Credit Spread for a Basket Product (CR)

Jeffrey Chong Hong Seng

June 2023 Cohort

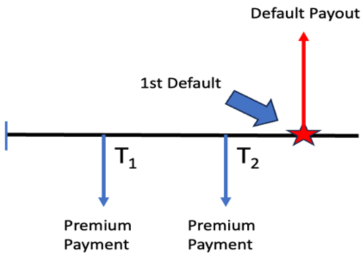
Table of Contents

1 Theory

1.1 Introduction - Credit Derivative

1.1.1 Credit Default Swap

Credit Default Swap is a credit derivative that provides payout when an underlying issuer undergoes a credit event. It provides credit hedging to the single underlying

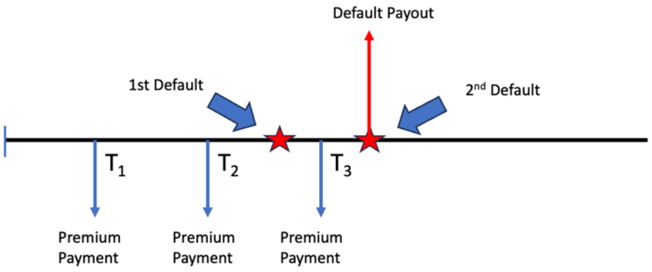


1.1.2 Introduction Kth to default

CDS usually refers to a single underlying, basket credit derivatives are financial instruments that refers to the credit risk of multiple underlying. Kth to Default is a very common basket credit derivative.

Kth to default (e.g. 2nd to default) is a credit derivative whose default leg payout is determined by the time of the Kth default (2nd default) in a basket of credit references. Unlike the CDS, this provides credit hedging to a basket of underlying, although the hedge is not as perfect as CDS.

2nd to Default



1.1.3 Credit Event

The occurrence of a Credit Event is what triggers the payment of a credit protection amount from Seller to Buyer. Broadly speaking, the occurrence of a Credit Event reflects a decline in the creditworthiness of the relevant Reference Entity. Some of the events contained in ISDA 2014 definitions: Bankruptcy, Failure to Pay, Restructuring etc. (ISDA, 2020). Henceforth, the first credit event of an issuer with be known as the default time.

1.2 Introduction – Pricing Concepts

1.2.1 Modelling of Credit Event(s)

1.2.1.1 Poisson Processes

A Poisson process is widely used to model the time at which arrivals enters a system (and hence it's used to model the default time). The parameter required is intensity (Lambda), also known as hazard rate.

The first time of arrival, X_1, in a Poisson process can be modelled by $P(t_n)$

$$PDF(X_1>t) = P(\text{no arrival in } (0,t]) = P(\text{arrival at } t) = P(t_n) \; =$$

Which is an exponential function, and hence it's CDF is

$$CDF(t) =$$

Given that continuous CDFs are uniformly distributed, uniform random values, U, can be generated, to produce randomly generated default time using the function below

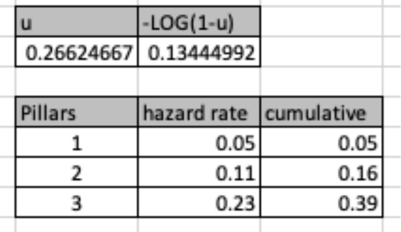
$$CDF(t) = U = \;\; , 1 - U = ,$$

For a homogeneous Poisson process, the hazard rate is constant over time, and it's easy to determine the default time as t is the only unknown.

However, hazard is not constant over the period 0 to t , hence discretization λt means :

Given that difference between t for each interval is 1.

T can be then inferred from the cumulative hazard rates.



u	-LOG(1-u)
0.26624667	0.13444992

Pillars	hazard rate	cumulative
1	0.05	0.05
2	0.11	0.16
3	0.23	0.39

Example, the -log(1-u) computed is 0.134, which is between cumulative of pillar 1 and 2, and hence t is inferred to happen between time pillar 1 and 2.

1.2.2 Correlated Default Time

Section above explains the generating random default times, however, in the valuation of a basket credit, correlated default times is to be considered, hence, it's required to generate correlated random variables. In this case, we aim to generate correlated uniform random variable, which is then converted to default times using the concept explained above.

Sklar's theorem states that any multivariate joint distribution can be written in terms of univariate marginal distribution functions and a copula which describes the dependence structure between the variables.

$F(x_1,\cdots,x_d) = C(F_1(x_1),\cdots,F_d(x_d))$

Where
$F(x_1,\cdots,x_d)$ is the CDF of the joint distribution function of parameters $(x_1,\cdots,x_d)$
$C(F_1(x_1),\cdots,F_d(x_d))$ is the copula function with univariate marginal CDF parameters
$F_1(x_1)$ hence is univariate marginal CDF.

Hence, joint CDF can be expressed in terms of marginal CDFs and the copula. Conversely, if we know the joint CDF F and the marginals CDF $F_1(x_1),\cdots,F_d(x_d))$, we can find the copula via

Given that marginal CDF are uniformly distributed, $F_1(x_1) = u_1$, conversely, $F^{-1}(u_1) = x_1$

$C(F_1(x_1),\cdots,F_d(x_d)) = C(u_1,\cdots,u_d)$
$F(x_1,\cdots,x_d) = F(F^{-1}(u_1),\cdots,F^{-1}(u_d))$

Hence,

$C(u_1,\cdots,u_d) = F(F^{-1}(u_1),\cdots,F^{-1}(u_d))$

Copula can then be viewed as a CDF of a uniform random vector, $C(u_1,\cdots,u_d)$.

For this task, we will study 2 joint CDF F, namely the Gaussian Copula and Student's t Copula.

1.2.2.1 Random Number Generation/Sampling

Sampling random number from a distribution such that the distribution (can be visualized by histogram) of the sample, resembles the shape of the known distribution.

Example: if X is a standard normal random variable, $X \sim N(0,1)$, when the random numbers generated to represent X, the distribution of X should reflect a standard normal distribution.

1.2.2.2 Gaussian Copula and Random Number Generation

$(u_1,\cdots,u_d) = ((u_1),\cdots , (u_d))$

Where :

is the inverse normal CDF
is the multivariate normal joint CDF
is the Gaussian Copula function CDF
is the correlation Matrix

Generating correlated normal random variable will be described in implementations section below.

At this point you have vector X, $(X_1,\cdots , X_d)$, which has the dependence structure, , and known the marginal distributions are known to be normal distribution.

Gaussian Copula, $(X_1,\cdots , X_d)$ requires marginal distribution to be uniform, this is easily achievable as we know the marginal CDF of X, an inverse marginal will transform X into correlated Uniform Distribution.

$(X_1,\cdots , X_d) = ((X_1),\cdots , (X_d)) = (u_1,\cdots ,u_d)$

It means that if X is transformed into uniform vector through marginal normal CDF, it should represent the input to a Gaussian copula with a dependence structure . Hence the distribution of $(u_1,\cdots ,u_d)$ should follow a Gaussian copula function.

$(u_1,\cdots ,u_d) \sim$

And hence, $(u_1,\cdots ,u_d)$ is a sampling from a Gaussian Copula.

1.2.2.3 Student’s t Copula and Random Number Generation

Similar to Gaussian Copula, Student’s t Copula is a joint probability density distribution function that requires one additional input on top of correlation, which is degree of Freedom.

Gaussian copula preserves the underlying distribution of the individual random variables but the joint distribution is like a multidimensional Gaussian. This naturally assigns very little weight to the tails. In reality, we find that within the financial markets, tail events occur much more frequently. So we would like a joint distribution which has fatter tails but preserves the same (bell shaped, non-skewed) characteristics of the Gaussian, hence t-Student copula should also be considered. (Fathi & Nader 2007)

The t-copula function is as below,

$$C_{\nu,\Sigma}(u_1 \ldots u_n) = \int_{-\infty}^{\Phi^{-1}(u_1)} \ldots \int_{-\infty}^{\Phi^{-1}(u_n)} K_{\nu,\Sigma}\left(1 + \frac{1}{\nu}(V-U)^T \Sigma^{-1}(V-U)\right)^{-\frac{\nu+n}{2}} dV_1 \ldots dV_2$$

Where v is the degree of freedom

To determine the degree of freedom, maximum likelihood estimation methods are used, and the degree of freedom that maximizes the log maximum likelihood should be the degree of freedom used in inputs above.

$$\nu^{CML} = \arg\max_{\nu \in (2,\infty]} \sum_{t=1}^{T} Log C^{Student}\left(\hat{u}_1^t, \hat{u}_2^t, \ldots, \hat{u}_N^t, R^{CML}, \nu\right).$$

*CML means canonical maximum likelihood method.

Generation of student t copula random variable will be described in implementation section below.

1.3 Correlation Matrix

Correlation Matrix is required for the generation of correlated standard normal variables in the section above, specifically the correlation of random variables with normal distributions.

A **Pearson correlation** is a measure of a linear association between 2 normally distributed random variables. As such, the data must be transformed/mapped to a normal distribution.

To do so, the analytical probability distribution function is required to come up with continuous cumulative distribution function, which can be then converted to a uniform distribution, and subsequently, using inverse normal CDF, be converted to a uniform random variable. The first step of the above can be done using kernel smoothing to find a pdf that fits the empirical data, which can be integrated to find the CDF.

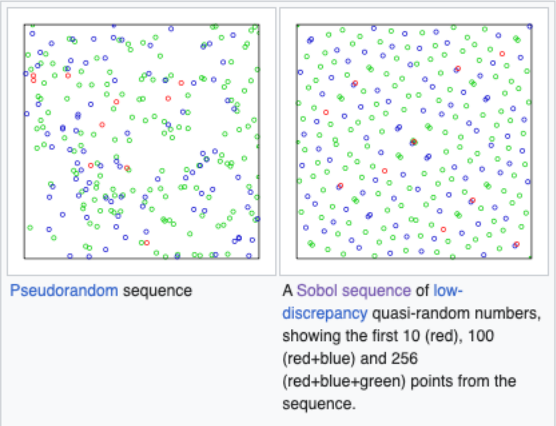
However, for this implementation, we rely on python function QuantileTransform to skip the steps above to directly produce a uniformly distributed data from an input of empirical data.

Given that the T-copula marginal distributions are student t distribution, which does not follow the definition of Pearson correlation, where normal distribution is assumed, the **Kendall correlation** is used in T-copula input.

Kendall's rank correlation provides a distribution free test of independence and a measure of the strength of dependence between two variables.

1.4 Random Number Generation

In finance, it is a practice to use low discrepancy (no large gaps, no clustering) sequences/algorithms to generate quasi random number to improve convergence. Many study has been done on this and below is a chart from Wikipedia on Quasi Monte Carlo Method.

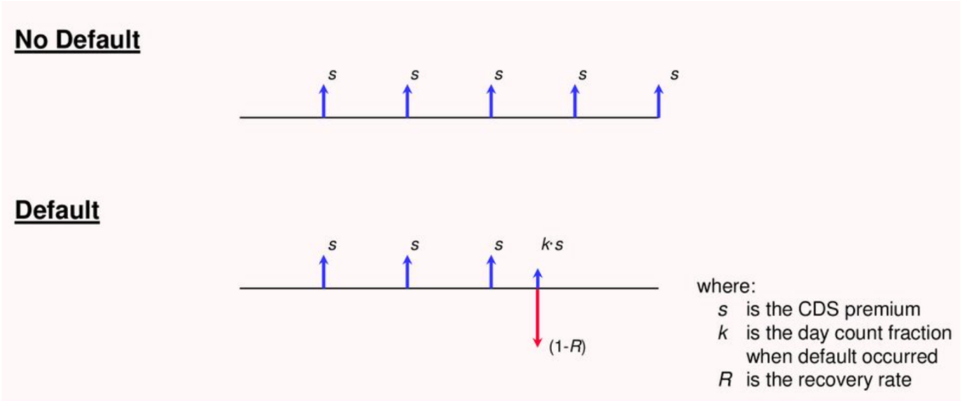


Pseudorandom sequence

A Sobol sequence of low-discrepancy quasi-random numbers, showing the first 10 (red), 100 (red+blue) and 256 (red+blue+green) points from the sequence.

1.5 Credit Curve Bootstrapping

1.5.1 CDS Pricing



CDS cash flow can be split into two legs, like an IRS, where the first cash flow leg (blue) is called the premium leg, and the contingent cash flow leg (red) is called the default leg.

The PV of the premium leg is calculated by:

$PV(Premium\ Leg) = spread\ *$

The PV of the default leg is calculated by:

PV(Default Leg) =

Spread is then calculated as

=

1.5.2 Credit Curve Bootstrapping

Given that

= =

We can iteratively bootstrap the credit curve, given the spread and discount factors, to obtain , which is then used to calculate the hazard rate.

Example :

For the first pillar,

=

= =

Rearrange to make the subject, closed form solution survival probability,

For the second pillar,

=

Rearranging to make the subject, like above, will yield the closed form for

After bootstrapping the survival probabilities up to the last pillar, the hazard rate can be inferred from relationship

P(arrival at t) =

For non-homogeneous Poisson Process, the equation above generalized to the above to accommodate for different hazard rate

P(arrival at t) =

Discretization of the exponent in the exponential brings us to

P(arrival at t) =

Given that is always one year. Then the formula can be simplified to

P(arrival at t) =

And hence, knowing , the hazard rate can be calculate using

And iteratively, the second hazard rate,

And so on for the remaining hazard rates.

1.6 Kth to Default Par Spread

Kth to Default, does not terminate until kth default happen. Furthermore, since it's referring to a basket of credit underlying, the CDS valuation above does not consider the correlation of default times between credit underlying.

Hence, the time to default of credit underlying sampled from the copula (joint probability), that will then produce a vector of default times, , where the kth to default timing can be obtained.

For this project, if default time is found to between t_1 and t_2 via the cumulative hazard rate, it is assumed default to happen in the middle of the period, $t_{1.5.}$

Example:

Assumed that the default time vector obtained is , the the 1st to default is at 2.5 years, while the second to default is at 4.5 years, whereas 3rd to default onwards, there was no default within a 5-year period.

Then, the par spread is then calculated using the CDS formula,

= =

Where N = 2.5 for 1st to Default Par Spread, and N = 4.5 for 2nd to Default Par Spread, and subsequently, will be 0 for 3rd to default onwards, resulting in a Par Spread of 0.

For 1st to default, year 1, n = 1 and

= 1 (No Default, Survival Probability 100%)
= 1 (No Default, Survival Probability 100%)

2nd year, n = 2

= 1. (No Default, Survival Probability 100%)

However, for 2.5 year,
= 0 (Default, Survival Probability 0%)

Hence,

=

0.5 because the last period, the premium is only paid for 0.5 of the year fraction, assuming 1st default time is 2.5 years.

For 2nd to default, 2nd to default happens on 4.5 years

= 1 (First Default, Survival Probability still 100%)
= 1
= 1
= 0

=

4/5 factor is introduced because after the first default, the notional is reduced by 1/5.

2 Implementation

2.1 Data Required

Credit Spread for Basket Underlying

FAANG group will be used as the underlying in the basket credit study

Tenor (Yrs)	AAPL	META	AMZN	NFLX	GOOG
0	0	0	0	0	0
1	10.18	13.79	6.19	8.03	41.26
2	14.52	18.14	8.89	10.94	58.37
3	21.86	23.7	12.61	14.41	72.44
4	26.67	29.56	18.53	19.13	85.53
5	31.58	35.71	24.31	26.1	113.8

Discount Curve

Rate Curve Construction is not the focus of this project, hence a simple discount curve is used to provide Discount Factor for computation Purposes

Pillars	Time (Year Frac)	Discount Factor
1W	0.019178082	0.999424823
1M	0.082191781	0.997537284
3M	0.246575342	0.992630032
6M	0.5	0.98511194
1Y	1	0.970445534
2Y	2	0.941764534
3Y	3	0.913931185
5Y	5	0.860707976
10Y	10	0.740818221

Historical Stock Price

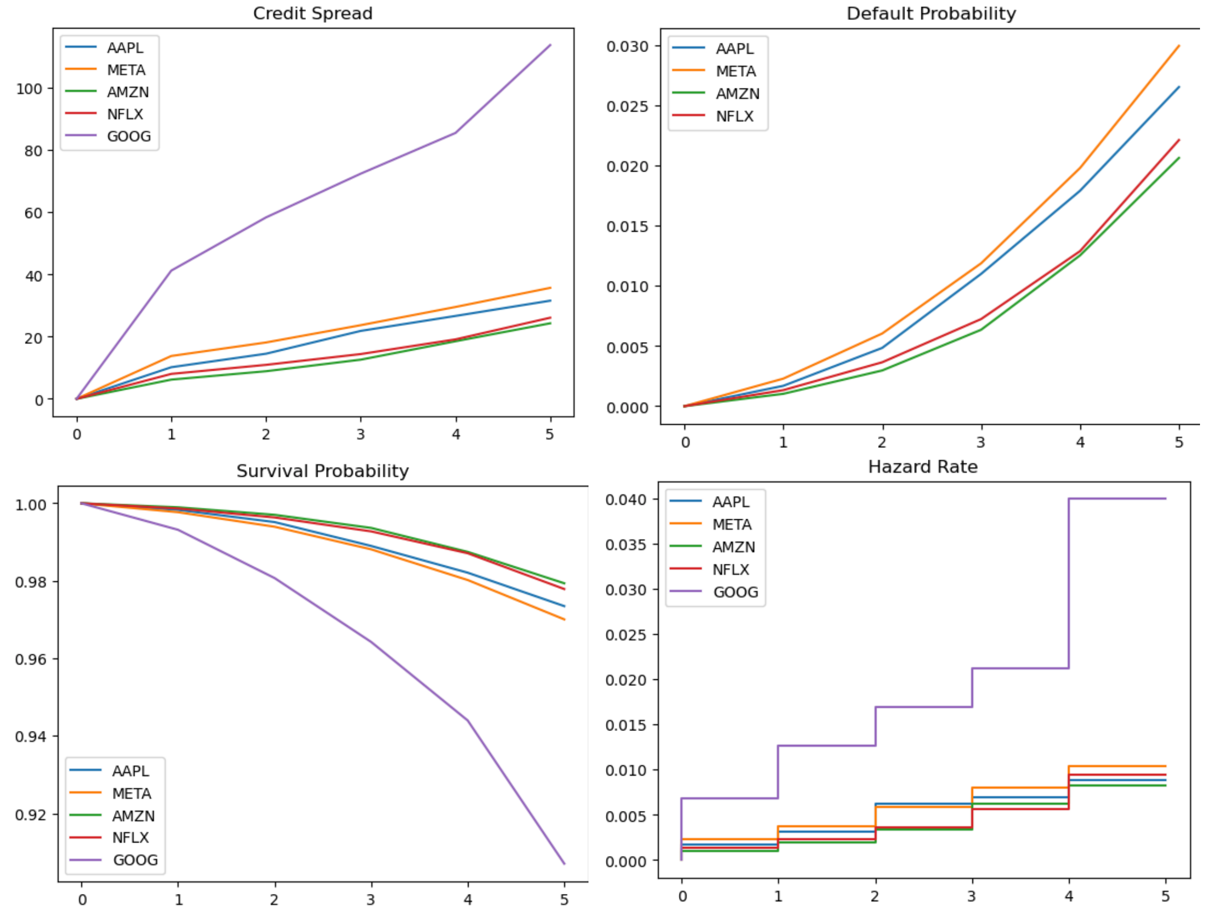
Historical Data is extracted using Yahoo Finance for the Past 5 Years, in order to calculate the log return.

Code implementation in Utilities.py function below

```
def getHistLogReturn(tickers, period):
```

2.2 Credit Curve Bootstrapping

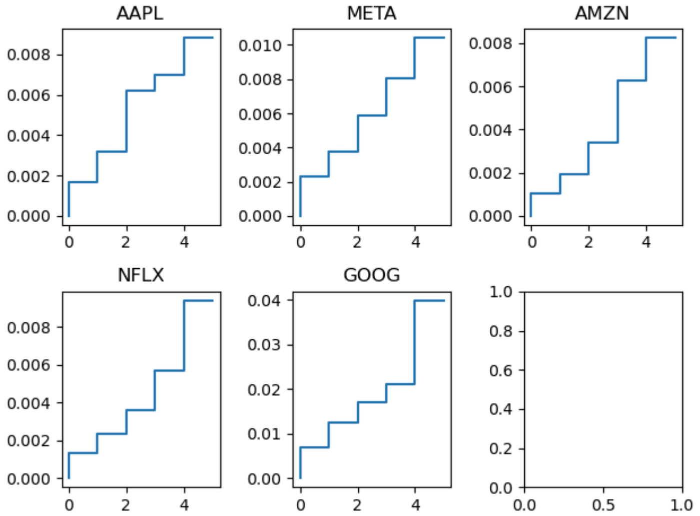
Below are the credit curve data from credit curve bootstrapping. Graph below display the credit spread trend, which is expected to be upward sloping as longer time requires longer credit premium, which is captured in longer tenure credit spread.



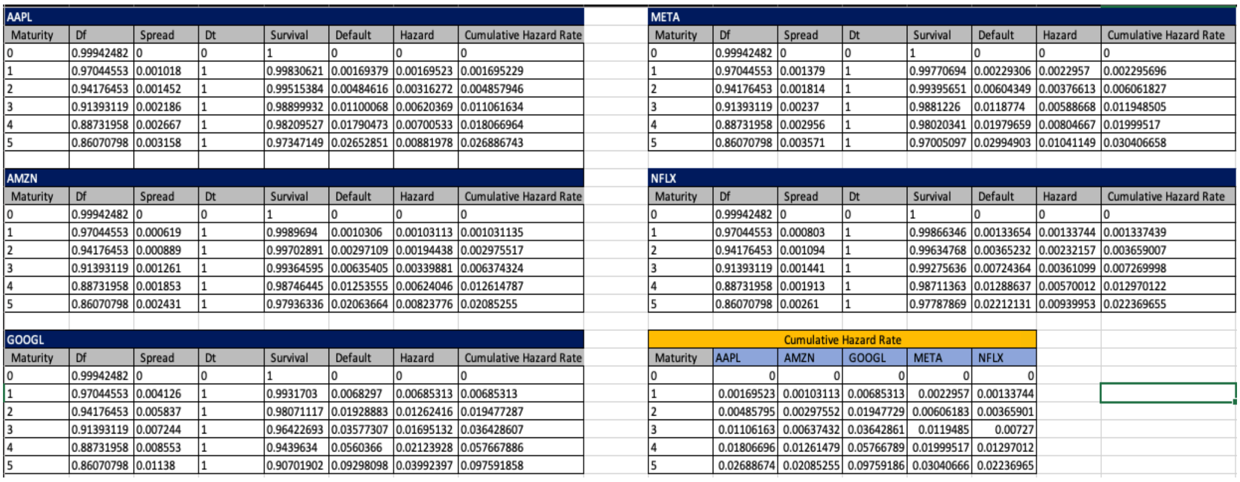
.

It’s expected for default probabilty and survival probability to be upward sloping

Individual hazard rate shown below.



Results of Credit Curve Bootstrapping



AAPL

Maturity	Df	Spread	Dt	Survival	Default	Hazard	Cumulative Hazard Rate
0	0.99942482	0	0	1	0	0	0
1	0.97044553	0.001018	1	0.99830621	0.00169379	0.00169523	0.001695229
2	0.94176453	0.001452	1	0.99515384	0.00484616	0.00316272	0.004857946
3	0.91393119	0.002186	1	0.98899932	0.01100068	0.00620369	0.011061634
4	0.88731958	0.002667	1	0.98209527	0.01790473	0.00700533	0.018066964
5	0.86070798	0.003158	1	0.97347149	0.02652851	0.00881978	0.026886743

META

Maturity	Df	Spread	Dt	Survival	Default	Hazard	Cumulative Hazard Rate
0	0.99942482	0	0	1	0	0	0
1	0.97044553	0.001379	1	0.99770694	0.00229306	0.0022957	0.002295696
2	0.94176453	0.001814	1	0.99395651	0.00604349	0.00376613	0.006061827
3	0.91393119	0.00237	1	0.9881226	0.0118774	0.00588668	0.011948505
4	0.88731958	0.002956	1	0.98020341	0.01979659	0.00804667	0.01999517
5	0.86070798	0.003571	1	0.97005097	0.02994903	0.01041149	0.030406658

AMZN

Maturity	Df	Spread	Dt	Survival	Default	Hazard	Cumulative Hazard Rate
0	0.99942482	0	0	1	0	0	0
1	0.97044553	0.000619	1	0.9989694	0.0010306	0.00103113	0.001031135
2	0.94176453	0.000889	1	0.99702891	0.00297109	0.00194438	0.002975517
3	0.91393119	0.001261	1	0.99364595	0.00635405	0.00339881	0.006374324
4	0.88731958	0.001853	1	0.98746445	0.01253555	0.00624046	0.012614787
5	0.86070798	0.002431	1	0.97936336	0.02063664	0.00823776	0.02085255

NFLX

Maturity	Df	Spread	Dt	Survival	Default	Hazard	Cumulative Hazard Rate
0	0.99942482	0	0	1	0	0	0
1	0.97044553	0.000803	1	0.99866346	0.00133654	0.00133744	0.001337439
2	0.94176453	0.001094	1	0.99634768	0.00365232	0.00232157	0.003659007
3	0.91393119	0.001441	1	0.99275636	0.00724364	0.00361099	0.007269998
4	0.88731958	0.001913	1	0.98711363	0.01288637	0.00570012	0.012970122
5	0.86070798	0.00261	1	0.97787869	0.02212131	0.00939953	0.022369655

GOOGL

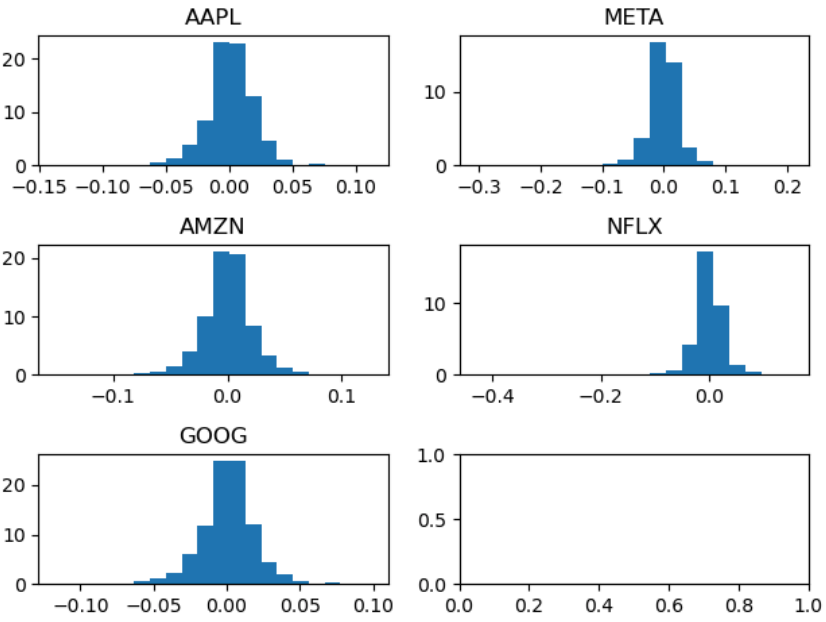
Maturity	Df	Spread	Dt	Survival	Default	Hazard	Cumulative Hazard Rate
0	0.99942482	0	0	1	0	0	0
1	0.97044553	0.004126	1	0.9931703	0.0068297	0.00685313	0.00685313
2	0.94176453	0.005837	1	0.98071117	0.01928883	0.01262416	0.019477287
3	0.91393119	0.007244	1	0.96422693	0.03577307	0.01695132	0.036428607
4	0.88731958	0.008553	1	0.9439634	0.0560366	0.02123928	0.057667886
5	0.86070798	0.01138	1	0.90701902	0.09298098	0.03992397	0.097591858

Cumulative Hazard Rate

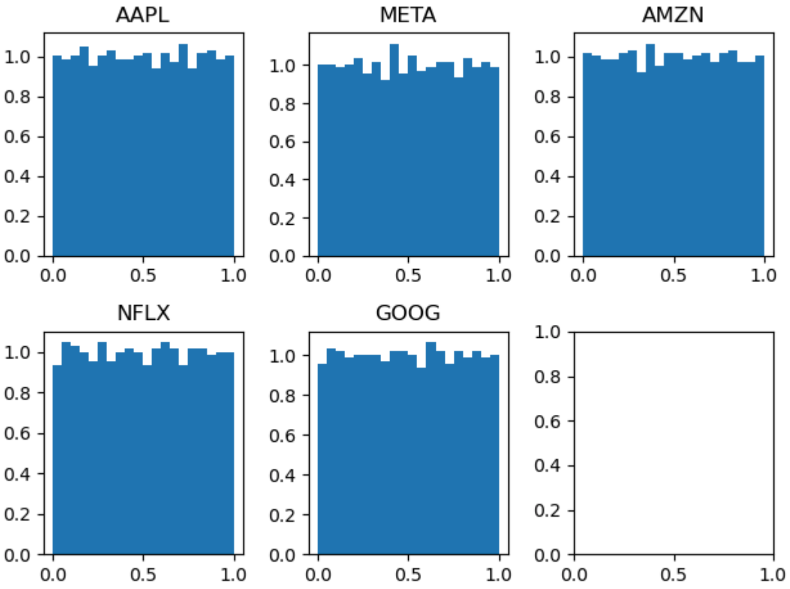
Maturity	AAPL	AMZN	GOOGL	META	NFLX
0	0	0	0	0	0
1	0.00169523	0.00103113	0.00685313	0.0022957	0.00133744
2	0.00485795	0.00297552	0.01947729	0.00606183	0.00365901
3	0.01106163	0.00637432	0.03642861	0.0119485	0.00727
4	0.01806696	0.01261479	0.05766789	0.01999517	0.01297012
5	0.02688674	0.02085255	0.09759186	0.03040666	0.02236965

2.3 Pseudo Normalized Data & Correlation Matrix

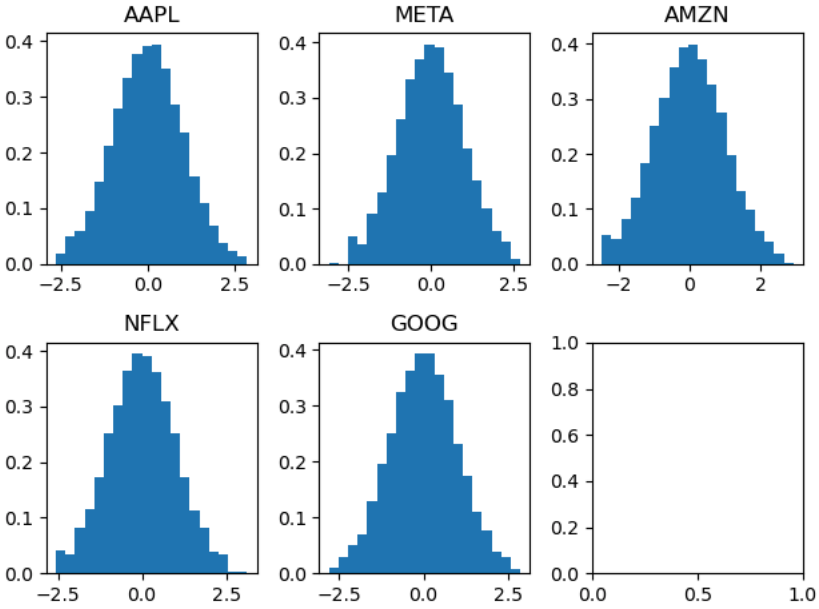
Log Returns empirical distribution generated from stock price extraction using yahoo finance.



Using sklearn.preprocessing.Quantiletransform which converts empirical distributions to uniform distributions. Study of N_Quantile parameter in Appendix, N_quantile used = 100

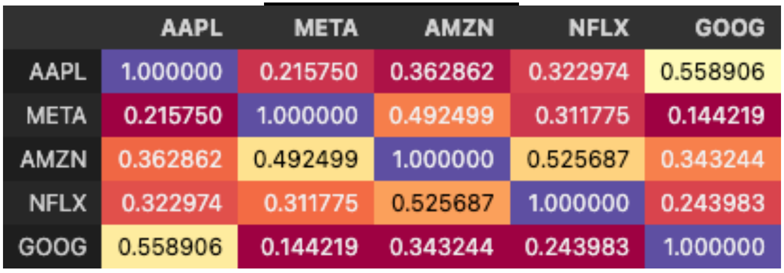


Inverse Norm CDF is used to transform pseudo uniform data into pseudo normal data.



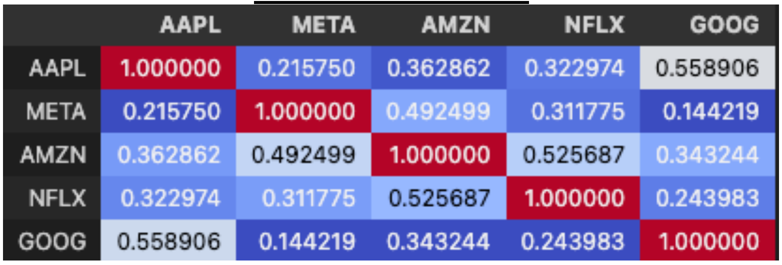
Correlation Matrix Generated from Pseudo Normal Data.

Pearson Correlation



	AAPL	META	AMZN	NFLX	GOOG
AAPL	1.000000	0.215750	0.362862	0.322974	0.558906
META	0.215750	1.000000	0.492499	0.311775	0.144219
AMZN	0.362862	0.492499	1.000000	0.525687	0.343244
NFLX	0.322974	0.311775	0.525687	1.000000	0.243983
GOOG	0.558906	0.144219	0.343244	0.243983	1.000000

Kendall's Correlation



	AAPL	META	AMZN	NFLX	GOOG
AAPL	1.000000	0.215750	0.362862	0.322974	0.558906
META	0.215750	1.000000	0.492499	0.311775	0.144219
AMZN	0.362862	0.492499	1.000000	0.525687	0.343244
NFLX	0.322974	0.311775	0.525687	1.000000	0.243983
GOOG	0.558906	0.144219	0.343244	0.243983	1.000000

2.4 Correlated Random Number Generation

Reference to Monte Carlo Methods In Finance, Peter Jackel, steps below are used for random number generation from Gaussian Copula and Student T Copula.

Gaussian Copula Random Number (Pg 46)

1. Find the correlation's lower triangular with Cholesky Decomposition such that $R = A \, . \, A^T$
2. Generate a vector of n uncorrelated standard normal variables, $Z = (z_1, z_1, \ldots z_n)$
3. Compute X = A.Z, where X is your correlated standard normal variables
4. Use inverse norm CDF to transform X into uniform distribution.

Gaussian Copula Random Number (Pg 49)

1. Find the correlation's lower triangular with Cholesky Decomposition such that $R = A \, . \, A^T$
2. Generate a vector of n uncorrelated standard normal variables, $Z = (z_1, z_1, \ldots z_n)$
3. Compute X = A.Z, where X is your correlated standard normal variables
4. Generate independent standard normal variables for $S = (z_1, z_1, \ldots z_n)$, where n is the Degree of Freedom. Take the Sum of Square of the vector S.
5. Compute T = to generate correlated student t variables
6. Use inverse student t CDF to transform T into uniform distribution.

Degree of Freedom used is 4, calibration study can be found in Appendix.

2.5 Correlated Default Time

To determine the correlated default time vector of each simulation,

1. Compute log(1-u) where u is the uniform correlated vector input, u = (u_1, u_1,... u_n)
2. Compute kth to default time vector by comparing u_1 to u_n to their respective cumulative hazard rate values.

Example:

u_1 is compared against AAPL cumulative hazard rate, u_2 is compared against AMZN cumulative hazard rate, and so on.

2.6 Basket Par Spread Calculation

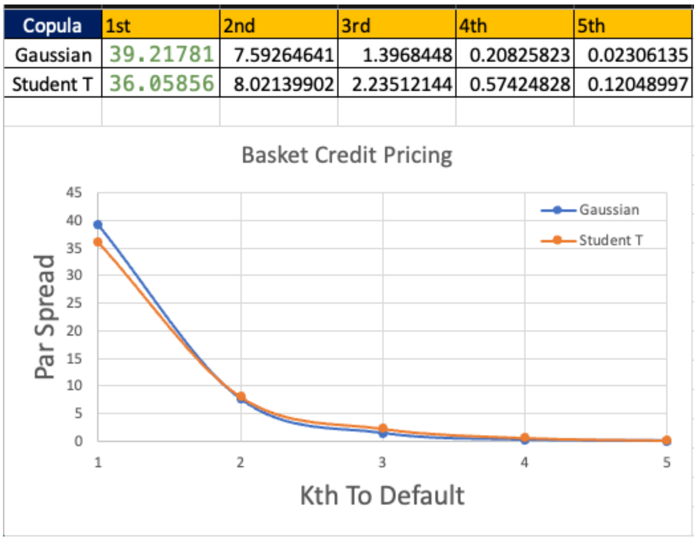
Once the kth to default time vector is obtained, use the basket credit pricing formulas described in section above to compute par spread for each simulation, and the final par spread is the average of premium and default leg.

2.7 Results

Using 10,000 simulations, the result generated using Gaussian Copula and Student T below:

Results shows that par spread is lower when kth to default increase, which is expected due to the lower probability of k defaults happening in a reference basket, and compared to the probability of k-1 defaults happening.

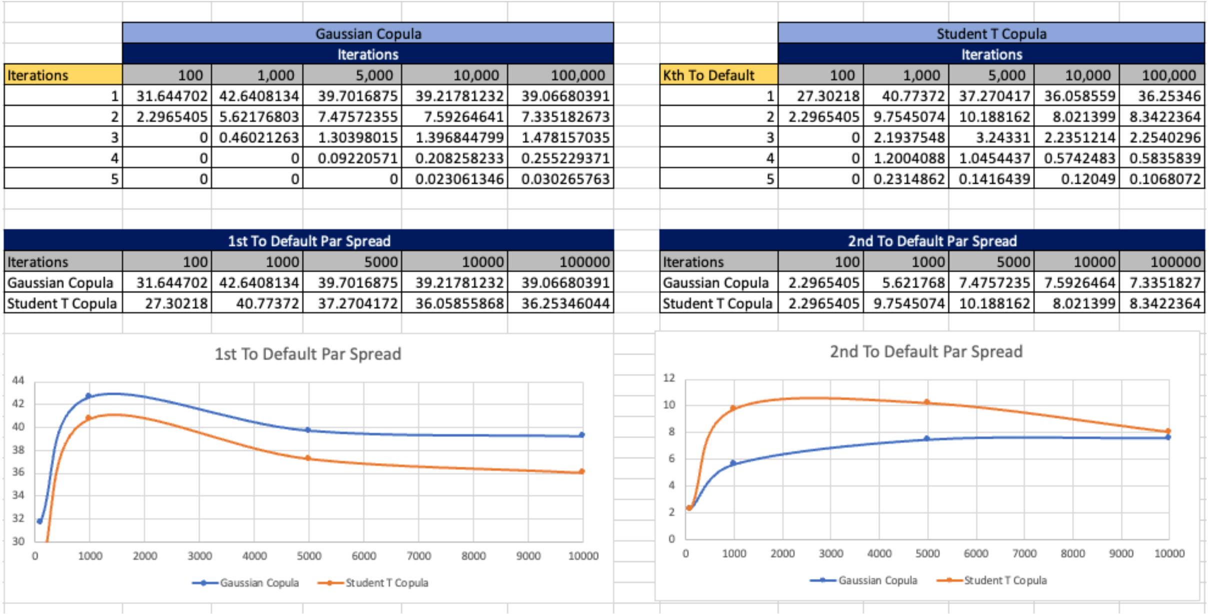
Also, it’s observed that par spread calculated from student T is higher when k is higher, which can be explained by T-student distribution capturing tail dependence better than Gaussian Copula).



Copula	1st	2nd	3rd	4th	5th
Gaussian	39.21781	7.59264641	1.3968448	0.20825823	0.02306135
Student T	36.05856	8.02139902	2.23512144	0.57424828	0.12048997

3 Analysis

3.1 Number of Simulation – Convergence Study



	Gaussian Copula				
	Iterations				
Iterations	100	1,000	5,000	10,000	100,000
1	31.644702	42.6408134	39.7016875	39.21781232	39.06680391
2	2.2965405	5.62176803	7.47572355	7.59264641	7.335182673
3	0	0.46021263	1.30398015	1.396844799	1.478157035
4	0	0	0.09220571	0.208258233	0.255229371
5	0	0	0	0.023061346	0.030265763

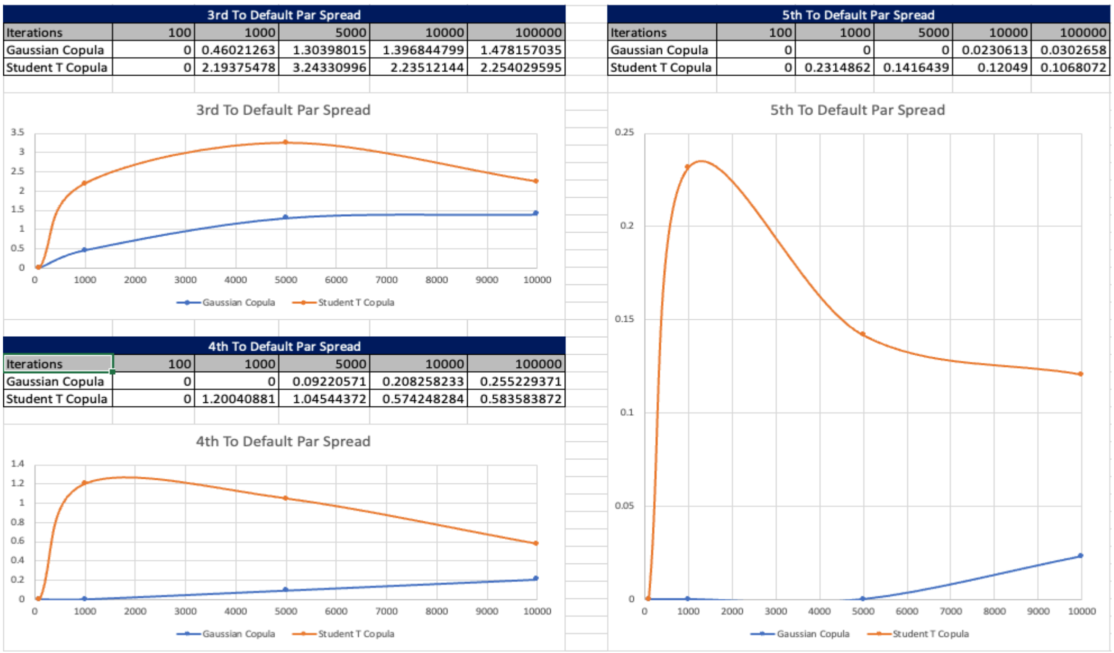
	Student T Copula				
	Iterations				
Kth To Default	100	1,000	5,000	10,000	100,000
1	27.30218	40.77372	37.270417	36.058559	36.25346
2	2.2965405	9.7545074	10.188162	8.021399	8.3422364
3	0	2.1937548	3.24331	2.2351214	2.2540296
4	0	1.2004088	1.0454437	0.5742483	0.5835839
5	0	0.2314862	0.1416439	0.12049	0.1068072

1st To Default Par Spread					
Iterations	100	1000	5000	10000	100000
Gaussian Copula	31.644702	42.6408134	39.7016875	39.21781232	39.06680391
Student T Copula	27.30218	40.77372	37.2704172	36.05855868	36.25346044

2nd To Default Par Spread					
Iterations	100	1000	5000	10000	100000
Gaussian Copula	2.2965405	5.621768	7.4757235	7.5926464	7.3351827
Student T Copula	2.2965405	9.7545074	10.188162	8.021399	8.3422364

Testing the convergence with different number of simulation, from studying the 1st to default and 2nd to default par spread calculation, it can be noted that increase in simulation will result in a convergence to a value.

From the graph, it can be observed that at 10,000 simulation, the increase in convergence is marginal, hence, analysis moving forward will use 10,000 simulation. Results for other K are also displayed below, the trend of marginal increases after 10,000 simulation is also apparent.

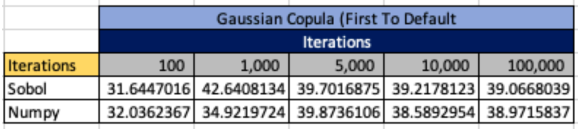


3rd To Default Par Spread					
Iterations	100	1000	5000	10000	100000
Gaussian Copula	0	0.46021263	1.30398015	1.396844799	1.478157035
Student T Copula	0	2.19375478	3.24330996	2.23512144	2.254029595

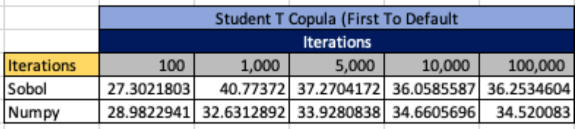
5th To Default Par Spread					
Iterations	100	1000	5000	10000	100000
Gaussian Copula	0	0	0	0.0230613	0.0302658
Student T Copula	0	0.2314862	0.1416439	0.12049	0.1068072

4th To Default Par Spread					
Iterations	100	1000	5000	10000	100000
Gaussian Copula	0	0	0.09220571	0.208258233	0.255229371
Student T Copula	0	1.20040881	1.04544372	0.574248284	0.583583872

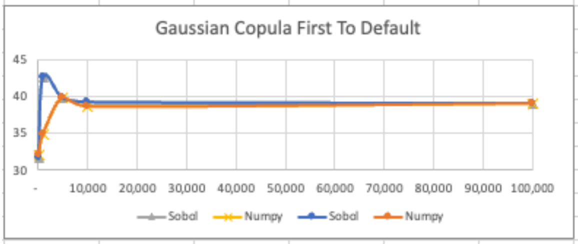
3.2 Random Number Generator – Convergence Study

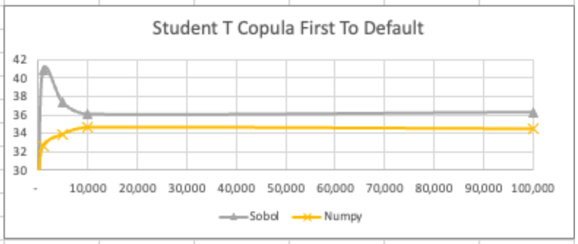


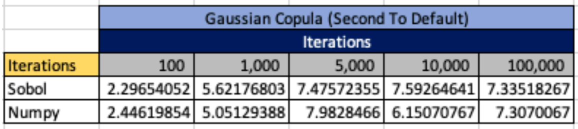
	Gaussian Copula (First To Default				
	Iterations				
Iterations	100	1,000	5,000	10,000	100,000
Sobol	31.6447016	42.6408134	39.7016875	39.2178123	39.0668039
Numpy	32.0362367	34.9219724	39.8736106	38.5892954	38.9715837



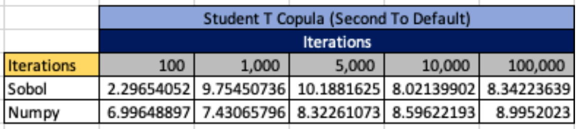
	Student T Copula (First To Default				
	Iterations				
Iterations	100	1,000	5,000	10,000	100,000
Sobol	27.3021803	40.77372	37.2704172	36.0585587	36.2534604
Numpy	28.9822941	32.6312892	33.9280838	34.6605696	34.520083



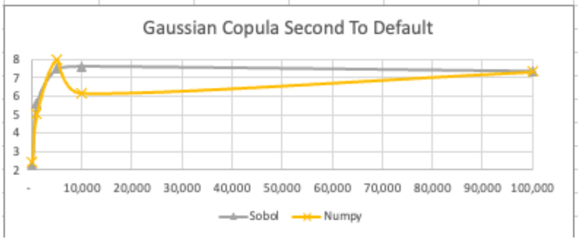


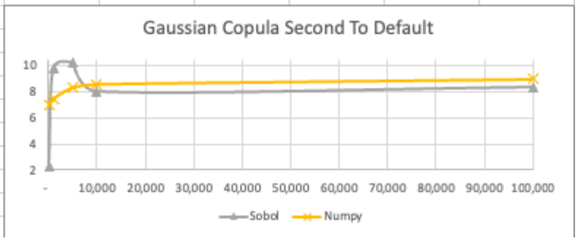


	Gaussian Copula (Second To Default)				
	Iterations				
Iterations	100	1,000	5,000	10,000	100,000
Sobol	2.29654052	5.62176803	7.47572355	7.59264641	7.33518267
Numpy	2.44619854	5.05129388	7.9828466	6.15070767	7.3070067



	Student T Copula (Second To Default)				
	Iterations				
Iterations	100	1,000	5,000	10,000	100,000
Sobol	2.29654052	9.75450736	10.1881625	8.02139902	8.34223639
Numpy	6.99648897	7.43065796	8.32261073	8.59622193	8.9952023



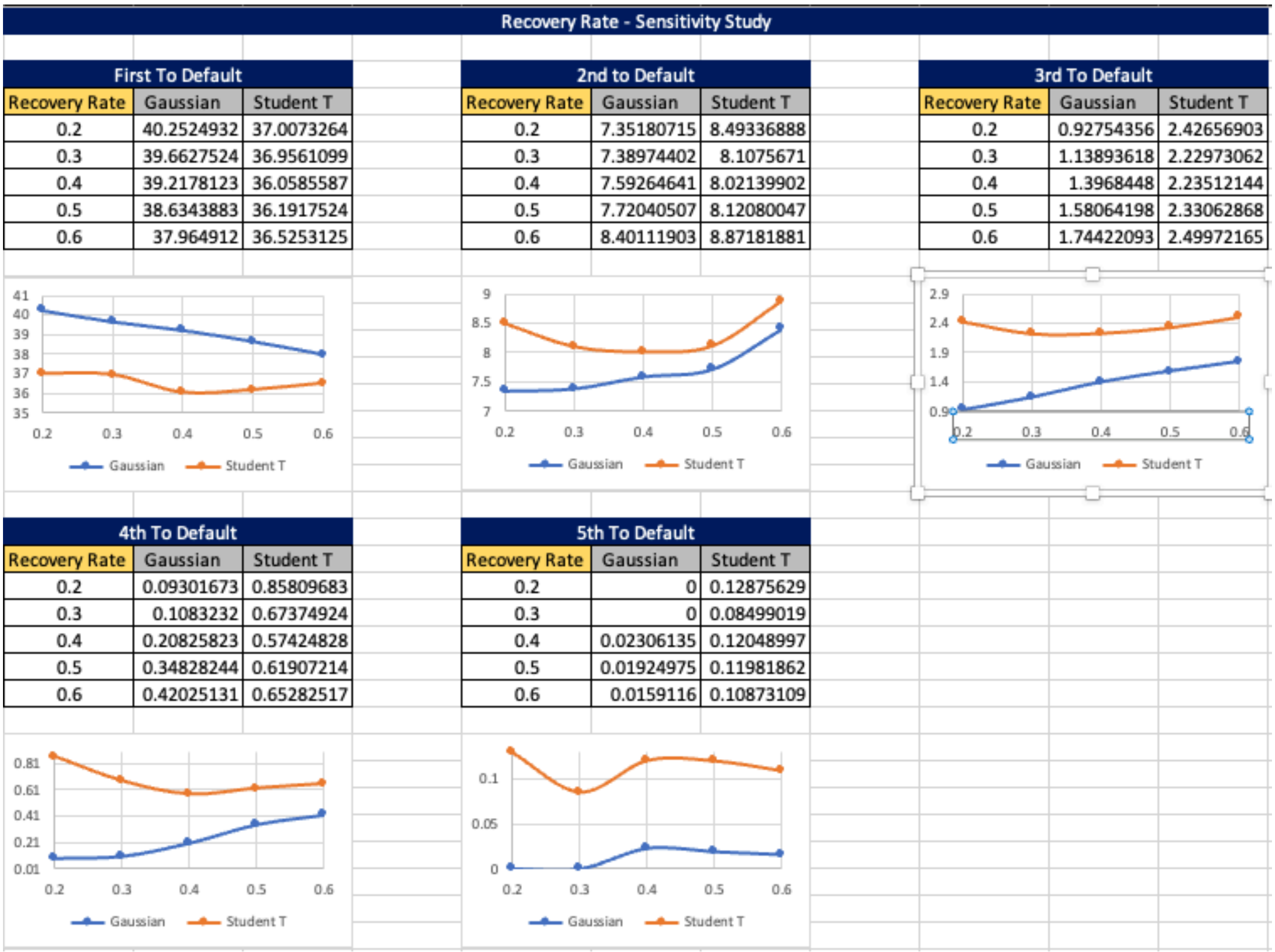


To study random number generator, the np.random.normal and sobol sequence generators are used to generate normal random values.

From the graphs, generally, Sobol will have an initial spike at lower level of iterations, this is due to Sobol is not encouraged for lower dimension random number generations.

However, it can be seen that Sobol random number provides convergence faster, which is most apparent in the Gaussian Copula second to default graph (Bottom Left).

3.3 Recovery Rate – Sensitivity Study



Recovery Rate - Sensitivity Study

First To Default

Recovery Rate	Gaussian	Student T
0.2	40.2524932	37.0073264
0.3	39.6627524	36.9561099
0.4	39.2178123	36.0585587
0.5	38.6343883	36.1917524
0.6	37.964912	36.5253125

2nd to Default

Recovery Rate	Gaussian	Student T
0.2	7.35180715	8.49336888
0.3	7.38974402	8.1075671
0.4	7.59264641	8.02139902
0.5	7.72040507	8.12080047
0.6	8.40111903	8.87181881

3rd To Default

Recovery Rate	Gaussian	Student T
0.2	0.92754356	2.42656903
0.3	1.13893618	2.22973062
0.4	1.3968448	2.23512144
0.5	1.58064198	2.33062868
0.6	1.74422093	2.49972165

4th To Default

Recovery Rate	Gaussian	Student T
0.2	0.09301673	0.85809683
0.3	0.1083232	0.67374924
0.4	0.20825823	0.57424828
0.5	0.34828244	0.61907214
0.6	0.42025131	0.65282517

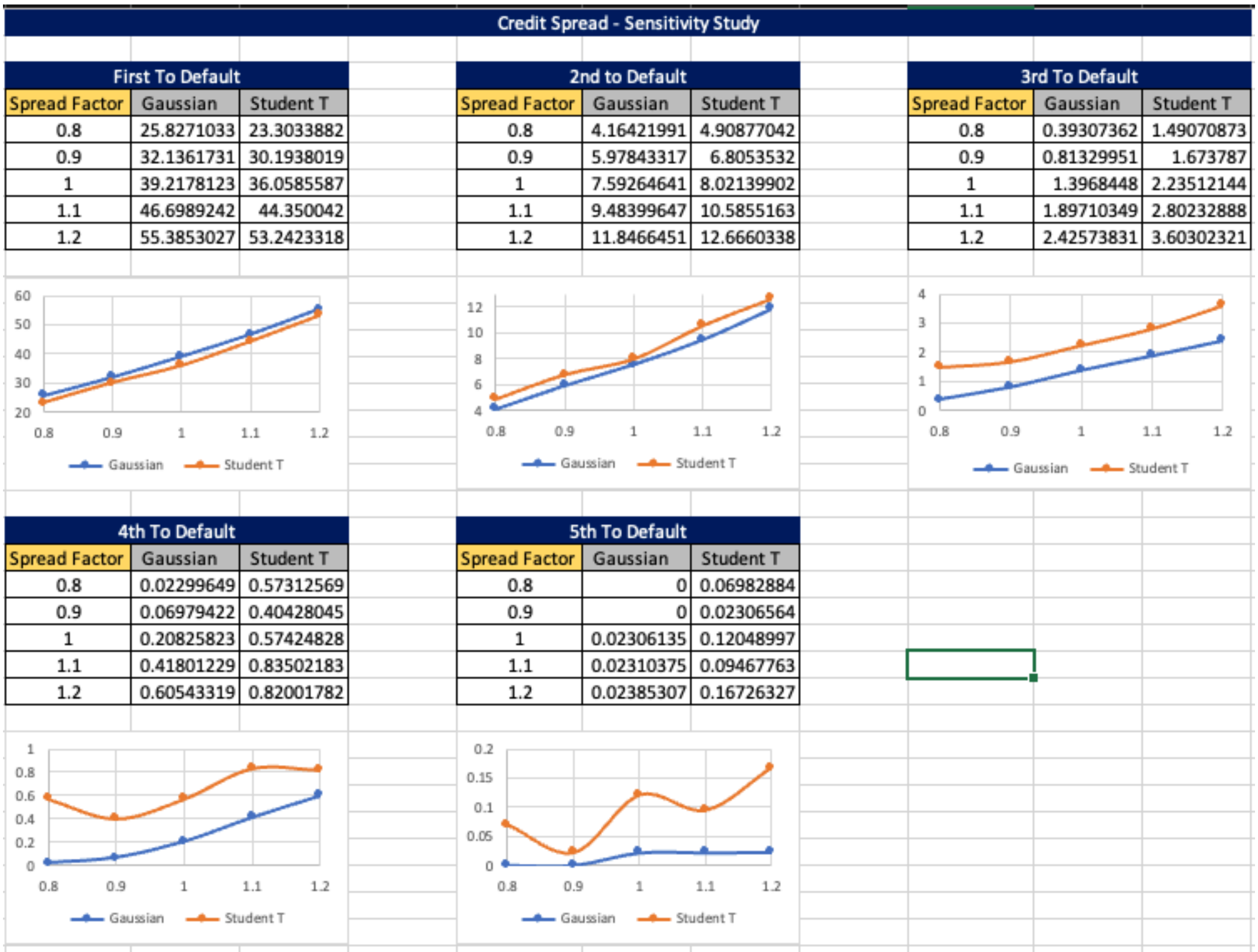
5th To Default

Recovery Rate	Gaussian	Student T
0.2	0	0.12875629
0.3	0	0.08499019
0.4	0.02306135	0.12048997
0.5	0.01924975	0.11981862
0.6	0.0159116	0.10873109

Intuitively, recovery rate determines the default leg pay-out amount, and hence, a lower recovery rate will command a higher par spread, given that the default leg on credit events will be higher. This is apparent in first to default graph (top right).

However, recovery rate will also impact the hazard rate from the bootstrapping exercise, and hence, as K increases, the graph will generally have still have higher par spread during low recovery, however, there’s also concave graphs, which may be due to the recovery rate impact on hazard rate bootstrapping.

3.4 Credit Spread – Sensitivity Study



Credit Spread - Sensitivity Study

First To Default

Spread Factor	Gaussian	Student T
0.8	25.8271033	23.3033882
0.9	32.1361731	30.1938019
1	39.2178123	36.0585587
1.1	46.6989242	44.350042
1.2	55.3853027	53.2423318

2nd to Default

Spread Factor	Gaussian	Student T
0.8	4.16421991	4.90877042
0.9	5.97843317	6.8053532
1	7.59264641	8.02139902
1.1	9.48399647	10.5855163
1.2	11.8466451	12.6660338

3rd To Default

Spread Factor	Gaussian	Student T
0.8	0.39307362	1.49070873
0.9	0.81329951	1.673787
1	1.3968448	2.23512144
1.1	1.89710349	2.80232888
1.2	2.42573831	3.60302321

4th To Default

Spread Factor	Gaussian	Student T
0.8	0.02299649	0.57312569
0.9	0.06979422	0.40428045
1	0.20825823	0.57424828
1.1	0.41801229	0.83502183
1.2	0.60543319	0.82001782

5th To Default

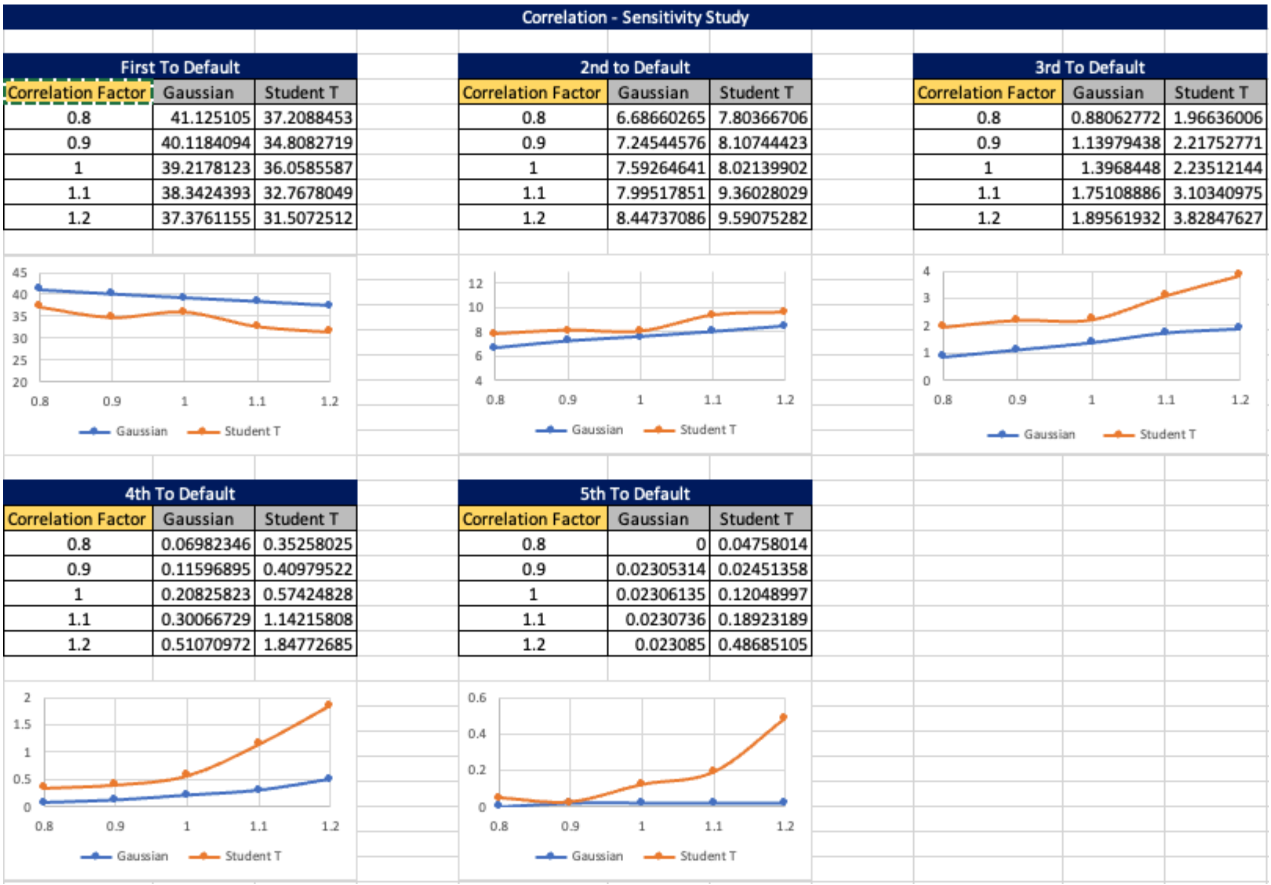
Spread Factor	Gaussian	Student T
0.8	0	0.06982884
0.9	0	0.02306564
1	0.02306135	0.12048997
1.1	0.02310375	0.09467763
1.2	0.02385307	0.16726327

The Credit Spread are shocked by a spread factor, where 1 means the values are multiplied by 1, which is the same as original credit spreads; and 1.1 means that each credit spread are multiplied by a factor of 1.1, equivalent to a 10% upward shock, and so on.

The shock applied varies from a 20% downward (0.8 spread factor), up to a 20% upward (1.2 spread factor) shock.

It can be seen that higher credit spread generally means that the par spread is higher, this is observed most apparently in K = 1 to K = 3 par spreads, however, at higher K values, the par spread will not necessary be sensitive to credit spreads of underlying, but more sensitive to correlation between the underlying.

3.5 Correlation – Sensitivity Study



Correlation - Sensitivity Study

First To Default

Correlation Factor	Gaussian	Student T
0.8	41.125105	37.2088453
0.9	40.1184094	34.8082719
1	39.2178123	36.0585587
1.1	38.3424393	32.7678049
1.2	37.3761155	31.5072512

2nd to Default

Correlation Factor	Gaussian	Student T
0.8	6.68660265	7.80366706
0.9	7.24544576	8.10744423
1	7.59264641	8.02139902
1.1	7.99517851	9.36028029
1.2	8.44737086	9.59075282

3rd To Default

Correlation Factor	Gaussian	Student T
0.8	0.88062772	1.96636006
0.9	1.13979438	2.21752771
1	1.3968448	2.23512144
1.1	1.75108886	3.10340975
1.2	1.89561932	3.82847627

4th To Default

Correlation Factor	Gaussian	Student T
0.8	0.06982346	0.35258025
0.9	0.11596895	0.40979522
1	0.20825823	0.57424828
1.1	0.30066729	1.14215808
1.2	0.51070972	1.84772685

5th To Default

Correlation Factor	Gaussian	Student T
0.8	0	0.04758014
0.9	0.02305314	0.02451358
1	0.02306135	0.12048997
1.1	0.0230736	0.18923189
1.2	0.023085	0.48685105

The correlation matrix are shocked by a correlation factor, where by 0.8 spread factor corresponds to a 20% downward shock (similar to spread factor concept above).

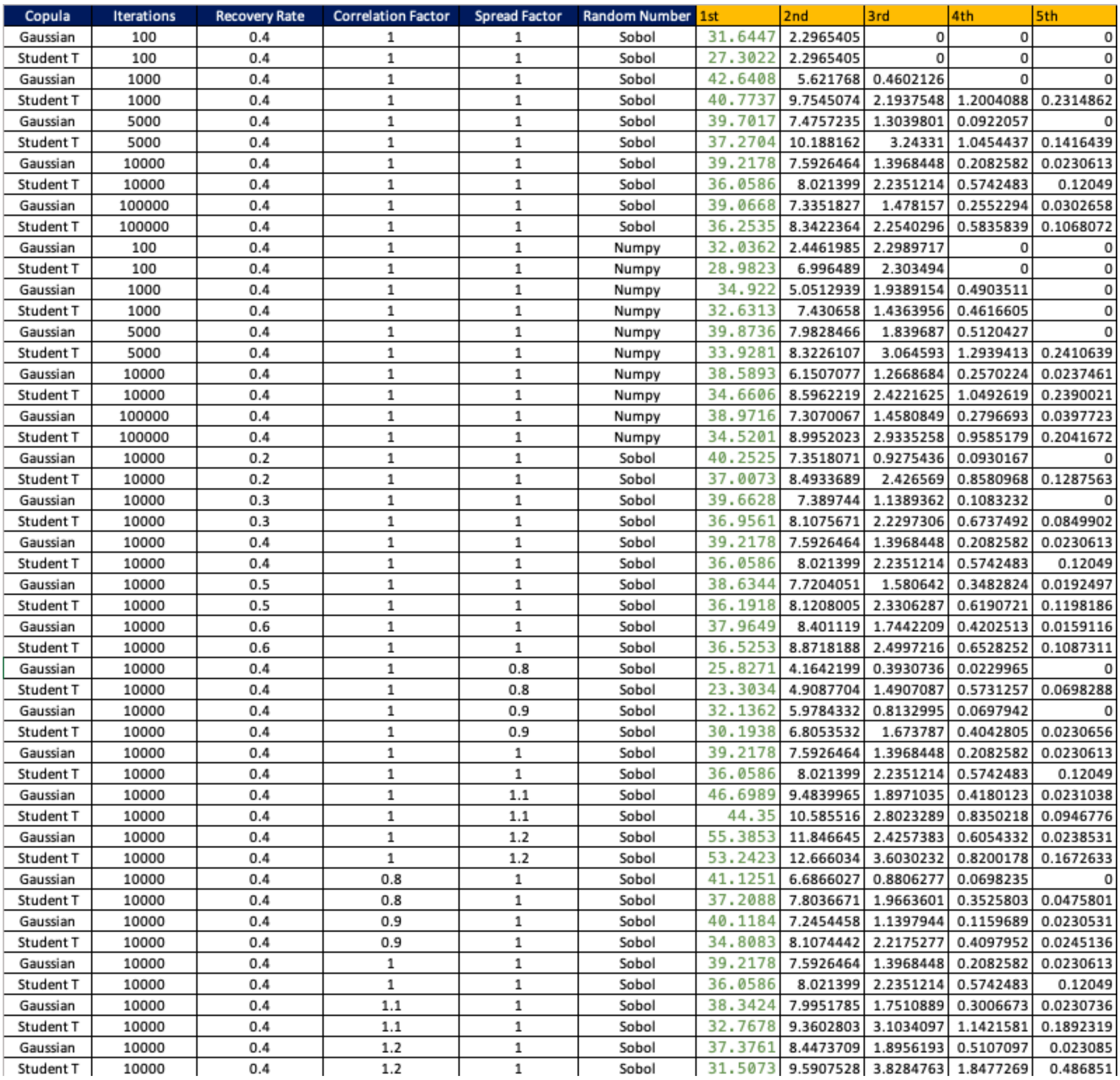
The shock applied varies from a 20% downward (0.8 spread factor), up to a 20% upward (1.2 spread factor) shock.

For K = 1, as the pay-out is determined by the first default, Par spread will not be highly sensitive to correlation matrix shocks, as also shown by the graphs.

However, as we move up the K value, it can be observed that the par spread calculated is higher for higher correlation values (at higher upward shocks).

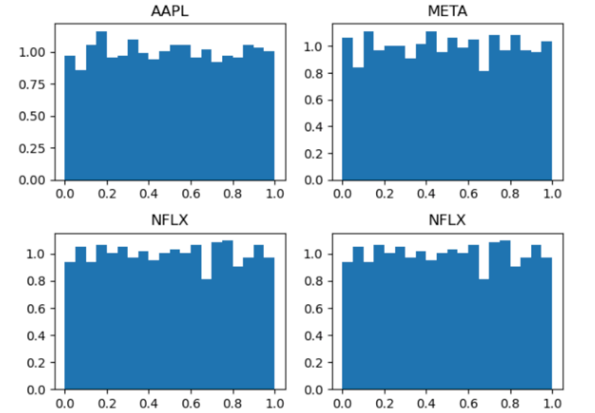
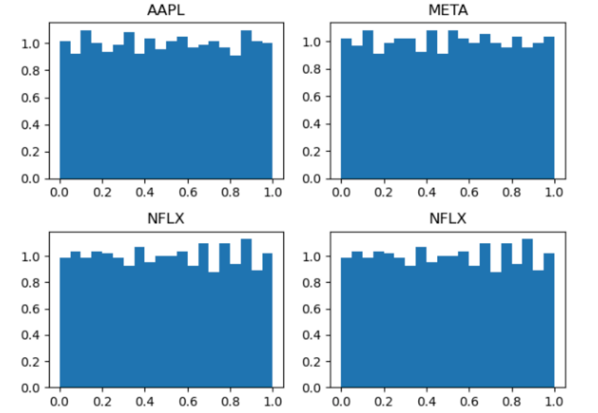
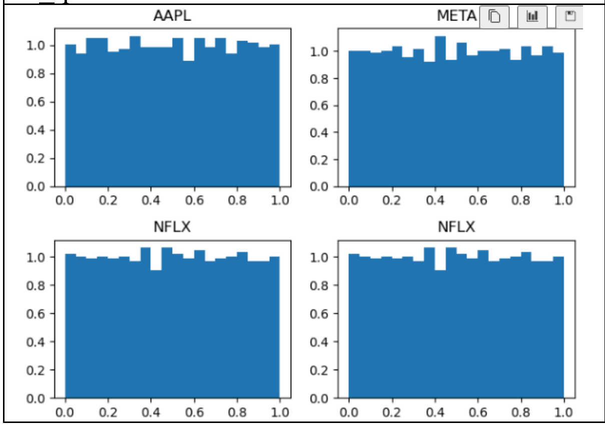
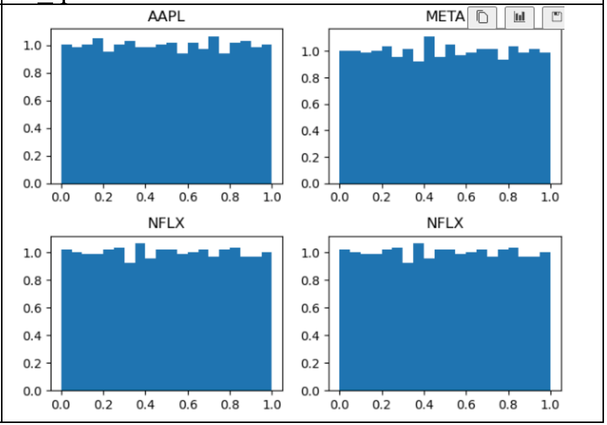
4 Appendix

4.1 Compiled Results Run

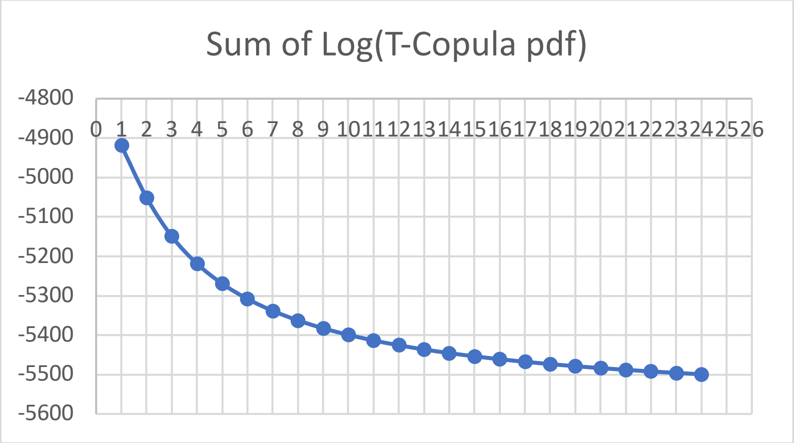


Copula	Iterations	Recovery Rate	Correlation Factor	Spread Factor	Random Number	1st	2nd	3rd	4th	5th
Gaussian	100	0.4	1	1	Sobol	31.6447	2.2965405	0	0	0
Student T	100	0.4	1	1	Sobol	27.3022	2.2965405	0	0	0
Gaussian	1000	0.4	1	1	Sobol	42.6408	5.621768	0.4602126	0	0
Student T	1000	0.4	1	1	Sobol	40.7737	9.7545074	2.1937548	1.2004088	0.2314862
Gaussian	5000	0.4	1	1	Sobol	39.7017	7.4757235	1.3039801	0.0922057	0
Student T	5000	0.4	1	1	Sobol	37.2704	10.188162	3.24331	1.0454437	0.1416439
Gaussian	10000	0.4	1	1	Sobol	39.2178	7.5926464	1.3968448	0.2082582	0.0230613
Student T	10000	0.4	1	1	Sobol	36.0586	8.021399	2.2351214	0.5742483	0.12049
Gaussian	100000	0.4	1	1	Sobol	39.0668	7.3351827	1.478157	0.2552294	0.0302658
Student T	100000	0.4	1	1	Sobol	36.2535	8.3422364	2.2540296	0.5835839	0.1068072
Gaussian	100	0.4	1	1	Numpy	32.0362	2.4461985	2.2989717	0	0
Student T	100	0.4	1	1	Numpy	28.9823	6.996489	2.303494	0	0
Gaussian	1000	0.4	1	1	Numpy	34.922	5.0512939	1.9389154	0.4903511	0
Student T	1000	0.4	1	1	Numpy	32.6313	7.430658	1.4363956	0.4616605	0
Gaussian	5000	0.4	1	1	Numpy	39.8736	7.9828466	1.839687	0.5120427	0
Student T	5000	0.4	1	1	Numpy	33.9281	8.3226107	3.064593	1.2939413	0.2410639
Gaussian	10000	0.4	1	1	Numpy	38.5893	6.1507077	1.2668684	0.2570224	0.0237461
Student T	10000	0.4	1	1	Numpy	34.6606	8.5962219	2.4221625	1.0492619	0.2390021
Gaussian	100000	0.4	1	1	Numpy	38.9716	7.3070067	1.4580849	0.2796693	0.0397723
Student T	100000	0.4	1	1	Numpy	34.5201	8.9952023	2.9335258	0.9585179	0.2041672
Gaussian	10000	0.2	1	1	Sobol	40.2525	7.3518071	0.9275436	0.0930167	0
Student T	10000	0.2	1	1	Sobol	37.0073	8.4933689	2.426569	0.8580968	0.1287563
Gaussian	10000	0.3	1	1	Sobol	39.6628	7.389744	1.1389362	0.1083232	0
Student T	10000	0.3	1	1	Sobol	36.9561	8.1075671	2.2297306	0.6737492	0.0849902
Gaussian	10000	0.4	1	1	Sobol	39.2178	7.5926464	1.3968448	0.2082582	0.0230613
Student T	10000	0.4	1	1	Sobol	36.0586	8.021399	2.2351214	0.5742483	0.12049
Gaussian	10000	0.5	1	1	Sobol	38.6344	7.7204051	1.580642	0.3482824	0.0192497
Student T	10000	0.5	1	1	Sobol	36.1918	8.1208005	2.3306287	0.6190721	0.1198186
Gaussian	10000	0.6	1	1	Sobol	37.9649	8.401119	1.7442209	0.4202513	0.0159116
Student T	10000	0.6	1	1	Sobol	36.5253	8.8718188	2.4997216	0.6528252	0.1087311
Gaussian	10000	0.4	1	0.8	Sobol	25.8271	4.1642199	0.3930736	0.0229965	0
Student T	10000	0.4	1	0.8	Sobol	23.3034	4.9087704	1.4907087	0.5731257	0.0698288
Gaussian	10000	0.4	1	0.9	Sobol	32.1362	5.9784332	0.8132995	0.0697942	0
Student T	10000	0.4	1	0.9	Sobol	30.1938	6.8053532	1.673787	0.4042805	0.0230656
Gaussian	10000	0.4	1	1	Sobol	39.2178	7.5926464	1.3968448	0.2082582	0.0230613
Student T	10000	0.4	1	1	Sobol	36.0586	8.021399	2.2351214	0.5742483	0.12049
Gaussian	10000	0.4	1	1.1	Sobol	46.6989	9.4839965	1.8971035	0.4180123	0.0231038
Student T	10000	0.4	1	1.1	Sobol	44.35	10.585516	2.8023289	0.8350218	0.0946776
Gaussian	10000	0.4	1	1.2	Sobol	55.3853	11.846645	2.4257383	0.6054332	0.0238531
Student T	10000	0.4	1	1.2	Sobol	53.2423	12.666034	3.6030232	0.8200178	0.1672633
Gaussian	10000	0.4	0.8	1	Sobol	41.1251	6.6866027	0.8806277	0.0698235	0
Student T	10000	0.4	0.8	1	Sobol	37.2088	7.8036671	1.9663601	0.3525803	0.0475801
Gaussian	10000	0.4	0.9	1	Sobol	40.1184	7.2454458	1.1397944	0.1159689	0.0230531
Student T	10000	0.4	0.9	1	Sobol	34.8083	8.1074442	2.2175277	0.4097952	0.0245136
Gaussian	10000	0.4	1	1	Sobol	39.2178	7.5926464	1.3968448	0.2082582	0.0230613
Student T	10000	0.4	1	1	Sobol	36.0586	8.021399	2.2351214	0.5742483	0.12049
Gaussian	10000	0.4	1.1	1	Sobol	38.3424	7.9951785	1.7510889	0.3006673	0.0230736
Student T	10000	0.4	1.1	1	Sobol	32.7678	9.3602803	3.1034097	1.1421581	0.1892319
Gaussian	10000	0.4	1.2	1	Sobol	37.3761	8.4473709	1.8956193	0.5107097	0.023085
Student T	10000	0.4	1.2	1	Sobol	31.5073	9.5907528	3.8284763	1.8477269	0.486851

4.2 Quantile Transformer – N_quantile Parameter Study

N_quantile = 25	N_quantile = 50
	
N_quantile = 75	N_quantile = 100
	

4.3 Student T Degree of Freedom Calibration



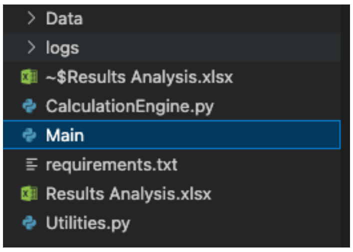
Canonical MLE does not display a explicit optimal degree of freedom, hence the DOF used for this project is 4. Future findings and recommendation is to recheck this section for correctness of implementation.

5 References

1. ISDA, ISDA Legal Guidelines for Smart Derivatives Contracts: Credit Derivatives, 2020
2. Monte Carlo Methods in Finance, Peter Jackel, Wiley Finance
3. Calibration of basket default swap: Evidence from Japanese market , Fathi Abid, Nader Naifar, October 2007

6 Code Implementation

1. Python Requirement.txt file is included in the code zip folder
2. Python version used : Python 3.10.9
3. 2 Module files : Utilities and Calculation Engine are included in Zip File.
   - Imported at the top of Main.py
4. There is log folder created to store logs.
5. Submitted version has all matplotlib codes commented out.
6. Ensure that module files and Data Folder are in the same directory of Main.py



```
> Data
> logs
~$Results Analysis.xlsx
CalculationEngine.py
Main
requirements.txt
Results Analysis.xlsx
Utilities.py
```

Coded: Basket Pricing Formula (calculationEngine.py), CDS Bootstrapping (calculationEngine.py), t-copula density formula (utilities.py)

Ready Solutions: Quantile Transform, Correlation Matrix, Sobol Sequence