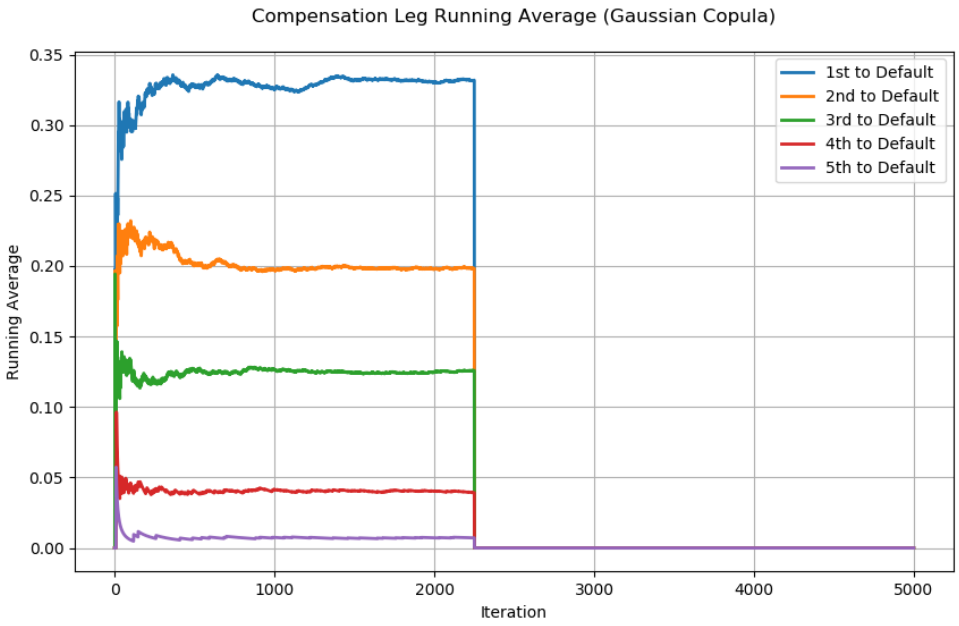

Figure 9.5: Running Average of Compensation Leg for the Gaussian Copula

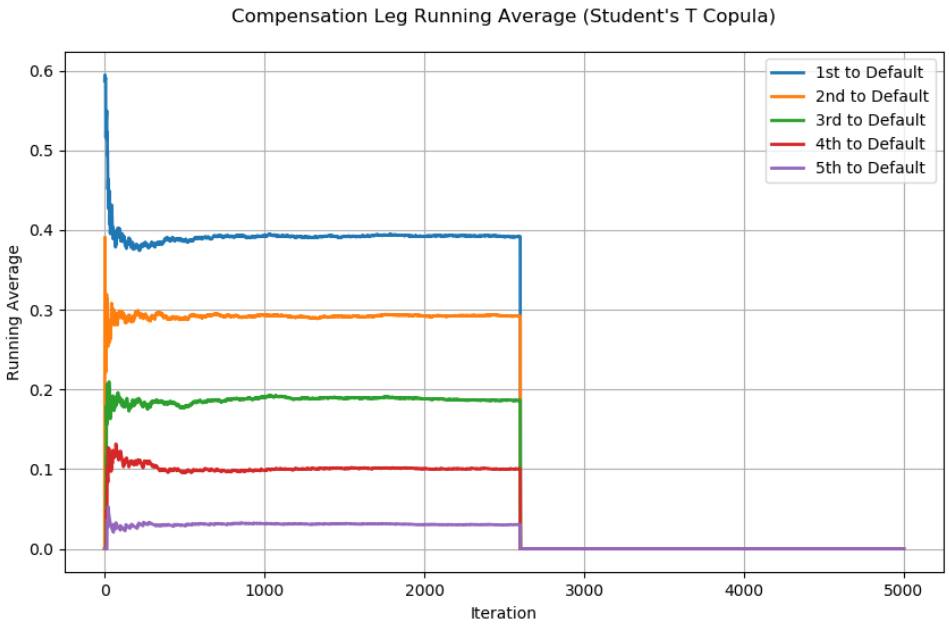

Figure 9.6: Running Average of Compensation Leg for the Student's T Copula

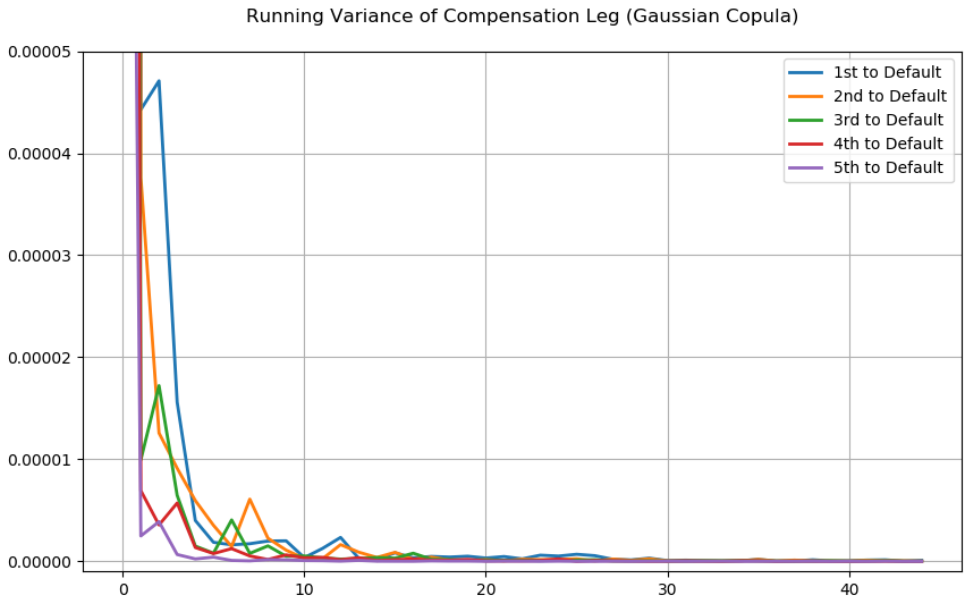

Figure 9.7: Running Variance of Compensation Leg for the Gaussian Copula

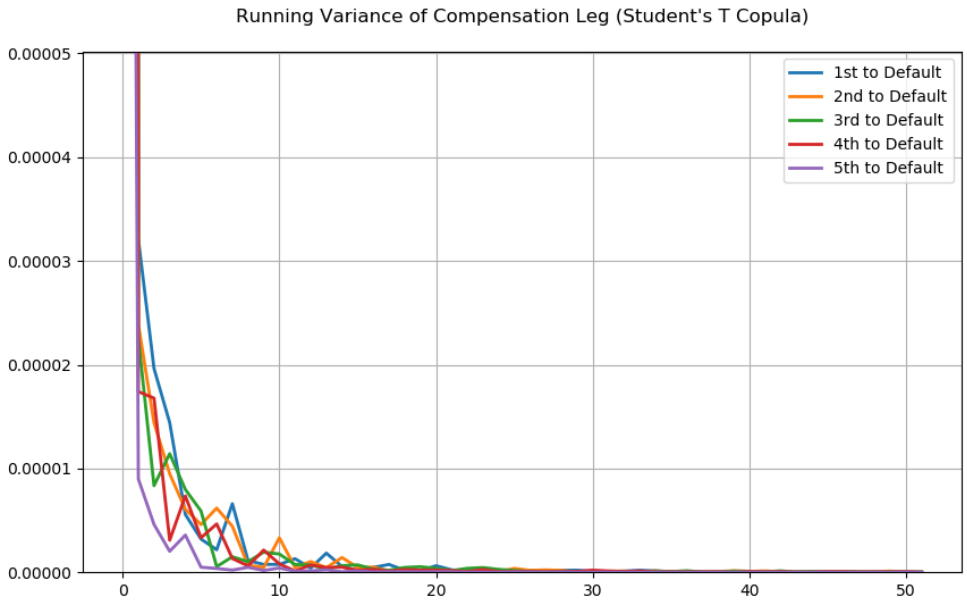

Figure 9.8: Running Variance of Compensation Leg for the Student's T Copula

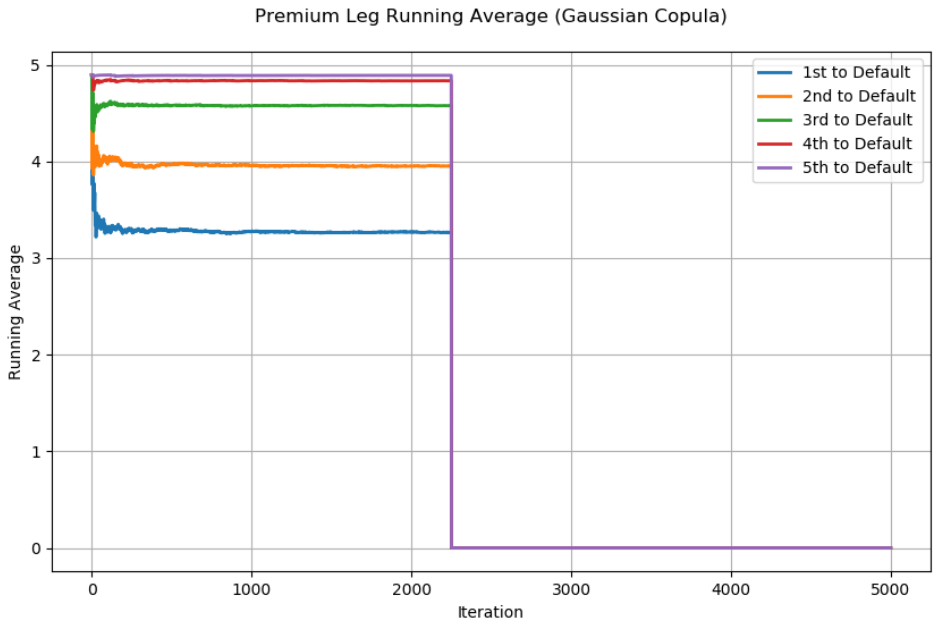

Figure 9.9: Running Average of Premium Leg for the Gaussian Copula

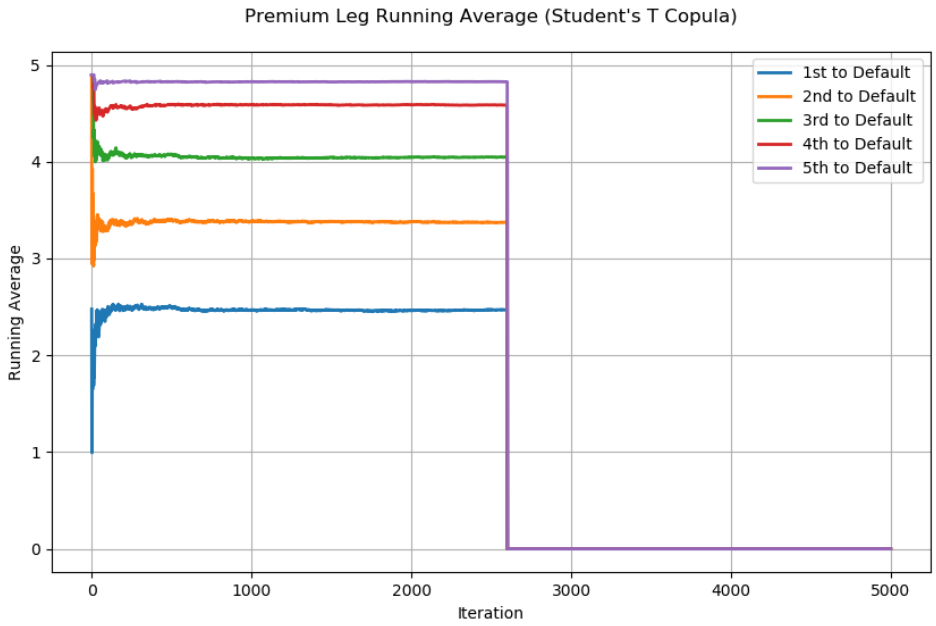

Figure 9.10: Running Average of Premium Leg for the Student's T Copula

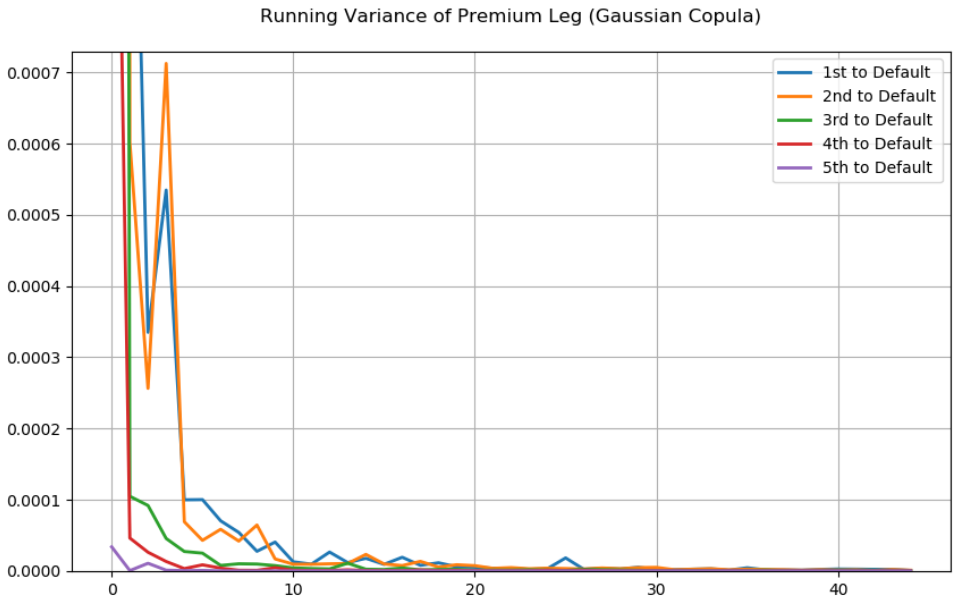

Figure 9.11: Running Variance of Premium Leg for the Gaussian Copula

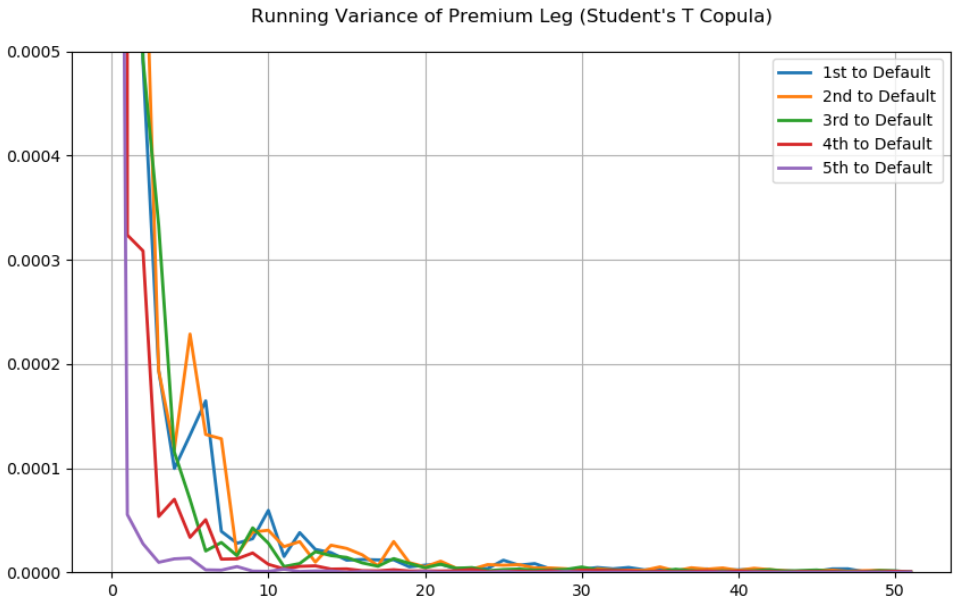

Figure 9.12: Running Variance of Premium Leg for the Student's T Copula

The simulation outputs the following estimates of fair spreads for the 1^{st} to the 5^{th} to default instruments respectively for both the Gaussian and the Student's T assumption.

	1	2	3	4	5
Gaussian	0.101567982756	0.0501498757102	0.0274859057615	0.00812602603563	0.00146327454895
Student's T	0.15864249598	0.0865140811474	0.046047608693	0.021924513683	0.00629194399778

I compare these values with the market spreads on 5 year CDS contracts for the reference names as it would make intuitive sense to observe that the smallest observed market spread is greater than or equal to the 5^{th}-to-default fair basket spread seeing as the 5^{th}-to-default basket must by definition default after or at the same time as the default of the reference name with the smallest spread. Conversely, one should see that the largest observed marked spread is less than or equal to the 1^{st}-to-default fair basket spread seeing as the 1^{st}-to-default basket must by definition default before or at the same time as the default of the reference name with the greatest spread. So comparing these rates with the 5 year CDS contracts, I notice that this condition is met.

I next compare the differences in the fair spreads for the different k^{th}-to-default baskets due to the choice of underlying default distribution.

I observe that the spread of the baskets that sampled from a Student's T distribution is always greater than the Gaussian comparison. The earlier simulated defaults of the Student's T copula reflect the copula's heavier tails which can be seen in Figure 9.3 in comparison to Figure 9.1.

Chapter 10

Model Validation

10.1 Re-Pricing with Fair Spread to compare Premiums and Compensation

In this section, I use the fair spread calculated above and perform another simulation, this time using the fair spread. The goal is to check that the mean value of the compensation legs should be roughly equal to the mean value of the premium legs, otherwise the spread is not fair.

10.2 Comparison of spreads by instrument

If one compares the k^{th} to default spread with the $(k-1)^{th}$ spread, the latter should be less because there is by definition a lower probability that the k^{th} reference name defaults before the maturity of the CDS. Stated explicitly, this is because the $(k-1)^{th}$ reference name must, by definition, have already defaulted. This constraint is satisfied according to resulting spreads above.

Conclusion

If I consider an individual Asian Call Option, then the difference in expected value between the different specification of contract doesn't differ overly much. However, I notice that the combination of Floating Geometric and Fixed Arithmetic have slightly higher values than Floating Arithmetic and Fixed Geometric. The opposite scenario is observed for the Asian Put Option. This is because the Average is larger when calculated using the Arithmetic approach over the geometric approach. This penalises the payoff of the Asian Call Option with floating strike and rewards the payoff of the Asian Call Option with fixed strike.

The option is priced at the money, hence the expected value of the put is only marginally smaller than the expected value of the call. This makes sense from the drift of the underlying that rewards the call and penalises the put.

The introduction of Jumps in the underlying stock substantially increases the Expected Payoff of the Asian Call Option and marginally increases the value of the Asian Put Option. If we consider the jumps to be similar to an increase in the volatility of the stock, then despite the interest rate adjustment paramter r_j, the increased spread of paths causes the possible payoff to increase to higher levels. Whereas in the opposite direction, both the payoff and underlying stock are bounded below by zero. This effect is observed by comparing the histograms given with and without jumps. I notice, that the probabilty of having a discounted payoff of 0 is lower when jumps are introduced.

References

- **CQF Intro to Python Lecture given by Dr. Yves J. Hilpisch with notes at: https://gist.github.com/yhilpisch/e2af84bc2ff1cab88d6be019459f5672.**

- **CQF Lecture Notes**