# Model Documentation and Reported Values

Portfolio value today = 2.459556520976903e+08

CR01 = 17,100

Duration = 8.05

Convexity = 0.96

DV01 = 210,300

Notation: All numbers consider the gains distribution

## 1 day Market Monte Carlo Simulation: 5000 simulations

The data was gathered and cleaned as discussed above. The risk factor returns were computed for all risk factors. The risk factors were modelled as a multivariate normal random vector. A basic Monte Carlo simulation was used to compute the statistical measures reported below.

**Value at Risk**

1 day 99%

-3.794412794066981e+06

1 day 95%

-2.682935604614869e+06

10 day 99%

-1.199898681213510e+07

10 day 95%

-8.484187326143945e+06

**Conditional Value at Risk**

1 day 99%

-4.331694948432796e+06

1 day 95%

-3.349219375900260e+06

**Marginal Value at Risk (First percentile conditional expectation approximation)**

The marginal value at risk values were computed using the standard conditional expectation approximation.

MVaRBonds = -3.875677101084818e+06

MVaRCDS =1.583664991607715e+04

MVaROptions = -8.132530598024673e+03

MVaRSTOCKS =-5.142686786892892e+04

Sum of Marginal VaR’s = -3.919399849635694e+06 (pretty close to actual value at risk (modulo some numerical precision!))

**Incremental One Day 99% Value at Risk = VaR – VaR without asset class**

Bonds = 3.457357705787945e+06

CDS = 4.639840700576007e+05

Options = 5.397995765466839e+05

Stocks = 6.785293734007329e+05

**Value at Risk by Asset Class ( 1 day 99%)**

NO BONDS: -3.370550882790359e+05

NO CDS: -3.330428724009380e+06

NO OPTIONS: -3.254613217520297e+06

NO STOCKS: -3.115883420666248e+06

ONLY BONDS: -3.143531419669971e+06

ONLY CDS: -2.467024394043556e+08

ONLY OPTIONS: -3.914184377430077e+04

ONLY STOCKS: -2.811170144531494e+05

**1 Year Monte Carlo Market Value at Risk (2000 simulations)**

A 1 year Monte Carlo Value at Risk was computed due a lack of accuracy inherent in the square root of time rule. The hazard rates of the CDS’s, equities and option spot prices were modelled using GBM. The FX rates and implied bond spreads were modelled using a CIR model. The interest rates were simulated using principal component analysis assuming that the first principal component followed a Vasicek Process. All models were calibrated under the real world probability measure. The correlated processes were simulated over a year 1 year period using a 1 day time step. The theta of the instruments were also considered in the computation. Moreover, cash flows received were assumed to reinvested at the risk free rate. The following statistical measures were computed on the empirical distribution of the change in portfolio value:

**Value at Risk**

1 Year 95%

-5.444723859863874e+07

1 Year 99%

-7.838707050645650e+07

**Conditional Value at Risk**

1 Year 95%

-6.845196095960255e+07

**1 Year 99%**

-8.684251925440446e+07

## Historic Scenarios Computation of Value at Risk

**Value at Risk**

1 day 99%

-4.574142177442358e+06

1 day 95%

-2.965721716378355e+06

10 day 99%

-1.446470762215992e+07

10 day 95%

-9.378435529979493e+06

**Conditional Value at Risk**

1 day 99%

-1.132913456732699e+07

1 day 95%

-5.400742248045450e+06

**Comparison between Historic and MC**

By definition, the historic computation captures correlation and risk factors movements of past history.

Since the MC value at risk is less in absolute value, it is clearly not capturing some of the risks.

**Marginal Value at Risk (First percentile conditional expectation approximation)**

Due to a lack of historic data, marginal value at risk computations could not be computed accurately. This is because the conditional expectation approximation needs many simulations to be accurate.

**Incremental One Day 99% Value at Risk = VaR – VaR without asset class**

Bonds = 4.055062095744104e+06

CDS = -3.255234391375762e+05

Options = -3.225662024852028e+05

Stocks = -2.559787501177602e+05

**1 Year Credit Value at Risk (5000 simulations)**

**Credit VaR for Bonds**

The CreditMetrics approach using a Gaussian copula was used to compute the CreditVaR for bonds. The correlations between the issuers of the corporate bonds was estimated using the sector correlation matrix. The intra sector correlation was chosen to be the minimum value such that correlation matrix remained positive semi-definite. The intra sector correlations were determined to be 0.94. The theta of the bonds were not considered since it was already considered in the market value at risk computation. Therefore, we assume the shock occurs instantaneously.

*Credit Value at Risk (Bond Underlying) (1 year)*

95% -2.004451627069122e+07

99% -4.039523675556457e+07

99.9% -5.064057238647872e+07

**Credit VaR for CDS’s**

The CreditMetrics approach using a Gaussian copula was used to compute the CreditVaR for CDS underlying the instrument. The correlations between the issuers of the CDS’s was estimated using the sector correlation matrix. The intra sector correlation was chosen to be the minimum value such that correlation matrix remained positive semi-definite. The intra sector correlations were determined to be 0.89. The theta of the bonds were not considered since it was already considered in the market value at risk computation. Therefore, we assume the shock occurs instantaneously.

*Credit Value at Risk (CDS Underlying) (1 year)*

95% -2.817924122034073e+06

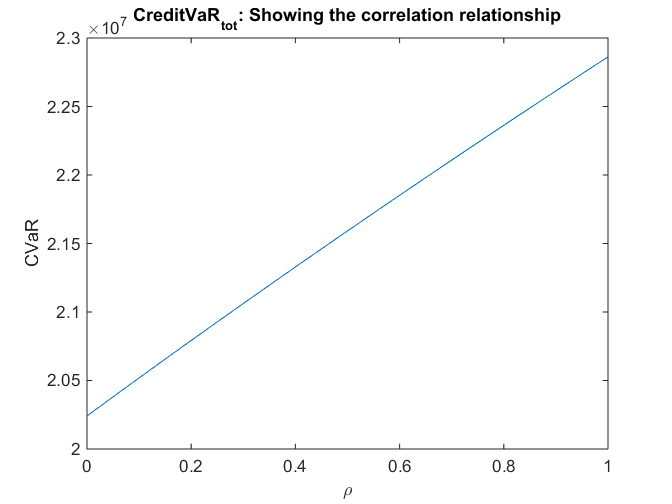
99% -4.616523998511046e+06

99.9% -7.772461573455572e+06

**Correlation in Credit Value at Risk Computations**

The correlation between CDS underlying’s and bond issuers is clearly not zero. We can get a bound on the Credit Value at Risk:

We can get the bounds assuming and



**Correlation between Market and Credit Risk**

Clearly there is correlation between market and credit risk. In there were perfect correlation, the total value at risk would be the sum of the CreditVaR and MarketVaR. In other cases,

We can get the bounds assuming and .

We also need to account for the bounds on the CreditVaR discussed above.

Combining all the inequalities, we can get a bound on the total VaR.

If CreditVaR takes its maximum value, then

8.980152744337106e+07 TotalVaR. 1.097049596471298e+08

If CreditVaR takes its minimum value, then

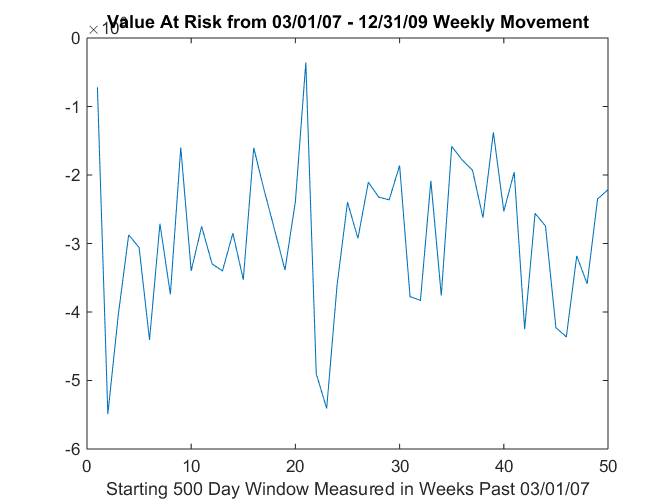
8.917032286212245e+07 TotalVaR. 1.070841429265200e+08

Therefore, we have the following bounds on the TotalVaR:

8.917032286212245e+07 TotalVaR. 1.097049596471298e+08

**Stressed Value at Risk using Historical Simulation**

To compute the stressed value at risk, data was taken from 03/01/07 - 12/31/09. It was found that the most stressed 504 day window occurred from 03/07/07 – 03/07/09. The reported stressed value at risk during this period was -5.490570602563009e+06.

****

**CDS Pricing**

The pricing of the Credit Default Swaps was done similarly to that of Hull in Chapter 25. In particular, the probability of survival to a period end and probability of default during a period was used to compute the present value of the expected payments, expected payoffs and accrual payments. In Hull, it is assumed that defaults can only occur at the midpoint of payment periods. We relax this assumption by allowing default events to occur on a daily basis. This greatly reduces numerical integration errors.

The CDS prices are reported as follows:

-3844133.31471624

-3211645.76442841

-1740857.70894546

-829861.519547225

-633670.452465153

-68110.6327644099

1629742.02252925

705762.662829994

-573604.271939782

**Credit Value Adjustment and Debt Value Adjustment**

Let be the risk neutral probability of default of the counterparty defaulting during the interval. Let be the present value of the expected loss to the bank if the counterparty defaults at the midpoint of the interval. Let be the risk neutral probability of the bank defaulting during the interval. Let be the present value of the expected loss to the counterparty if the bank defaults at the midpoint of the interval.

The probability of the counterparty defaulting during the interval is given by

The hazard rates are a function of the credit quality of the counterparty. Therefore, the default probability is a function of the current spread curves implied by the market. The computation of the involved a very time consuming Monte Carlo simulation. The spreads were simulated under the risk neutral pricing measure using the Ho-Lee model. The drift was chosen to be a function of the forward rate today. On each simulation trial, the potential exposure to the counterparty was computed at the midpoint of each interval. The exposure is equal to the max(V,0), where V is the expected value of each CDS contract at that future point in time. The variable, was set equal to the present value of the average exposure across each simulation multiplied by one minus the recovery rate. The variable was calculated similarly using the max(-V,0) to get the counterparty’s exposure to us. We assumed that the we are a AA rated firm.

The results are as follows:

CVA

4094.59605183508

1790.59080152533

0

0

7.14516539461641

12228.5385587937

42515.4317850556

53922.3989966669

281.764799690500

DVA

84570.0413030876

155831.594008841

3556.64714573887

1632.99310644233

1259.28492899519

5468.16123778521

1674.98574817641

571.604766328071

2808.94352655410

Then, the price adjusted value of the CDS’s are:

-3763657.86946499

-3057604.76122109

-1737301.06179972

-828228.526440783

-632418.312701553

-74871.0100854184

1588901.57649237

652411.868599656

-571077.093212918

**Economic Capital**

The expected value of the CDS portfolio in 10 days is -8.1815e+08.

## Model Vetting and Back Testing

The pricing models were vetted using a variety of methods. The most obvious method is simply looking at the change in portfolio value empirical distribution. If the distribution is not centered near zero, then there is probably an error with the model. We also checked if the percentage of the Value at Risk with respect to the total portfolio value was logical. For example, if the value at risk was a similar magnitude to the total portfolio value, there is most likely an issue with the model.

The instrument prices were checked with current market prices today. The Value at Risk calculations were also checked within models. For example, we found that the historical Value at Risk was not too different from the Monte Carlo Value at Risk. The models were also vetted by comparing them with elementary example found in Hull’s book.

For the 1 Year Monte Carlo simulation, we compared the value at risk generated to the scaled value computed in the 1 day Monte Carlo simulation. For example, using the square root of time rule, we find that the 1 year (99%) Value at Risk is about 70 million. Using the 1 year simulation model discussed above, the Value at Risk was computed to be 78 million. Therefore, the 1 day model agrees with the 1 year model. This implies that the modelling assumptions used in the 1 year simulation are not very different from simply assuming a normal distribution. However, there are other reasons why we wouldn’t choose a Gaussian process for all risk factors. For example, some of the risk factors cannot take negative values.

The most important model vetting technique we used was back testing. The back testing technique we used involved considering a moving window. For each window, we recalibrated the model and computed the Value at Risk. We then compared the next out of sample change in price to check if it had breached the VaR (95%). This was done for the Monte Carlo and Historical Value at Risk computations over 125 windows. The number of breaches in 125 trials was 5. Therefore, the empirical probability of breaching is approximately 4%.

The number of breaches in 125 trials is distributed according to a binomial distribution with probability of success equal to 5%. The probability of 5 or more breaches can be computed as:

1 – BINOMDIST(4, 125, 0.01, TRUE) = 0.754085499

At a 5% confidence level, we cannot reject the Monte Carlo model. This implies that the Monte Carlo simulation is not an unsuccessful model.

A similar computation was considered for the Historical Value at Risk; however, it was deemed that the computation was rather meaningless since it only tells us if past historic data is indicative/predictive of future historic data. For reference, the number of breaches in 125 trials was 8. This implies that the empirical probability of breaching is approximately 6.4%.

The number of breaches in 125 trials is distributed according to a binomial distribution with probability of success equal to 5%. The probability of 8 or more breaches can be computed as:

1 – BINOMDIST(8, 125, 0.01, TRUE) = 0.174548045

At a 5% confidence level, we cannot reject the Historical scenario model. This implies that the Historical scenario simulation is not an unsuccessful model.

In order to reject any of our models, we would need at least 10 breaches since the probability of 10 or more breaches is

1 – BINOMDIST(10, 125, 0.01, TRUE) = 0.049219173

# Appendix (INFRASTRUCTURE, DESIGN AND CODE)

## Object Oriented Pricing Design

The pricing was done using an object oriented design in MATLAB. Firstly a PortfolioConstants class was created. A pricing input class was created to be fed into the pricing function. The pricing function then takes an instance of the PricingInput class. Within the pricer, individual instrument pricing functions are called.

classdef PortfolioConstants

properties (Constant)

% Bonds

% Days in a year: days per annum

dpa = 252;

% Number of bonds in the portfolio

num\_of\_bonds = 25;

% Bond notionals

bond\_notionals = [100; 100; 100; 100; 100; 100; 100; 100; 100;...

100; 100; 100; 100; 100; 100; 100; 164.119708; 100; 100; 100;...

100; 100; 100; 100; 100];

% Bond positions

bond\_positions = [3100000; 1100000; 26600000; 3600000; 29600000;...

12400000; 17000000; -25800000; -1600000; 17400000; -25400000;...

17300000; -25900000; 20100000; 27200000; -26200000; 7200000;...

17000000; 24900000; 4500000; 18300000; -4900000; 15500000;...

-12000000; 24900000]./[100; 100; 100; 100; 100; 100; 100; 100; 100;...

100; 100; 100; 100; 100; 100; 100; 164.119708; 100; 100; 100;...

100; 100; 100; 100; 100];

% Bond coupon frequencies

bond\_coupon\_frequencies = [2; 2; 2; 2; 2; 1; 2; 2; 2; 2; 2; 2; 2; 2;...

2; 2; 2; 2; 2; 2; 2; 2; 2; 2; 2];

% Bond coupons

bond\_coupons = [0.0600; 0.0542; 0.0835; 0.1225; 0.0925; 0.0138;...

0.0688; 0.0600; 0.0538; 0.0613; 0.0775; 0.0765; 0.0675;...

0.0650; 0.0550; 0.0575; 0.0200; 0.0470; 0.0537; 0.0460;...

0.0430; 0.0570; 0.0495; 0.1100; 0.0900];

% Bond maturities

bonds\_maturity\_dates = {'2017-9-13'; '2017-3-15'; '2030-12-15';...

'2030-3-6'; '2032-6-1'; '2019-11-19'; '2023-9-27'; '2017-10-1';...

'2021-2-1'; '2017-10-5'; '2025-12-22'; '2031-12-30'; '2017-6-27';...

'2029-3-8'; '2018-11-15'; '2036-12-1'; '2036-12-1'; '2037-6-2';...

'2036-3-6'; '2021-6-1'; '2017-3-8'; '2017-3-2'; '2017-3-15';...

'2020-8-15'; '2024-8-23'};

% Bond sectors

bond\_sectors = {'Financial'; 'Financial'; 'Communications';...

'Government'; 'Communications'; 'Technology'; 'Government';...

'Financial'; 'Communications'; 'Communications';...

'Government'; 'Communications'; 'Government'; 'Government';...

'Government'; 'Government'; 'Government'; 'Government';...

'Financial'; 'Government'; 'Government'; 'Communications';...

'Communications'; 'Utilities'; 'Government'};

% Bond currencies

bond\_currencies = {'USD'; 'USD'; 'USD'; 'USD'; 'USD'; 'EUR';...

'USD'; 'USD'; 'USD'; 'USD'; 'CAD'; 'CAD'; 'CAD'; 'CAD';...

'CAD'; 'CAD'; 'CAD'; 'CAD'; 'CAD'; 'CAD'; 'CAD'; 'CAD';...

'CAD'; 'CAD'; 'CAD'};

% Bond spreads in bps

% bond\_spreads = [51.9065; 85.8025; 362.387; 387.6036; 165.6322;...

% 53.1067; 117.8879; 67.062; 239.0857; 115.8764; 144.2967;...

% 347.6342; 13.2579; 161.6595; 41.6512; 114.335; 8.066;...

% 114.0156; 310.8865; 36.6957; 14.442; 92.6546; 77.7333;...

% 62.8319; 101.5545];

% Options

num\_of\_options = 3;

% Strike prices

K = [56; 187; 58.84];

% Dividend yields

yields = [0.04; 0.03; 0.03];

% Options positions

options\_positions = [55000.00; 15000.00; 140000.00];

% Options maturities

options\_maturity\_dates = {'2016-10-24'; '2017-08-05'; '2017-10-04'};

% Stocks

num\_of\_stocks = 3;

% Stock positions

stock\_positions = [20000; 75000; 10000];

% CDS

num\_of\_cds = 9;

CDS\_accuracy = 30;

CDS\_frequency = 0.25;

CDS\_notional = 1e7;

CDS\_recovery\_rate = 0.4;

CDS\_longShort = [1; -1; 1; 1; 1; -1; 1; 1; 1];

CDS\_positions = [4; 2; 3; 2; 4; 4; 3; 1; 1];

CDS\_spreads = [0.0565; 0.0447; 0.0500; 0.0335; 0.0135; 0.0103;...

0.0089; 0.0115; 0.0500];

CDS\_lambda\_times = [0 0.5 1 2 3 4 5 7 10];

CDS\_time\_structure = [0 0.0833 0.25 0.5 1 2 3 5 7 10 30];

CDS\_zero\_coupon = [0, 0.00226367060325683, 0.0027666839924354,...

0.00409340443823036, 0.00532419495988368, 0.00755422001693362,...

0.00885593636286597, 0.0119704754783001, 0.0149700668534885,...

0.0172632827516919, 0.0271453738078202];

CDS\_lambda = [0, 0.25475, 0.425, 0.7, 0.825, 0.975, 1.092, 1.625, 2.2113;

0, 1.581, 2.025, 3.375, 4.4, 5.2, 6.7069, 7.825, 8.4695;

0, 0.99075, 1.1395, 1.7363, 2.4125, 3.0783, 3.7085, 5.1863, 5.652;

0, 0.207, 0.22, 0.31675, 0.44875, 0.592, 0.76803, 1.187, 1.3585;

0, 0.25475, 0.275, 0.475, 0.725, 0.925, 1.1251, 1.875, 2.3665;

0, 0.31, 0.3405, 0.43875, 0.5605, 0.7555, 1.0924, 1.3433, 1.814;

0, 0.307, 0.491, 0.758, 1.1627, 1.5397, 2.1225, 2.9635, 3.7382;

0, 0.50025, 0.575, 0.825, 1.05, 1.25, 1.5017, 1.925, 2.325;

0, 0.36125, 0.4, 0.725, 1.175, 1.6, 1.8541, 3.1015, 3.793]./100;

CDS\_maturity\_dates = {'2019-12-20'; '2020-06-20'; '2016-12-20';...

'2016-12-20'; '2016-12-20'; '2018-09-20'; '2019-12-20';...

'2021-06-20'; '2017-06-20'};

end

end

## A pricing input class was created to be fed into the pricer

classdef PricingInput

properties

num\_of\_days\_elapsed % Used to calculate times to maturity for all derivatives

stock\_prices

risk\_free\_rate % 1x1: USD risk-free rate used to price the three options

spot\_prices % 3x1: spot prices of underlyings

spot\_at\_expiration % 1x1: spot price at expiration for the first put option

implied\_vol % 3x1: implied vols used in BSM formula

zero\_curve\_USD

zero\_curve\_CAD

zero\_curve\_EUR

rate\_time\_structure

USDCAD

EURCAD

bond\_ratings % 25x1: each entry is an integer between 1 and 7. The best rating corresponds to 1 and the worst to 7.

bond\_market\_prices

historical\_implied\_spreads

cds\_underlying\_rating

% CDS data

CDS\_lambdas

% L\_GE

% L\_CNR

% L\_Sabre

% L\_Star

% L\_News

% L\_France

% L\_Cat

% L\_WF

% L\_Huntsm

% Pricing flags

CDS = 1;

STOCKS = 1;

OPTIONS = 1;

BONDS = 1;

end

%

methods

function obj = PricingInput(num\_of\_days\_elapsed\_, stock\_prices\_, risk\_free\_rate\_, spot\_prices\_,...

spot\_at\_expiration\_, implied\_vol\_, zero\_curve\_USD\_, zero\_curve\_CAD\_,...

zero\_curve\_EUR\_, rate\_time\_structure\_, USDCAD\_, EURCAD\_,...

bond\_ratings\_, lambdas)

if nargin > 0

obj.num\_of\_days\_elapsed = num\_of\_days\_elapsed\_;

obj.stock\_prices = stock\_prices\_;

obj.spot\_prices = spot\_prices\_;

obj.spot\_at\_expiration = spot\_at\_expiration\_;

obj.implied\_vol = implied\_vol\_;

obj.zero\_curve\_USD = zero\_curve\_USD\_;

obj.zero\_curve\_CAD = zero\_curve\_CAD\_;

obj.zero\_curve\_EUR = zero\_curve\_EUR\_;

obj.rate\_time\_structure = rate\_time\_structure\_;

obj.USDCAD = USDCAD\_;

obj.EURCAD = EURCAD\_;

obj.bond\_ratings = bond\_ratings\_;

obj.risk\_free\_rate = risk\_free\_rate\_;

obj.CDS\_lambdas = lambdas;

% obj.historical\_implied\_spreads = historical\_implied\_spreads\_;

% obj.L\_GE = lambdas(1, :);

% obj.L\_CNR = lambdas(2, :);

% obj.L\_Sabre = lambdas(3, :);

% obj.L\_Star = lambdas(4, :);

% obj.L\_News = lambdas(5, :);

% obj.L\_France = lambdas(6, :);

% obj.L\_Cat = lambdas(7, :);

% obj.L\_WF = lambdas(8, :);

% obj.L\_Huntsm = lambdas(9, :);

end

end

end

end

## The pricing function then takes an instance of the PricingInput class.

function [portfolio\_value, bonds\_value, options\_value, stocks\_value, cds\_values] = price(x)

% ------------------------------------------------------------------------

% Inputs: See the definition of the class 'PricingInput'.

%

% The typeof(x) = PricingInput

% ------------------------------------------------------------------------

% Create an object of type PortfolioConstants that contains as memebers all

% the portfolio constants necessary for pricing.

c = PortfolioConstants;

% Flags used to select what to price

bonds = x.BONDS;

options = x.OPTIONS;

cds = x.CDS;

stocks = x.STOCKS;

portfolio\_value = 0;

%% Bond pricing

if bonds == 1

% bonds\_value = price\_bonds(file\_path);

bonds\_value = price\_bonds\_with\_r\_and\_s\_implied(x, c);

else

bonds\_value = 0;

end

%% Option pricing

if options == 1

options\_value = price\_options(x, c);

else

options\_value = 0;

end

%% The value of the stocks

if stocks == 1

stocks\_value = x.USDCAD\*sum(x.stock\_prices.\*c.stock\_positions);

else

stocks\_value = 0;

end

%% CDS prices

if cds == 1

cds\_values = price\_cds(x, c);

else

cds\_values = 0;

end

%% Nothing priced, if all flags == 0

if sum([bonds cds options stocks]) == 0

fprintf('\nThe way you set the flags, nothing is being priced.\n')

return

end

%% Return

portfolio\_value = bonds\_value + options\_value + stocks\_value + sum(cds\_values);

end

## BOND PRICING

function bonds\_value = price\_bonds\_with\_r\_and\_s(x, c)

% This function prices the bonds in the portfolio.

%% Collect and process the data necessary for the calculations.

% Store values from raw\_data into separate arrays for clarity and

% convenience. The names of the variables are self-explanatory.

% Positions

position = c.bond\_positions;

% Face values

notionals = c.bond\_notionals;

% Coupon frequencies

coupon\_frequencies = c.bond\_coupon\_frequencies;

% Coupons in $

coupons = (c.bond\_coupons)./coupon\_frequencies.\*notionals;

% Time to maturity in years

T = (datenum(c.bonds\_maturity\_dates) - repmat(...

today + x.num\_of\_days\_elapsed, c.num\_of\_bonds, 1))/c.dpa;

% Number of coupons to be paid

num\_of\_coupons = ceil(coupon\_frequencies.\*T);

% The sectors represented in the portfolio

sectors = c.bond\_sectors;

% The spreads for each sector

% unique\_sectors = unique(sectors);

% spreads = load('spreads.mat');

% spreads = spreads.spreads;

% The currency each bond is in

currency = c.bond\_currencies;

% The ratings of the bonds

ratings = x.bond\_ratings;

% Tenor in years for the corporate spreads

tenor = [0.25; 0.5; 1; 2; 3; 4; 5; 7; 8; 9; 10; 15; 20; 25; 30]';

% Read the zero rates

% zero\_time\_structure = [0.25 0.5 1:10 15 20 30];

zero\_time\_structure = x.rate\_time\_structure;

% CAD rates

CAD\_rates = x.zero\_curve\_CAD;

% EURO rates

EUR\_rates = x.zero\_curve\_EUR;

% USD rates

USD\_rates = x.zero\_curve\_USD;

%% Array pre-allocation

bond\_prices\_r\_s = zeros(c.num\_of\_bonds, 1);

s\_impl = [0.0356; 0.0328; 0.0576; 0.0653; 0.0508; 0.0045; 0.0383; 0.0344;...

0.0423; 0.0347; 0.0363; 0.0536; 0.0280; 0.0334; 0.0240; 0.0345;...

0.0054\*5.9; 0.0323; 0.0478; 0.0237; 0.0181; 0.0297; 0.0236; 0.0369;...

0.0356];

exchange\_rate = [];

for k = 1:c.num\_of\_bonds

if(ratings(k) == 8)

bond\_prices\_r\_s(k) = 0.4 \* c.bond\_notionals(k);

continue;

end

% The spreads

% the\_spreads = sector2spreads(sectors{k}, unique\_sectors, spreads);

% the\_spreads(:, 1) = [];

% the\_spreads = cell2mat(the\_spreads(2:end, ratings(k)))'\*1e-4;

% The rates, depending on currency

switch currency{k}

case 'CAD'

zero\_rates = CAD\_rates;

exchange\_rate = 1.0;

case 'EUR'

zero\_rates = EUR\_rates;

exchange\_rate = x.EURCAD;

case 'USD'

zero\_rates = USD\_rates;

exchange\_rate = x.USDCAD;

end

% t(k) is the time from today to the k-th cashflow

t = zeros(1, num\_of\_coupons(k));

% cc(k) is the k-th cashflow

cc = t;

t(2:end) = -1/coupon\_frequencies(k);

t = fliplr(cumsum(t)) + T(k);

cc(:) = coupons(k);

cc(end) = cc(end) + notionals(k);

s = s\_impl(k);

%s = interp1([0 tenor 10\*tenor(end)], [the\_spreads(1) the\_spreads the\_spreads(end)], t);

r = interp1([0 zero\_time\_structure 10\*zero\_time\_structure(end)], [zero\_rates(1) zero\_rates zero\_rates(end)], t)/100;

if k == 17

I(length(t)) = 1.015;

I(:) = I(end);

I = I.^t;

bond\_prices\_r\_s(k) = exchange\_rate\*sum(I.\*cc.\*exp(-(s + r).\*t));

continue

end

bond\_prices\_r\_s(k) = exchange\_rate\*sum(cc.\*exp(-(s + r).\*t));

end

bonds\_value = position'\*bond\_prices\_r\_s;

end % price\_bonds function

## OPTION PRICING

function [options\_value, option\_prices] = price\_options(x, c)

%% Parameters necessary for pricing the European options

% Spot prices

S0 = x.spot\_prices;

% Strikes

K = c.K;

% Risk-free rates

r = [x.risk\_free\_rate; x.risk\_free\_rate; x.risk\_free\_rate];

% Time to maturity in years

T = (datenum(c.options\_maturity\_dates) -...

repmat(today + x.num\_of\_days\_elapsed, c.num\_of\_options, 1))/c.dpa;

% Implied volatilities

vols = x.implied\_vol;

% Dividend yields

y = c.yields;

%% Pricing of the options

option\_prices = zeros(c.num\_of\_options, 1);

if sum(T > 0) == c.num\_of\_options

% Index directly into the arrays to select the parameters for the European

% options

eur = 1:1:2;

amer = 3;

[~, option\_prices(eur)] = blsprice(S0(eur), K(eur), r(eur), T(eur),...

vols(eur), y(eur));

[~, tmp] = binprice(S0(amer), K(amer), r(amer), T(amer), 0.1,...

vols(amer), 1, y(amer));

option\_prices(amer) = tmp(1);

% All our options are USD

option\_prices = x.USDCAD\*option\_prices;

% mkt\_option\_prices = values\_of('Last Price');

else

% Assuming we won't simulate for more than a year, only the first

% option may expire.

if find(T <= 0) ~= 1

fprintf('\nSomething went wrong. Probably, your VaR horizon is too long\n')

return

end

% Treat the first European put separately

eur = 2;

amer = 3;

[~, option\_prices(eur)] = blsprice(S0(eur), K(eur), r(eur), T(eur),...

vols(eur), y(eur));

[~, tmp] = binprice(S0(amer), K(amer), r(amer), T(amer), 0.1,...

vols(amer), 1, y(amer));

option\_prices(amer) = tmp(1);

% Time in years for which the payoff of the put has been invested

t = (today + x.num\_of\_days\_elapsed - datenum(c.options\_maturity\_dates{1}))/c.dpa;

% Risk-free rate at which the payoff of the put has been invested

if t < min(x.rate\_time\_structure)

rfr = x.zero\_curve\_USD(1);

elseif t > max(x.rate\_time\_structure)

rfr = x.zero\_curve\_USD(end);

else

rfr = interp1(x.rate\_time\_structure, x.zero\_curve\_USD, t);

end

option\_prices(1) = max([0, K(1) - x.spot\_at\_expiration])\*exp(t\*rfr);

% All our options are USD

option\_prices = x.USDCAD\*option\_prices;

end

options\_value = option\_prices'\*c.options\_positions;

end

## CDS PRICING

function cds\_values = price\_cds(x, c)

% function price = price\_cds(quantity, notional, spread, frequency, daysToMat, lambdaTimes, lambda, longShort, recoveryRate, zeroTermTimes, zeroCurve, accuracy)

%

% Description: This function outputs the price of a CDS with the following inputs:

%

% quantity = the absolute value of the position in the CDS

% notional = the principle notional on the payments

% spread = the percent of the principle notional payed to the seller

% frequency = the number of payments per year

% daysToMat = the number of days until maturity

% lambda = the hazard rate term structure

% lambdaTimes = the time structure of the lambda structure

% longShort = long==1 and short==0

% recoveryRate = the recovery rate in case of default

% zeroTermTimes = the time structure of the zero rate points

% zeroCurve = a structure of zero rate points

% accuracy = Numerical integration accuracy coefficient: Choose greater

% than 30.

%

% MODEL: This model is an extension of pricing in Hull. The

% extension is that the number of default periods can be arbitrary; so any

% level of accuracy can be calculated.

% The number of CDS in the portfolio

num\_of\_cds = c.num\_of\_cds;

% Collect all the data necessary for the CDS pricing

daysToMat = datenum(c.CDS\_maturity\_dates) - repmat(...

today + x.num\_of\_days\_elapsed, c.num\_of\_cds, 1);

accuracy(num\_of\_cds) = c.CDS\_accuracy;

accuracy(:) = accuracy(end);

frequency(num\_of\_cds) = c.CDS\_frequency;

frequency(:) = frequency(end);

notional(num\_of\_cds) = c.CDS\_notional;

notional(:) = notional(end);

recoveryRate(num\_of\_cds) = c.CDS\_recovery\_rate;

recoveryRate(:) = recoveryRate(end);

longShort = c.CDS\_longShort;

quantity = c.CDS\_positions;

%spread = c.CDS\_spreads;

spread = x.historical\_implied\_spreads;

%ratings = x.cds\_underlying\_rating;

lambdaTimes = repmat(c.CDS\_lambda\_times, num\_of\_cds, 1);

zeroTermTimes = repmat(c.CDS\_time\_structure, num\_of\_cds, 1);

lambda = x.CDS\_lambdas;

%lambda = sortrows(lambda);

zeroCurve = repmat(c.CDS\_zero\_coupon, num\_of\_cds, 1);

%% CDS pricing

cds\_prices = zeros(num\_of\_cds, 1);

for k = 1:num\_of\_cds

T = daysToMat(k)/PortfolioConstants.dpa; % Convert days to years

deltaDefault = T/accuracy(k); % default periods

paymentTimes = fliplr(T:-1/frequency(k):0.001);

defaultTimes = deltaDefault/2:deltaDefault:T-deltaDefault/2; % Assume default times are in the middle of default periods

pSurvivalToPeriodEnd = exp(-repmat(interp(T, lambdaTimes(k, :), lambda((k), :)),1,length(paymentTimes)).\*paymentTimes);

pDefaultDuringPeriod = exp(-repmat(interp(T, lambdaTimes(k, :), lambda((k), :)), 1, length(defaultTimes)).\*(defaultTimes-deltaDefault/2))...

- exp(-repmat(interp(T, lambdaTimes(k, :), lambda((k), :)), 1, length(defaultTimes)).\*(defaultTimes+deltaDefault/2));

PVexpectedPayments = pSurvivalToPeriodEnd .\* (notional(k)\*spread((k))) \* exp(-interp(paymentTimes, zeroTermTimes(k, :), zeroCurve(k, :)) .\* paymentTimes');

PVexpectedPayoff = pDefaultDuringPeriod .\* (notional(k)\*(1-recoveryRate(k))) \* exp(-interp(defaultTimes, zeroTermTimes(k, :), zeroCurve(k, :)) .\* defaultTimes');

PVaccrualPayments = pDefaultDuringPeriod .\* (notional(k)\*deltaDefault\*spread((k))) \* exp(-interp(defaultTimes, zeroTermTimes(k, :), zeroCurve(k, :)) .\* defaultTimes');

cds\_prices(k) = PVexpectedPayoff - PVexpectedPayments - PVaccrualPayments;

end

cds\_values = x.USDCAD\*longShort.\*quantity.\*cds\_prices;

end

## 1 DAY MARKET VALUE AT RISK, MARGINAL, INCREMENTAL

clc;

clear all;

%INPUTS: 1. Historical time series for stocks, interest rates, fx rates,

% 2. Implied vol surface (for option pricing)

% 3. Spread curves for each credit rating

% 4. Transition matrices for credit quality

% Gathering Risk Factors

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

equities = xlsread('data.xlsx','Stocks','B6:D649');

underlying = xlsread('data.xlsx','Underlying','B7:D650');

impVol = xlsread('portfolio\_data.xlsm','Options','K2:K4');

zeroUSD = xlsread('data.xlsx','USD','B6:P649')./100;

zeroCurveTimes = [3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30];

zeroCAD = xlsread('data.xlsx','CAD','B7:AD650')./100;

zeroCAD( :, all( isnan( zeroCAD ), 1 ) ) = [];

zeroEUR = xlsread('data.xlsx','EUR','B7:AD650')./100;

zeroEUR( :, all( isnan( zeroEUR ), 1 ) ) = [];

fxUSDCAD = xlsread('data.xlsx','FX','B7:B650');

fxEURCAD = xlsread('data.xlsx','FX','E7:E650');

spreads = xlsread('SpreadsbySector','Communications','L2:R16')./10^4;

% Inputing spread time series for each CDS

cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

cdsCNQCN = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'K7:R650')))./100./100;

cdsSABR = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'S7:Z650')))./100./100;

cdsHOT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AA7:AH650')))./100./100;

cdsFOXA = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AI7:AP650')))./100./100;

cdsFRANCE = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AQ7:AX650')))./100./100;

cdsCAT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AY7:BF650')))./100./100;

cdsWFC = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BG7:BN650')))./100./100;

cdsHUNT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BO7:BV650')))./100./100;

bondSpreads = xlsread('Implied\_Yield','Yield','B2:Z645');

% Convert to hazard rates

lambdaGE = cdsGE ./ (1-0.4);

lambdaCNQCN = cdsCNQCN ./ (1-0.4);

lambdaSABR = cdsSABR ./ (1-0.4);

lambdaHOT = cdsHOT ./ (1-0.4);

lambdaFOXA = cdsFOXA ./ (1-0.4);

lambdaFRANCE = cdsFRANCE ./ (1-0.4);

lambdaCAT = cdsCAT ./ (1-0.4);

lambdaCAT(1,6) = lambdaCAT(2,6)+0.0001;

lambdaWFC = cdsWFC ./ (1-0.4);

lambdaHUNT = cdsHUNT ./ (1-0.4);

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

%Input risk factors in seperate matrices

riskFactors = [equities, underlying, fxUSDCAD, fxEURCAD, zeroUSD, zeroCAD, zeroEUR, lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT...

lambdaFOXA, lambdaFRANCE, lambdaCAT, lambdaWFC, lambdaHUNT, bondSpreads];

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.historical\_implied\_spreads = bondSpreads(end,:);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

%% Computing returns

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:8;

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

for y=9:53

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

for y=54:125

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

for y=126:150

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

%% Compute Market Value at Risk and Incremental Value at Risk

Sigma = cov(riskFactorReturns); % sample covariance matrix of risk factors

rho = corr(riskFactorReturns); % sample covariance matrix of risk factors

nSims = 600;

simMovements = mvnrnd(zeros(size(riskFactorReturns,2), 1), Sigma, nSims);

simRiskFactors = repmat(riskFactors(end,:),nSims,1) + simMovements;

% Price the simulated scenarios

prices = zeros(nSims,1);

bonds\_value = zeros(nSims,1);

options\_value = zeros(nSims,1);

stocks\_value = zeros(nSims,1);

cds\_values = zeros(nSims,1);

equities = simRiskFactors(:,1:3);

underlying = simRiskFactors(:,4:6);

fxUSDCAD = simRiskFactors(:,7);

fxEURCAD = simRiskFactors(:,8);

zeroUSD = simRiskFactors(:,9:23);

zeroCAD = simRiskFactors(:,24:38);

zeroEUR = simRiskFactors(:,39:53);

L = simRiskFactors(:,54:125);

bondSpreads = simRiskFactors(:,126:end);

spot\_at\_expiration = [];

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x] = price(priceObject);

cds\_values(i) = sum(x);

end

deltaP = prices - p0;

VaROneDay99 = prctile(deltaP,1);

VaRTenDay99 = sqrt(10) \* VaROneDay99;

CVaROneDay99 = mean(deltaP(deltaP < VaROneDay99));

VaROneDay95 = prctile(deltaP,5);

VaRTenDay95 = sqrt(10) \* VaROneDay95;

CVaROneDay95 = mean(deltaP(deltaP < VaROneDay95));

%% Marginal Value at Risk

ind = prctile(deltaP,0) < deltaP & deltaP < prctile(deltaP,2);

sum(ind)

MVaRBOND = mean( bonds\_value(ind) - bonds\_valueZERO);

MVaROPTIONS = mean( options\_value(ind) - options\_valueZERO);

MVaRSTOCKS = mean( stocks\_value(ind) - stocks\_valueZERO);

MVaRCDS = mean( cds\_values(ind) - sum(cds\_valuesZERO));

sVaR = MVaRBOND + MVaROPTIONS + MVaRSTOCKS + MVaRCDS;

nSims = 1000;

%% Incremental Value at Risk Calculations

prices = zeros(nSims,1);

% Remove Bonds

% Compute the price today with no bonds

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 0;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 0;

priceObject.CDS = 1;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOBONDS = prctile(deltaP,1);

incVarOneDayBONDS = -VaROneDay99 - -VaROneDay99NOBONDS;

% Remove CDS

prices = zeros(nSims,1);

% Compute the price today with no cds's

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 0;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

prices = zeros(nSims,1);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 0;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOCDS = prctile(deltaP,1);

incVarOneDayCDS = -VaROneDay99 - -VaROneDay99NOCDS;

% Remove OPTIONS

prices = zeros(nSims,1);

% Compute the price today with no options

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 0;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

prices = zeros(nSims,1);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 1;

priceObject.OPTIONS = 0;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOOPTIONS = prctile(deltaP,1);

incVarOneDayOPTIONS = -VaROneDay99 - -VaROneDay99NOOPTIONS;

% Remove STOCKS

prices = zeros(nSims,1);

% Compute the price today with no stocks

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 0;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

prices = zeros(nSims,1);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 1;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 0;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOSTOCKS = prctile(deltaP,1);

incVarOneDaySTOCKS = -VaROneDay99 - -VaROneDay99NOSTOCKS;

%% Compute Market Value at Risk and Incremental Value at Risk

Sigma = cov(riskFactorReturns); % sample covariance matrix of risk factors

rho = corr(riskFactorReturns); % sample covariance matrix of risk factors

nSims = 5000;

simMovements = mvnrnd(zeros(size(riskFactorReturns,2), 1), Sigma, nSims);

simRiskFactors = repmat(riskFactors(end,:),nSims,1) + simMovements;

% Price the simulated scenarios

prices = zeros(nSims,1);

bonds\_value = zeros(nSims,1);

options\_value = zeros(nSims,1);

stocks\_value = zeros(nSims,1);

cds\_values = zeros(nSims,1);

equities = simRiskFactors(:,1:3);

underlying = simRiskFactors(:,4:6);

fxUSDCAD = simRiskFactors(:,7);

fxEURCAD = simRiskFactors(:,8);

zeroUSD = simRiskFactors(:,9:23);

zeroCAD = simRiskFactors(:,24:38);

zeroEUR = simRiskFactors(:,39:53);

L = simRiskFactors(:,54:125);

bondSpreads = simRiskFactors(:,126:end);

spot\_at\_expiration = [];

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x] = price(priceObject);

cds\_values(i) = sum(x);

end

deltaP = prices - p0;

VaROneDay99 = prctile(deltaP,1);

VaRTenDay99 = sqrt(10) \* VaROneDay99;

CVaROneDay99 = mean(deltaP(deltaP < VaROneDay99));

VaROneDay95 = prctile(deltaP,5);

VaRTenDay95 = sqrt(10) \* VaROneDay95;

CVaROneDay95 = mean(deltaP(deltaP < VaROneDay95));

%% Marginal Value at Risk

ind = prctile(deltaP,0) < deltaP & deltaP < prctile(deltaP,2);

sum(ind)

MVaRBOND = mean( bonds\_value(ind) - bonds\_valueZERO);

MVaROPTIONS = mean( options\_value(ind) - options\_valueZERO);

MVaRSTOCKS = mean( stocks\_value(ind) - stocks\_valueZERO);

MVaRCDS = mean( cds\_values(ind) - sum(cds\_valuesZERO));

sVaR = MVaRBOND + MVaROPTIONS + MVaRSTOCKS + MVaRCDS;

nSims = 1000;

%% Asset Class Breakdown Value at Risk Calculations

prices = zeros(nSims,1);

% Remove Bonds

% Compute the price today with no bonds

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 0;

z.OPTIONS = 0;

z.STOCKS = 0;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 0;

priceObject.OPTIONS = 0;

priceObject.STOCKS = 0;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99BONDS = prctile(deltaP,1);

% Remove CDS

prices = zeros(nSims,1);

% Compute the price today with no cds's

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 0;

z.OPTIONS = 0;

z.STOCKS = 0;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

prices = zeros(nSims,1);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 0;

priceObject.CDS = 1;

priceObject.OPTIONS = 0;

priceObject.STOCKS = 0;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99CDS = prctile(deltaP,1);

% Remove OPTIONS

prices = zeros(nSims,1);

% Compute the price today with no options

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 0;

z.CDS = 0;

z.OPTIONS = 1;

z.STOCKS = 0;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

prices = zeros(nSims,1);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 0;

priceObject.CDS = 0;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 0;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99OPTIONS = prctile(deltaP,1);

% Remove STOCKS

prices = zeros(nSims,1);

% Compute the price today with no stocks

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 0;

z.CDS = 0;

z.OPTIONS = 0;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(z);

prices = zeros(nSims,1);

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1), reshape(L(i,:),8,9)']);

priceObject.BONDS = 0;

priceObject.CDS = 0;

priceObject.OPTIONS = 0;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreads(i,:);

prices(i) = price(priceObject);

end

deltaP = prices - p0;

VaROneDay99STOCKS = prctile(deltaP,1);

## 1 YEAR MARKET VALUE AT RISK

clc;

clear all;

%INPUTS: 1. Historical time series for stocks, interest rates, fx rates,

% 2. Implied vol surface (for option pricing)

% 3. Spread curves for each credit rating

% 4. Transition matrices for credit quality

% Gathering Risk Factors

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

equities = xlsread('data.xlsx','Stocks','B6:D649');

underlying = xlsread('data.xlsx','Underlying','B7:D650');

impVol = xlsread('portfolio\_data.xlsm','Options','K2:K4');

zeroUSD = xlsread('data.xlsx','USD','B6:P649')./100;

zeroCurveTimes = [3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30];

zeroCAD = xlsread('data.xlsx','CAD','B7:AD650')./100;

zeroCAD( :, all( isnan( zeroCAD ), 1 ) ) = [];

zeroEUR = xlsread('data.xlsx','EUR','B7:AD650')./100;

zeroEUR( :, all( isnan( zeroEUR ), 1 ) ) = [];

fxUSDCAD = xlsread('data.xlsx','FX','B7:B650');

fxEURCAD = xlsread('data.xlsx','FX','E7:E650');

spreads = xlsread('SpreadsbySector','Communications','L2:R16')./10^4;

bondSpreads = xlsread('Implied\_Yield','Yield','B2:Z645');

% Inputing spread time series for each CDS

cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

cdsCNQCN = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'K7:R650')))./100./100;

cdsSABR = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'S7:Z650')))./100./100;

cdsHOT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AA7:AH650')))./100./100;

cdsFOXA = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AI7:AP650')))./100./100;

cdsFRANCE = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AQ7:AX650')))./100./100;

cdsCAT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AY7:BF650')))./100./100;

cdsWFC = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BG7:BN650')))./100./100;

cdsHUNT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BO7:BV650')))./100./100;

% Convert to hazard rates

lambdaGE = cdsGE ./ (1-0.4);

lambdaCNQCN = cdsCNQCN ./ (1-0.4);

lambdaSABR = cdsSABR ./ (1-0.4);

lambdaHOT = cdsHOT ./ (1-0.4);

lambdaFOXA = cdsFOXA ./ (1-0.4);

lambdaFRANCE = cdsFRANCE ./ (1-0.4);

lambdaCAT = cdsCAT ./ (1-0.4);

lambdaCAT(1,6) = lambdaCAT(2,6)+0.0001;

lambdaWFC = cdsWFC ./ (1-0.4);

lambdaHUNT = cdsHUNT ./ (1-0.4);

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

bondSpreads = xlsread('Implied\_Yield','Yield','B2:Z645');

%Input risk factors in seperate matrices

riskFactors = [equities, underlying, fxUSDCAD, fxEURCAD, zeroUSD, zeroCAD, zeroEUR, lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT...

lambdaFOXA, lambdaFRANCE, lambdaCAT, lambdaWFC,lambdaHUNT, bondSpreads];

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

%% Principal component analysis

zeroRates = [zeroUSD, zeroCAD, zeroEUR];

% USD

S = cov(zeroUSD);

% Eigenvalue decomposition

[QUSD, LambdaUSD] = pcacov(S);

Q1 = QUSD(:,1);

Lambda1 = LambdaUSD(1,1);

weights = zeros(size(zeroUSD,2),1);

for y=1:size(zeroUSD,2)

weights(y) = LambdaUSD(y) ./ sum(LambdaUSD);

end

xUSD = (diag(LambdaUSD)^(-1/2) \* QUSD' \* zeroUSD')';

% CAD

S = cov(zeroCAD);

% Eigenvalue decomposition

[QCAD, LambdaCAD] = pcacov(S);

Q1 = QCAD(:,1);

Lambda1 = LambdaCAD(1,1);

weights = zeros(size(zeroCAD,2),1);

for y=1:size(zeroCAD,2)

weights(y) = LambdaCAD(y) ./ sum(LambdaCAD);

end

xCAD = (diag(LambdaCAD)^(-1/2) \* QCAD' \* zeroCAD')';

% EUR

S = cov(zeroEUR);

% Eigenvalue decomposition

[QEUR, LambdaEUR] = pcacov(S);

Q1 = QEUR(:,1);

Lambda1 = LambdaEUR(1,1);

weights = zeros(size(zeroEUR,2),1);

for y=1:size(zeroEUR,2)

weights(y) = LambdaEUR(y) ./ sum(LambdaEUR);

end

xEUR = (diag(LambdaEUR)^(-1/2) \* QEUR' \* zeroEUR')';

PCAriskFactors = [ xUSD(:,1), xCAD(:,1), xEUR(:,1)];

%% Computing returns

riskFactors = [equities, underlying, fxUSDCAD, fxEURCAD, xUSD(:,1), xCAD(:,1), xEUR(:,1), lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT...

lambdaFOXA, lambdaFRANCE, lambdaCAT, lambdaWFC,lambdaHUNT, bondSpreads];

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:83;

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

for y=84:108;

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

%% Compute correlation matrix of risk factors

rho = corr(riskFactorReturns);

% Generate normalized correlated Gaussian random variables (can also be

% used for credit modelling)

delta = 1/252;

T = 1; % Number of years to simulate in the future

N = T/delta;

nSims = 200;

prices = zeros(nSims,1);

counterPartyRatings =[2 4 4 4 4 4 3 3 6];

impLambda = c.CDS\_lambda;

lambdaTimes = c.CDS\_lambda\_times;

for i=1:nSims

zz = mvnrnd(zeros(1,size(riskFactorReturns,2)), rho, N);

% Simulate all risk factors using correct models and correlations

zEquities = zz(:,1:3);

zUnderlying = zz(:,4:6);

zFXUSDCAD = zz(:,7);

zFXEURCAD = zz(:,8);

zZeroUSD = zz(:,9);

zZeroCAD = zz(:,10);

zZeroEUR = zz(:,11);

zLambda = zz(:,12:83);

zBondSpreads = zz(:,84:end);

x = 12:83;

lambdaSim = simGBM(riskFactors(end,x)',mean(riskFactorReturns(:,x))'.\*252, std(riskFactorReturns(:,x))'.\*sqrt(252), zLambda, delta,T);

plot(1:252+644,[riskFactors(:, x);lambdaSim(:,1:length(x))])

bondSpreadSim = simGBM(riskFactors(end,84:end)',mean(riskFactorReturns(:,84:end))'.\*252, std(riskFactorReturns(:,84:end))'.\*sqrt(252), zBondSpreads, delta,T);

plot(1:252+644,[riskFactors(:, 84:108);bondSpreadSim(:,1:length(84:108))])

equitiesSim = simGBM(equities(end,:)',mean(riskFactorReturns(:,1:3))'.\*252,std(riskFactorReturns(:,1:3))'.\*sqrt(252), zEquities, delta,T);

underylingSim = simGBM(underlying(end,:)',mean(riskFactorReturns(:,4:6))'.\*252,std(riskFactorReturns(:,4:6))'.\*sqrt(252), zUnderlying, delta,T);

[a1, m1, s1] = CalibrateCIR(fxUSDCAD,1/252);

[a2, m2, s2] = CalibrateCIR(fxEURCAD,1/252);

fxSim = simCIR([fxUSDCAD(end);fxEURCAD(end)],[a1;a2], [m1;m2], [s1;s2], [zFXUSDCAD,zFXEURCAD], T, delta);

[t, s, k] = CalibrateVasicek(xUSD(:,1),delta);

xSimUSD = simVas(xUSD, T, delta, zZeroUSD, t, s, k);

USDsim = (QUSD \* diag(LambdaUSD)^(1/2) \* xSimUSD')';

[t, s, k] = CalibrateVasicek(xCAD(:,1),delta);

xSimCAD = simVas(xCAD, T, delta, zZeroCAD, t, s, k);

CADsim = (QCAD \* diag(LambdaCAD)^(1/2) \* xSimCAD')';

[t, s, k] = CalibrateVasicek(xEUR(:,1),delta);

xSimEUR = simVas(xEUR, T, delta, zZeroEUR, t, s, k);

EURsim = (QEUR \* diag(LambdaEUR)^(1/2) \* xSimEUR')';

daysToExp = wrkdydif('7/11/2016', '10/24/2016', 1);

spot\_at\_expiration = equitiesSim(daysToExp/252/delta,1);

x = PricingInput(1, equitiesSim(end,:)', USDsim(end,1), underylingSim(end,:)', spot\_at\_expiration, impVol, USDsim(end,:), CADsim(end,:), EURsim(end,:), zeroCurveTimes,...

fxSim(end,1), fxSim(end,2), currentRatings, [zeros(9,1), reshape(lambdaSim(end,:),8,9)']);

x.historical\_implied\_spreads = bondSpreadSim(end,:);

[prices(i), ~, ~, ~, cds] = price(x);

end

deltaP = prices - p0;

hist(deltaP);

VaR99 = prctile(deltaP, 1);

VaR95 = prctile(deltaP, 5);

CVaR99 = mean(deltaP(deltaP < VaR99));

CVaR95 = mean(deltaP(deltaP < VaR95));

## HISTORIC VALUE AT RISK

clc;

clear all;

%

%% Gathering Risk Factors

equities = flipud(xlsread('data.xlsx','Stocks','B6:D649'));

underlying = xlsread('data.xlsx','Underlying','B7:D650');

impVol = xlsread('portfolio\_data.xlsm','Options','K2:K4');

zeroUSD = xlsread('data.xlsx','USD','B6:P649')./100;

zeroCurveTimes = [3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30];

zeroCAD = xlsread('data.xlsx','CAD','B7:AD650')./100;

zeroCAD( :, all( isnan( zeroCAD ), 1 ) ) = [];

zeroEUR = xlsread('data.xlsx','EUR','B7:AD650')./100;

zeroEUR( :, all( isnan( zeroEUR ), 1 ) ) = [];

fxUSDCAD = xlsread('data.xlsx','FX','B7:B650');

fxEURCAD = xlsread('data.xlsx','FX','E7:E650');

spreads = xlsread('SpreadsbySector','Communications','L2:R16')./10^4;

% Inputing spread time series for each CDS

cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

cdsCNQCN = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'K7:R650')))./100./100;

cdsSABR = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'S7:Z650')))./100./100;

cdsHOT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AA7:AH650')))./100./100;

cdsFOXA = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AI7:AP650')))./100./100;

cdsFRANCE = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AQ7:AX650')))./100./100;

cdsCAT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AY7:BF650')))./100./100;

cdsWFC = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BG7:BN650')))./100./100;

cdsHUNT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BO7:BV650')))./100./100;

% Convert to hazard rates

lambdaGE = cdsGE ./ (1-0.4);

lambdaCNQCN = cdsCNQCN ./ (1-0.4);

lambdaSABR = cdsSABR ./ (1-0.4);

lambdaHOT = cdsHOT ./ (1-0.4);

lambdaFOXA = cdsFOXA ./ (1-0.4);

lambdaFRANCE = cdsFRANCE ./ (1-0.4);

lambdaCAT = cdsCAT ./ (1-0.4);

lambdaCAT(1,6) = lambdaCAT(2,6)+0.0001;

lambdaWFC = cdsWFC ./ (1-0.4);

lambdaHUNT = cdsHUNT ./ (1-0.4);

% Input risk factors in seperate matrices

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

bondSpreads = xlsread('Implied\_Yield','Yield','B2:Z645');

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.historical\_implied\_spreads = bondSpreads(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

%% Computing returns

riskFactors = [equities, underlying, fxUSDCAD, fxEURCAD, zeroUSD, zeroCAD, zeroEUR, lambdaGE, lambdaCNQCN, ...

lambdaSABR, lambdaHOT, lambdaFOXA, lambdaFRANCE, lambdaCAT, lambdaWFC, lambdaHUNT, bondSpreads];

riskFactorsToday = riskFactors(end,:);

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:size(riskFactorReturns,2);

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

scenarios = zeros(size(riskFactorReturns));

for i=1:size(scenarios,1)

scenarios(i,:) = (1+riskFactorReturns(i,:)) .\* riskFactorsToday;

end

usdThreeMonth = scenarios(:,9);

scen = scenarios;

scen(find(usdThreeMonth<0),:) = [];

equitiesScen = scen(:,1:3);

underlyingScen = scen(:,4:6);

fxUSDCADScen = scen(:,7);

fxEURCADScen = scen(:,8);

zeroUSDScen = scen(:,9:23);

zeroCADScen = scen(:,24:38);

zeroEURScen = scen(:,39:53);

lambdaScen = scen(:,54:125);

bondSpreadScen = scen(:,126:end);

prices = zeros(size(scen,1),1);

bonds\_value = zeros(size(scen,1),1);

options\_value = zeros(size(scen,1),1);

stocks\_value = zeros(size(scen,1),1);

cds\_values = zeros(size(scen,1),1);

c = PortfolioConstants;

for i=1:size(scen,1)

x = PricingInput(1,equitiesScen(i,:)',zeroUSDScen(i,1), underlyingScen(i,:)', [], impVol, zeroUSDScen(i,:), zeroCADScen(i,:),...

zeroEURScen(i,:), zeroCurveTimes, fxUSDCADScen(i), fxEURCADScen(i), currentRatings, [zeros(9,1), reshape(lambdaScen(i,:),8,9)']);

x.historical\_implied\_spreads = bondSpreadScen(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x]= price(x);

end

deltaP = prices - p0;

hist(deltaP, 50);

VaROneDay99 = prctile(deltaP, 1);

VaROneDay95 = prctile(deltaP, 4);

VaR99Ten = sqrt(10) \* VaROneDay99;

VaR95Ten = sqrt(10) \* VaROneDay95;

CVaROneDay99 = mean(deltaP(deltaP < VaROneDay99));

CVaROneDay95 = mean(deltaP(deltaP < VaROneDay95));

%% Marginal Value at Risk

ind = prctile(deltaP,0) < deltaP & deltaP < prctile(deltaP,2);

sum(ind)

MVaRBOND = mean( bonds\_value(ind) - bonds\_valueZERO);

MVaROPTIONS = mean( options\_value(ind) - options\_valueZERO);

MVaRSTOCKS = mean( stocks\_value(ind) - stocks\_valueZERO);

MVaRCDS = mean( cds\_values(ind) - sum(cds\_valuesZERO));

sVaR = MVaRBOND + MVaROPTIONS + MVaRSTOCKS + MVaRCDS;

%% Incremental Value at Risk Calculations

% Remove Bonds

% Compute the price today with no bonds

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings,[zeros(9,1), lambda]);

z.BONDS = 0;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreadScen(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

prices = zeros(size(scen,1),1);

for i=1:size(scen,1)

priceObject = PricingInput(1,equitiesScen(i,:)',zeroUSDScen(i,1), underlyingScen(i,:)', [], impVol, zeroUSDScen(i,:), zeroCADScen(i,:),...

zeroEURScen(i,:), zeroCurveTimes, fxUSDCADScen(i), fxEURCADScen(i), currentRatings, [zeros(9,1),reshape(lambdaScen(i,:),8,9)']);

priceObject.BONDS = 0;

priceObject.CDS = 1;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreadScen(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x]= price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOBONDS = prctile(deltaP,1);

incVarOneDayBONDS = -VaROneDay99 - -VaROneDay99NOBONDS;

% Remove CDS

% Compute the price today with no bonds

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 0;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreadScen(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

prices = zeros(size(scen,1),1);

for i=1:size(scen,1)

priceObject = PricingInput(1,equitiesScen(i,:)',zeroUSDScen(i,1), underlyingScen(i,:)', [], impVol, zeroUSDScen(i,:), zeroCADScen(i,:),...

zeroEURScen(i,:), zeroCurveTimes, fxUSDCADScen(i), fxEURCADScen(i), currentRatings, [zeros(9,1),reshape(lambdaScen(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 0;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreadScen(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x]= price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOCDS = prctile(deltaP,1);

incVarOneDayCDS = -VaROneDay99 - -VaROneDay99NOCDS;

% Remove OPTIONS

% Compute the price today with no bonds

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 0;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreadScen(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

prices = zeros(size(scen,1),1);

for i=1:size(scen,1)

priceObject = PricingInput(1,equitiesScen(i,:)',zeroUSDScen(i,1), underlyingScen(i,:)', [], impVol, zeroUSDScen(i,:), zeroCADScen(i,:),...

zeroEURScen(i,:), zeroCurveTimes, fxUSDCADScen(i), fxEURCADScen(i), currentRatings, [zeros(9,1),reshape(lambdaScen(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 1;

priceObject.OPTIONS = 0;

priceObject.STOCKS = 1;

priceObject.historical\_implied\_spreads = bondSpreadScen(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x]= price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOOPTIONS = prctile(deltaP,1);

incVarOneDayOPTIONS = -VaROneDay99 - -VaROneDay99NOOPTIONS;

% Remove Stocks

% Compute the price today with no bonds

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 0;

z.historical\_implied\_spreads = bondSpreadScen(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

prices = zeros(size(scen,1),1);

for i=1:size(scen,1)

priceObject = PricingInput(1,equitiesScen(i,:)',zeroUSDScen(i,1), underlyingScen(i,:)', [], impVol, zeroUSDScen(i,:), zeroCADScen(i,:),...

zeroEURScen(i,:), zeroCurveTimes, fxUSDCADScen(i), fxEURCADScen(i), currentRatings, [zeros(9,1),reshape(lambdaScen(i,:),8,9)']);

priceObject.BONDS = 1;

priceObject.CDS = 1;

priceObject.OPTIONS = 1;

priceObject.STOCKS = 0;

priceObject.historical\_implied\_spreads = bondSpreadScen(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x]= price(priceObject);

end

deltaP = prices - p0;

VaROneDay99NOSTOCKS = prctile(deltaP,1);

incVarOneDaySTOCKS = -VaROneDay99 - -VaROneDay99NOSTOCKS;

## CREDIT VALUE AT RISK FOR BONDS

%% Credit Modelling of Sectors

% Here we assume perfect correlation between companies in a given sector

% Order is Communications, Financial, G

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

%rho = xlsread('portfolio\_data (CR).xlsm', 'CreditRisk', 'AD78:BB102');

nFirms = length(currentRatings);

nSectors = 5;

Trans = ...

[100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00;

0.30 95.21 4.49 0.00 0.00 0.00 0.00 0.00;

0.00 1.48 92.87 5.65 0.00 0.00 0.00 0.00;

0.00 0.06 3.33 91.37 5.24 0.00 0.00 0.00;

0.00 0.00 0.00 3.99 88.00 7.56 0.26 0.18;

0.00 0.00 0.00 0.17 4.13 87.67 5.27 2.76;

0.00 0.00 0.00 0.00 0.00 7.20 61.15 31.65;

zeros(1,7) 100] ./ 100;

n = 1000;

x = zeros(n,2);

% Use Gaussian Copula model to model credit movements

w = zeros(n,1);

for i=1:n

w(i) = 1;

y = i/n;

x(i,1) = y;

rho = [1 y 0.368 0.124 0.368 -0.314 0.124 y 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 y 0.124 0.124 0.368 0.368 0.01 0.124;

y 1 0.368 0.124 0.368 -0.314 0.124 y 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 y 0.124 0.124 0.368 0.368 0.01 0.124;

0.368 0.368 1 -0.672 y -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.124 0.124 -0.672 1 -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.368 0.368 y -0.672 1 -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

-0.314 -0.314 -0.964 0.561 -0.964 1 0.561 -0.314 -0.964 -0.964 0.561 -0.964 0.561 0.561 0.561 0.561 0.561 0.561 -0.314 0.561 0.561 -0.964 -0.964 -0.874 0.561;

0.124 0.124 -0.672 y -0.672 0.561 1 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

y y 0.368 0.124 0.368 -0.314 0.124 1 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 y 0.124 0.124 0.368 0.368 0.01 0.124;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 1 y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y 1 -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 1 -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y y -0.672 1 -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 1 y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y 1 y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y 1 y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y 1 y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y 1 y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y 1 0.124 y y -0.672 -0.672 -0.67 y;

y y 0.368 0.124 0.368 -0.314 0.124 y 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 1 0.124 0.124 0.368 0.368 0.01 0.124;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 1 y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y 1 -0.672 -0.672 -0.67 y;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 1 y 0.844 -0.672;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y 1 0.844 -0.672;

0.01 0.01 0.844 -0.67 0.844 -0.874 -0.67 0.01 0.844 0.844 -0.67 0.844 -0.67 -0.67 -0.67 -0.67 -0.67 -0.67 0.01 -0.67 -0.67 0.844 0.844 1 -0.67;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 1;];

if(sum(eig(rho) < 0) > 0)

x(i,2) = 0;

else

x(i,2) = 1;

end

end

y = 0.9734;

rho = [1 y 0.368 0.124 0.368 -0.314 0.124 y 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 y 0.124 0.124 0.368 0.368 0.01 0.124;

y 1 0.368 0.124 0.368 -0.314 0.124 y 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 y 0.124 0.124 0.368 0.368 0.01 0.124;

0.368 0.368 1 -0.672 y -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.124 0.124 -0.672 1 -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.368 0.368 y -0.672 1 -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

-0.314 -0.314 -0.964 0.561 -0.964 1 0.561 -0.314 -0.964 -0.964 0.561 -0.964 0.561 0.561 0.561 0.561 0.561 0.561 -0.314 0.561 0.561 -0.964 -0.964 -0.874 0.561;

0.124 0.124 -0.672 y -0.672 0.561 1 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

y y 0.368 0.124 0.368 -0.314 0.124 1 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 y 0.124 0.124 0.368 0.368 0.01 0.124;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 1 y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y 1 -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 1 -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y y -0.672 1 -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y y 0.844 -0.672;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 1 y y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y 1 y y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y 1 y y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y 1 y y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y 1 y 0.124 y y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y 1 0.124 y y -0.672 -0.672 -0.67 y;

y y 0.368 0.124 0.368 -0.314 0.124 y 0.368 0.368 0.124 0.368 0.124 0.124 0.124 0.124 0.124 0.124 1 0.124 0.124 0.368 0.368 0.01 0.124;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 1 y -0.672 -0.672 -0.67 y;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y 1 -0.672 -0.672 -0.67 y;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 1 y 0.844 -0.672;

0.368 0.368 y -0.672 y -0.964 -0.672 0.368 y y -0.672 y -0.672 -0.672 -0.672 -0.672 -0.672 -0.672 0.368 -0.672 -0.672 y 1 0.844 -0.672;

0.01 0.01 0.844 -0.67 0.844 -0.874 -0.67 0.01 0.844 0.844 -0.67 0.844 -0.67 -0.67 -0.67 -0.67 -0.67 -0.67 0.01 -0.67 -0.67 0.844 0.844 1 -0.67;

0.124 0.124 -0.672 y -0.672 0.561 y 0.124 -0.672 -0.672 y -0.672 y y y y y y 0.124 y y -0.672 -0.672 -0.67 1;];

eig(rho)

cumT = cumsum(Trans,2);

cumT(:,end) = ones(8,1);

transitionValues = norminv(cumT);

transitionValues(isnan( transitionValues )) = inf;

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = equities(end,1);

x = PricingInput(1,equities(end,:)',zeroUSD(end,1), underlying(end,:)', spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

x.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(x);

nSims = 500;

sims = mvnrnd(zeros(nFirms,1),rho, nSims);

futureRatings = zeros(nSims,nFirms);

prices = zeros(nSims,1);

x.num\_of\_days\_elapsed = 1;

for i=1:nSims

futureRatings(i,:) = simRating(sims(i,:), currentRatings, transitionValues);

x.bond\_ratings = futureRatings(i,:);

prices(i) = price(x);

end

hist(prices - p0,64);

CreditVaR99 = prctile(prices - p0, 1);

CreditVaR95 = prctile(prices - p0, 5);

CreditVar999 = prctile(prices - p0, 0.1);

## CREDIT VALUE AT RISK FOR CDS’

%% Credit Modelling For CDS's

% Order is Communications, Financial, G

currentRatings = [3 4 4 4 4 2 3 3 6]; % These are the current ratings of each firm

bondRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1];

%rho = xlsread('portfolio\_data (CR).xlsm', 'CreditRisk', 'AD78:BB102');

nFirms = length(currentRatings);

nSectors = 5;

Trans = ...

[100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00;

0.30 95.21 4.49 0.00 0.00 0.00 0.00 0.00;

0.00 1.48 92.87 5.65 0.00 0.00 0.00 0.00;

0.00 0.06 3.33 91.37 5.24 0.00 0.00 0.00;

0.00 0.00 0.00 3.99 88.00 7.56 0.26 0.18;

0.00 0.00 0.00 0.17 4.13 87.67 5.27 2.76;

0.00 0.00 0.00 0.00 0.00 7.20 61.15 31.65;

zeros(1,7) 100] ./ 100;

n = 1000;

x = zeros(n,2);

% Use Gaussian Copula model to model credit movements

w = zeros(n,1);

for i=1:n

w(i) = 1;

y = i/n;

x(i,1) = y;

rho = [1 0.737 0.968 0.887 0.879 0.161 0.913 y 0.828;

0.737 1 0.752 0.709 0.828 0.135 0.713 0.737 0.874;

0.968 0.752 1 0.944 0.908 0.285 0.965 0.968 0.879;

0.887 0.709 0.944 1 0.946 0.539 0.987 0.887 0.827;

0.879 0.828 0.908 0.946 1 0.416 0.935 0.879 0.872;

0.161 0.135 0.285 0.539 0.416 1 0.503 0.161 0.209;

0.913 0.713 0.965 0.987 0.935 0.503 1 0.913 0.844;

y 0.737 0.968 0.887 0.879 0.161 0.913 1 0.828;

0.828 0.874 0.879 0.827 0.872 0.209 0.844 0.828 1;];

if(sum(eig(rho) < 0) > 0)

x(i,2) = 0;

else

x(i,2) = 1;

end

end

y = 0.94;

rho = [1 0.737 0.968 0.887 0.879 0.161 0.913 y 0.828;

0.737 1 0.752 0.709 0.828 0.135 0.713 0.737 0.874;

0.968 0.752 1 0.944 0.908 0.285 0.965 0.968 0.879;

0.887 0.709 0.944 1 0.946 0.539 0.987 0.887 0.827;

0.879 0.828 0.908 0.946 1 0.416 0.935 0.879 0.872;

0.161 0.135 0.285 0.539 0.416 1 0.503 0.161 0.209;

0.913 0.713 0.965 0.987 0.935 0.503 1 0.913 0.844;

y 0.737 0.968 0.887 0.879 0.161 0.913 1 0.828;

0.828 0.874 0.879 0.827 0.872 0.209 0.844 0.828 1;];

eig(rho)

cumT = cumsum(Trans,2);

cumT(:,end) = ones(8,1);

transitionValues = norminv(cumT);

transitionValues(isnan( transitionValues )) = inf;

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = equities(end,1);

x = PricingInput(1,equities(end,:)',zeroUSD(end,1), underlying(end,:)', spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), bondRatings, [zeros(9,1),reshape(lambda,8,9)']);

x.cds\_underlying\_rating = [3 4 4 4 4 2 3 3 6];

x.historical\_implied\_spreads = bondSpreads(end,:);

p0 = price(x);

nSims = 1000;

sims = mvnrnd(zeros(nFirms,1),rho, nSims);

futureRatings = zeros(nSims,nFirms);

prices = zeros(nSims,1);

x.num\_of\_days\_elapsed = 1;

for i=1:nSims

futureRatings(i,:) = simRating(sims(i,:), currentRatings, transitionValues);

x.cds\_underlying\_rating = futureRatings(i,:);

x.historical\_implied\_spreads = bondSpreads(end,:);

prices(i) = price(x);

end

hist(prices - p0,100);

CreditVaR99 = prctile(prices - p0, 1);

CreditVaR95 = prctile(prices - p0, 5);

CreditVar999 = prctile(prices - p0, 0.1);

% Compute the new price of the portfolio

% Compute the change in price of the portfolio

## CREDIT VALUE ADJUSTEMENT AND DEBT VALUE ADJUSTMENT

% %clc;

% %clear all;

%

%

% %INPUTS: 1. Historical time series for stocks, interest rates, fx rates,

% % 2. Implied vol surface (for option pricing)

% % 3. Spread curves for each credit rating

% % 4. Transition matrices for credit quality

%

% % Gathering Risk Factors

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

equities = xlsread('data.xlsx','Stocks','B2:D649');

underlying = xlsread('data.xlsx','Underlying','B3:D650');

impVol = xlsread('portfolio\_data.xlsm','Options','K2:K4');

zeroUSD = xlsread('data.xlsx','USD','B6:P649')./100;

zeroCurveTimes = [3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30];

zeroCAD = xlsread('data.xlsx','CAD','B7:AD650')./100;

zeroCAD( :, all( isnan( zeroCAD ), 1 ) ) = [];

zeroEUR = xlsread('data.xlsx','EUR','B7:AD650')./100;

%zeroEUR( :, all( isnan( zeroEUR ), 1 ) ) = [];

fxUSDCAD = xlsread('data.xlsx','FX','B3:B650');

fxEURCAD = xlsread('data.xlsx','FX','E3:E650');

spreads = xlsread('SpreadsbySector','Communications','L2:R16')./10^4;

% Inputing spread time series for each CDS

cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

cdsCNQCN = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'K7:R650')))./100./100;

cdsSABR = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'S7:Z650')))./100./100;

cdsHOT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AA7:AH650')))./100./100;

cdsFOXA = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AI7:AP650')))./100./100;

cdsFRANCE = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AQ7:AX650')))./100./100;

cdsCAT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AY7:BF650')))./100./100;

cdsWFC = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BG7:BN650')))./100./100;

cdsHUNT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BO7:BV650')))./100./100;

% Convert to hazard rates

lambdaGE = cdsGE ./ (1-0.4);

lambdaCNQCN = cdsCNQCN ./ (1-0.4);

lambdaSABR = cdsSABR ./ (1-0.4);

lambdaHOT = cdsHOT ./ (1-0.4);

lambdaFOXA = cdsFOXA ./ (1-0.4);

lambdaFRANCE = cdsFRANCE ./ (1-0.4);

lambdaCAT = cdsCAT ./ (1-0.4);

lambdaCAT(1,6) = lambdaCAT(2,6)+0.0001;

lambdaWFC = cdsWFC ./ (1-0.4);

lambdaHUNT = cdsHUNT ./ (1-0.4);

impLambda = [zeros(9,1), xlsread('Counterparty\_Spreads.xlsx','Sheet1','B2:I10')]./100./100;

impLambda = impLambda ./ (1-0.4);

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

%Input risk factors in seperate matrices

equitiesToday = equities(end,:);

underlyingToday = underlying(end,:);

fxUSDCADToday = fxUSDCAD(end,:);

fxEURCADToday = fxEURCAD(end,:);

riskFactors = [zeroUSD, zeroCAD, zeroEUR, lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT, lambdaFOXA,...

lambdaFRANCE, lambdaCAT, lambdaWFC, lambdaHUNT];

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.historical\_implied\_spreads = bondSpreads(end,:);

[p0, b, ~, ~, cdsZ] = price(z);

%% Principal component analysis on interest rate curves

zeroRates = [zeroUSD, zeroCAD, zeroEUR];

% USD

S = cov(zeroUSD);

% Eigenvalue decomposition

[QUSD, LambdaUSD] = pcacov(S);

Q1 = QUSD(:,1);

Lambda1 = LambdaUSD(1,1);

weights = zeros(size(zeroUSD,2),1);

for y=1:size(zeroUSD,2)

weights(y) = LambdaUSD(y) ./ sum(LambdaUSD);

end

xUSD = (diag(LambdaUSD)^(-1/2) \* QUSD' \* zeroUSD')';

% CAD

S = cov(zeroCAD);

% Eigenvalue decomposition

[QCAD, LambdaCAD] = pcacov(S);

Q1 = QCAD(:,1);

Lambda1 = LambdaCAD(1,1);

weights = zeros(size(zeroCAD,2),1);

for y=1:size(zeroCAD,2)

weights(y) = LambdaCAD(y) ./ sum(LambdaCAD);

end

xCAD = (diag(LambdaCAD)^(-1/2) \* QCAD' \* zeroCAD')';

% EUR

S = cov(zeroEUR);

% Eigenvalue decomposition

[QEUR, LambdaEUR] = pcacov(S);

Q1 = QEUR(:,1);

Lambda1 = LambdaEUR(1,1);

weights = zeros(size(zeroEUR,2),1);

for y=1:size(zeroEUR,2)

weights(y) = LambdaEUR(y) ./ sum(LambdaEUR);

end

xEUR = (diag(LambdaEUR)^(-1/2) \* QEUR' \* zeroEUR')';

PCAriskFactors = [ xUSD(:,1), xCAD(:,1), xEUR(:,1)];

%% Computing returns

riskFactors = [xUSD(:,1), xCAD(:,1), xEUR(:,1), lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT, lambdaFOXA,...

lambdaFRANCE, lambdaCAT, lambdaWFC, lambdaHUNT];

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:size(riskFactorReturns,2);

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

%% Compute correlation matrix of risk factors

rho = corr(riskFactorReturns);

%counterPartyRatings =[2 4 4 4 4 4 3 3 6];

counterPartyRatings =[4 5 4 5 4 4 3 4 3];

T = ['2019-12-20';'2020-06-20';'2016-12-20';'2016-12-20';'2016-12-20';'2018-09-20';'2019-12-20';'2021-06-20';'2017-06-20'];

today = ['7-11-2016'; '7-11-2016'; '7/11/2016'; '7/11/2016';'7/11/2016';'7/11/2016';'7/11/2016' ;'7/11/2016';'7/11/2016'];

daysToExp = wrkdydif(today, T, 1);

yearsToExp = daysToExp ./ 252;

lambdaTimes = [0, c.CDS\_lambda\_times];

% Generate normalized correlated Gaussian random variables (can also be

% used for credit modelling)

delta = 1/252;

T = max(yearsToExp); % Number of years to simulate in the future

N = T/delta;

nSims = 100;

prices = zeros(nSims,1);

exposuresCVA = zeros(ceil(N/2),length(counterPartyRatings));

exposuresDVA = zeros(ceil(N/2),length(counterPartyRatings));

currExpCVA = zeros(ceil(N/2),length(counterPartyRatings));

currExpDVA = zeros(ceil(N/2),length(counterPartyRatings));

expPrice = zeros(ceil(N/2),length(counterPartyRatings));

currExpPrice = zeros(ceil(N/2),length(counterPartyRatings));

expo = max(cdsZ,0)';

for i=1:nSims

tic

z = mvnrnd(zeros(1,size(riskFactors,2)), rho, N);

% Simulate all risk factors using correct models and correlations

zZeroUSD = z(:,1);

zZeroCAD = z(:,2);

zZeroEUR = z(:,3);

zLambda = z(:,4:end);

lambdaSim = simGBM(riskFactors(end,4:end)',mean(riskFactorReturns(:,4:end))'.\*252, std(riskFactorReturns(:,4:end))'.\*sqrt(252), zLambda, delta,T);

[t, s, k] = CalibrateVasicek(xUSD(:,1),delta);

xSimUSD = simVas(xUSD, T, delta, zZeroUSD, t, s, k, 1); % Simulate under risk neutral measure

USDsim = (QUSD \* diag(LambdaUSD)^(1/2) \* xSimUSD')';

[t, s, k] = CalibrateVasicek(xCAD(:,1),delta);

xSimCAD = simVas(xCAD, T, delta, zZeroCAD, t, s, k, 1); % Simulate under risk neutral measure

CADsim = (QCAD \* diag(LambdaCAD)^(1/2) \* xSimCAD')';

[t, s, k] = CalibrateVasicek(xEUR(:,1),delta);

xSimEUR = simVas(xEUR, T, delta, zZeroEUR, t, s, k, 1); % Simulate under risk neutral measure

EURsim = (QEUR \* diag(LambdaEUR)^(1/2) \* xSimEUR')';

spot\_at\_expiration = equities(end,1);

for j=1:2:max(daysToExp)

x = PricingInput(j, equities(end,:)', USDsim(j,1), underlying(end,:)', spot\_at\_expiration, impVol, USDsim(j,:), CADsim(j,:), EURsim(j,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), reshape(lambdaSim(j,:),8,9)'] );

%x = PricingInput(j, equities(end,:)', USDsim(j,1), underlying(end,:)', spot\_at\_expiration, impVol, USDsim(j,:), CADsim(j,:), EURsim(j,:), zeroCurveTimes,...

% fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), c.CDS\_lambda]);

x.BONDS = 0;

x.CDS = 1;

x.OPTIONS = 0;

x.STOCKS = 0;

x.historical\_implied\_spreads = bondSpreads(end,:);

%[~, ~, ~, ~, cds] = price(x);

cds = price\_cds\_cva(x, c);

ind = (j-1)/2 + 1;

currExpPrice(ind,:) = cds;

currExpCVA(ind,:) = max(cds,0);

currExpDVA(ind,:) = max(-cds,0) ;

end

expPrice = expPrice + currExpPrice;

exposuresCVA = exposuresCVA + currExpCVA;

exposuresDVA = exposuresDVA + currExpDVA;

toc

end

% Compute the average exposure at each time point

t = 0:2:max(daysToExp)-1;

t = t ./ 252;

DF = exp(-zeroCAD(end,1) .\* t);

exposuresCVA = exposuresCVA / nSims;

exposuresDVA = exposuresDVA / nSims;

%exposuresCVA = exposuresCVAS;

%exposuresDVA = exposuresDVAS;

DiscountFactors = repmat(DF,9,1);

exposuresCVATraspose = exposuresCVA' ;

exposuresDVATraspose = exposuresDVA' ;

exposuresCVA = exposuresCVATraspose .\* DiscountFactors .\* (1-0.4);

exposuresDVA = exposuresDVATraspose .\* DiscountFactors .\* (1-0.4);

exposuresCVA = exposuresCVA';

exposuresDVA = exposuresDVA';

cva = zeros(length(counterPartyRatings),1);

dva = zeros(length(counterPartyRatings),1);

times = zeros(length(counterPartyRatings), ceil(max(daysToExp)/2));

for i=1:length(counterPartyRatings)

times(i,1:length(0:2:daysToExp(i))) = 0:2:daysToExp(i);

end

times = times./252;

lambDVA = impLambda(1,:);

for i=1:length(counterPartyRatings) % moving through CVA calculations for each CDS

lamb = impLambda(counterPartyRatings(i),:);

newTimes = [0; nonzeros(times(i,:))];

qCVA = exp(-interp(yearsToExp(i), lambdaTimes, lamb).\* newTimes(1:end-1)')...

- exp(-interp(yearsToExp(i), lambdaTimes, lamb).\* newTimes(2:end)');

% Assume that our firm is AA rated so that we take the smallest spreads

qDVA = exp(-interp(yearsToExp(i), lambdaTimes, lambDVA).\* newTimes(1:end-1)')...

- exp(-interp(yearsToExp(i), lambdaTimes, lambDVA).\* newTimes(2:end)');

cva(i) = qCVA \* exposuresCVA(1:length(newTimes)-1,i);

dva(i) = qDVA \* exposuresDVA(1:length(newTimes)-1,i);

end

% Compute new price

Vnew = cdsZ - cva + dva;

## GBM CORRELATED ASSET PATHS

function S = AssetPathsCorrelated(S0,mu,sig,corr,dt,steps,nsims)

% Function to generate correlated sample paths for assets assuming

% geometric Brownian motion.

%

% S = AssetPathsCorrelated(S0,mu,sig,corr,dt,steps,nsims)

%

% Inputs: S0 - stock price

% : mu - expected return

% : sig - volatility

% : corr - correlation matrix

% : dt - size of time steps

% : steps - number of time steps to calculate

% : nsims - number of simulation paths to generate

%

% Output: S - a (steps+1)-by-nsims-by-nassets 3-dimensional matrix where

% each row represents a time step, each column represents a

% seperate simulation run and each 3rd dimension represents a

% different asset.

%

% Notes: This code focuses on details of the implementation of the

% Monte-Carlo algorithm.

% It does not contain any programatic essentials such as error

% checking.

% It does not allow for optional/default input arguments.

% It is not optimized for memory efficiency or speed.

% get the number of assets

nAssets = length(S0);

% calculate the drift

nu = mu - sig.\*sig/2;

% do a Cholesky factorization on the correlation matrix

R = chol(corr);

% pre-allocate the output

S = nan(steps+1,nsims,nAssets);

% generate correlated random sequences and paths

for idx = 1:nsims

% generate uncorrelated random sequence

x = randn(steps,size(corr,2));

% correlate the sequences

ep = x\*R;

% Generate potential paths

S(:,idx,:) = [ones(1,nAssets); ...

cumprod(exp(repmat(nu\*dt,steps,1)+ep\*diag(sig)\*sqrt(dt)))]\*diag(S0);

end

% If only one simulation then remove the unitary dimension

if nsims==1

S = squeeze(S);

end

BACKTESTING FOR HISTORICAL VALUE AT RISK

clc;

clear all;

%INPUTS: 1. Historical time series for stocks, interest rates, fx rates,

% 2. Implied vol surface (for option pricing)

% 3. Spread curves for each credit rating

% 4. Transition matrices for credit quality

% Gathering Risk Factors

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

equities = xlsread('data.xlsx','Stocks','B6:D649');

underlying = xlsread('data.xlsx','Underlying','B7:D650');

impVol = xlsread('portfolio\_data.xlsm','Options','K2:K4');

zeroUSD = xlsread('data.xlsx','USD','B6:P649')./100;

zeroCurveTimes = [3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30];

zeroCAD = xlsread('data.xlsx','CAD','B7:AD650')./100;

zeroCAD( :, all( isnan( zeroCAD ), 1 ) ) = [];

zeroEUR = xlsread('data.xlsx','EUR','B7:AD650')./100;

zeroEUR( :, all( isnan( zeroEUR ), 1 ) ) = [];

fxUSDCAD = xlsread('data.xlsx','FX','B7:B650');

fxEURCAD = xlsread('data.xlsx','FX','E7:E650');

spreads = xlsread('SpreadsbySector','Communications','L2:R16')./10^4;

% Inputing spread time series for each CDS

cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

cdsCNQCN = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'K7:R650')))./100./100;

cdsSABR = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'S7:Z650')))./100./100;

cdsHOT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AA7:AH650')))./100./100;

cdsFOXA = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AI7:AP650')))./100./100;

cdsFRANCE = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AQ7:AX650')))./100./100;

cdsCAT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AY7:BF650')))./100./100;

cdsWFC = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BG7:BN650')))./100./100;

cdsHUNT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BO7:BV650')))./100./100;

% Convert to hazard rates

lambdaGE = cdsGE ./ (1-0.4);

lambdaCNQCN = cdsCNQCN ./ (1-0.4);

lambdaSABR = cdsSABR ./ (1-0.4);

lambdaHOT = cdsHOT ./ (1-0.4);

lambdaFOXA = cdsFOXA ./ (1-0.4);

lambdaFRANCE = cdsFRANCE ./ (1-0.4);

lambdaCAT = cdsCAT ./ (1-0.4);

lambdaCAT(1,6) = lambdaCAT(2,6)+0.0001;

lambdaWFC = cdsWFC ./ (1-0.4);

lambdaHUNT = cdsHUNT ./ (1-0.4);

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

%Input risk factors in seperate matrices

riskFactors = [equities, underlying, fxUSDCAD, fxEURCAD, zeroUSD, zeroCAD, zeroEUR, lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT...

lambdaFOXA, lambdaFRANCE, lambdaCAT, lambdaWFC,lambdaHUNT];

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, lambda);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

%% Computing returns

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:8;

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

for y=9:53

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

for y=54:125

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

%% Compute Market Value at Risk and Incremental Value at Risk

VaROneDay95 = zeros(125,1);

outSamplePrices = zeros(125+1,1);

for i=1:125+1

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(518+i,:)',zeroUSD(518+i,1), underlying(518+i,:)',spot\_at\_expiration, impVol, zeroUSD(518+i,:), zeroCAD(518+i,:), zeroEUR(518+i,:), zeroCurveTimes,...

fxUSDCAD(518+i), fxEURCAD(518+i), currentRatings, lambda);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

outSamplePrices(i) = price(z);

end

n = 125;

window = 1:5:n; % we move over 125 windows

% MOVING WINDOW BAKCTESTING

for j=window

inSampleRiskFactorReturns = riskFactorReturns(j:(518+j),:);

inSampleRiskFactors = riskFactors(j:(518+j),:);

Sigma = cov(inSampleRiskFactorReturns); % sample covariance matrix of risk factors

simMovements = mvnrnd(zeros(size(inSampleRiskFactorReturns,2), 1), Sigma, size(inSampleRiskFactorReturns,1));

simRiskFactors = repmat(inSampleRiskFactors(end,:),size(inSampleRiskFactorReturns,1),1) + simMovements;

pricesB = zeros(size(inSampleRiskFactorReturns,1),1);

% Price the simulated scenarios

equitiesx = simRiskFactors(:,1:3);

underlyingx = simRiskFactors(:,4:6);

fxUSDCADx = simRiskFactors(:,7);

fxEURCADx = simRiskFactors(:,8);

zeroUSDx = simRiskFactors(:,9:23);

zeroCADx = simRiskFactors(:,24:38);

zeroEURx = simRiskFactors(:,39:53);

Lx = simRiskFactors(:,54:end);

spot\_at\_expiration = [];

for i=1:size(inSampleRiskFactorReturns,1)

priceObject = PricingInput(1,equitiesx(i,:)',zeroUSDx(i,1), underlyingx(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCADx(i,:), zeroEURx(i,:), zeroCurveTimes,...

fxUSDCADx(i), fxEURCADx(i), currentRatings, reshape(Lx(i,:),8,9)');

pricesB(i) = price(priceObject);

end

deltaP = pricesB - outSamplePrices(j+1);

hist(deltaP,100);

VaROneDay95(j) = prctile(deltaP,5);

end

changeOutSample = outSamplePrices(2:end)-outSamplePrices(1:end-1);

breach = changeOutSample < VaROneDay95;

percentBreaches = sum(breach(1:10:125)) ./ length(breach(1:10:125));

VaRTenDay99 = sqrt(10) \* VaROneDay95;

% hist(prices - p0,50)

% x = prctile(prices - p0, 1)

% set(get(gca,'child'),'FaceColor','none','EdgeColor','r');

% hold on;

% hist(pricesB - p0,50)

% x = prctile(pricesB - p0, 1)

%

BACKTESTING FOR MONTE CARLO 1 DAY

clc;

clear all;

%INPUTS: 1. Historical time series for stocks, interest rates, fx rates,

% 2. Implied vol surface (for option pricing)

% 3. Spread curves for each credit rating

% 4. Transition matrices for credit quality

% Gathering Risk Factors

currentRatings = [3 4 4 5 4 2 4 3 6 4 2 4 3 3 2 3 3 3 4 1 3 4 4 3 1]; % These are the current ratings of each firm

equities = xlsread('data.xlsx','Stocks','B6:D649');

underlying = xlsread('data.xlsx','Underlying','B7:D650');

impVol = xlsread('portfolio\_data.xlsm','Options','K2:K4');

zeroUSD = xlsread('data.xlsx','USD','B6:P649')./100;

zeroCurveTimes = [3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30];

zeroCAD = xlsread('data.xlsx','CAD','B7:AD650')./100;

zeroCAD( :, all( isnan( zeroCAD ), 1 ) ) = [];

zeroEUR = xlsread('data.xlsx','EUR','B7:AD650')./100;

zeroEUR( :, all( isnan( zeroEUR ), 1 ) ) = [];

fxUSDCAD = xlsread('data.xlsx','FX','B7:B650');

fxEURCAD = xlsread('data.xlsx','FX','E7:E650');

spreads = xlsread('SpreadsbySector','Communications','L2:R16')./10^4;

bondSpreads = xlsread('Implied\_Yield','Yield','B2:Z645');

% Inputing spread time series for each CDS

cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

cdsCNQCN = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'K7:R650')))./100./100;

cdsSABR = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'S7:Z650')))./100./100;

cdsHOT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AA7:AH650')))./100./100;

cdsFOXA = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AI7:AP650')))./100./100;

cdsFRANCE = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AQ7:AX650')))./100./100;

cdsCAT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'AY7:BF650')))./100./100;

cdsWFC = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BG7:BN650')))./100./100;

cdsHUNT = cleanData((xlsread('CDS HIstoric.xlsx','SPREAD', 'BO7:BV650')))./100./100;

% Convert to hazard rates

lambdaGE = cdsGE ./ (1-0.4);

lambdaCNQCN = cdsCNQCN ./ (1-0.4);

lambdaSABR = cdsSABR ./ (1-0.4);

lambdaHOT = cdsHOT ./ (1-0.4);

lambdaFOXA = cdsFOXA ./ (1-0.4);

lambdaFRANCE = cdsFRANCE ./ (1-0.4);

lambdaCAT = cdsCAT ./ (1-0.4);

lambdaCAT(1,6) = lambdaCAT(2,6)+0.0001;

lambdaWFC = cdsWFC ./ (1-0.4);

lambdaHUNT = cdsHUNT ./ (1-0.4);

lambda = [lambdaGE(end,:); lambdaCNQCN(end,:); lambdaSABR(end,:); lambdaHOT(end,:); lambdaFOXA(end,:); lambdaFRANCE(end,:);...

lambdaCAT(end,:); lambdaWFC(end,:); lambdaHUNT(end,:)];

%Input risk factors in seperate matrices

riskFactors = [equities, underlying, fxUSDCAD, fxEURCAD, zeroUSD, zeroCAD, zeroEUR, lambdaGE, lambdaCNQCN, lambdaSABR, lambdaHOT...

lambdaFOXA, lambdaFRANCE, lambdaCAT, lambdaWFC,lambdaHUNT, bondSpreads];

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(end,:)',zeroUSD(end,1), underlying(end,:)',spot\_at\_expiration, impVol, zeroUSD(end,:), zeroCAD(end,:), zeroEUR(end,:), zeroCurveTimes,...

fxUSDCAD(end), fxEURCAD(end), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(end,:);

[p0, bonds\_valueZERO, options\_valueZERO, stocks\_valueZERO, cds\_valuesZERO] = price(z);

%% Computing returns

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:8;

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

for y=9:53

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

for y=54:125

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

for y=126:150

riskFactorReturns(:,y) = riskFactors(2:end,y) - riskFactors(1:end-1,y);

end

%% Compute Market Value at Risk and Incremental Value at Risk

VaROneDay95 = zeros(125,1);

outSamplePrices = zeros(125+1,1);

for i=1:125+1

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equities(518+i,:)',zeroUSD(518+i,1), underlying(518+i,:)',spot\_at\_expiration, impVol, zeroUSD(518+i,:), zeroCAD(518+i,:), zeroEUR(518+i,:), zeroCurveTimes,...

fxUSDCAD(518+i), fxEURCAD(518+i), currentRatings, [zeros(9,1), lambda]);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

z.historical\_implied\_spreads = bondSpreads(518+i,:);

outSamplePrices(i) = price(z);

end

n = 125;

nSims = 700;

window = 1:1:125; % we move over 125 windows

% MOVING WINDOW BAKCTESTING

for j=window

tic

j

inSampleRiskFactorReturns = riskFactorReturns(j:(518+j),:);

inSampleRiskFactors = riskFactors(j:(518+j),:);

Sigma = cov(inSampleRiskFactorReturns); % sample covariance matrix of risk factors

nSims = 700;

simMovements = mvnrnd(zeros(size(inSampleRiskFactorReturns,2), 1), Sigma, nSims);

simRiskFactors = repmat(inSampleRiskFactors(end,:),nSims,1) + simMovements;

% Price the simulated scenarios

prices = zeros(nSims,1);

bonds\_value = zeros(nSims,1);

options\_value = zeros(nSims,1);

stocks\_value = zeros(nSims,1);

cds\_values = zeros(nSims,1);

equities = simRiskFactors(:,1:3);

underlying = simRiskFactors(:,4:6);

fxUSDCAD = simRiskFactors(:,7);

fxEURCAD = simRiskFactors(:,8);

zeroUSD = simRiskFactors(:,9:23);

zeroCAD = simRiskFactors(:,24:38);

zeroEUR = simRiskFactors(:,39:53);

L = simRiskFactors(:,54:125);

bondSpreadScen = simRiskFactors(:,126:end);

spot\_at\_expiration = [];

for i=1:nSims

priceObject = PricingInput(1,equities(i,:)',zeroUSD(i,1), underlying(i,:)',spot\_at\_expiration, impVol, zeroUSD(i,:), zeroCAD(i,:), zeroEUR(i,:), zeroCurveTimes,...

fxUSDCAD(i), fxEURCAD(i), currentRatings, [zeros(9,1),reshape(L(i,:),8,9)']);

priceObject.historical\_implied\_spreads = bondSpreadScen(i,:);

[prices(i), bonds\_value(i), options\_value(i), stocks\_value(i), x] = price(priceObject);

cds\_values(i) = sum(x);

end

deltaP = prices - outSamplePrices(j+1);

hist(deltaP,100);

VaROneDay95(j) = prctile(deltaP,5);

toc

end

changeOutSample = outSamplePrices(2:end)-outSamplePrices(1:end-1);

breach = changeOutSample < VaROneDay95;

percentBreaches = sum(breach(window)) ./ length(breach(window));

% hist(prices - p0,50)

% x = prctile(prices - p0, 1)

% set(get(gca,'child'),'FaceColor','none','EdgeColor','r');

% hold on;

% hist(pricesB - p0,50)

% x = prctile(pricesB - p0, 1)

%

## CIR MODEL CALIBRATION

function [alpha, mu, sigma] = CalibrateCIR(x, delta)

%This function calibrates the CIR model

% Inputs:

% x = historic term structure

x1Tilde = x(1:end-1);

dx = diff(x);

dx = dx./x1Tilde.^0.5;

regressors = [delta./x1Tilde.^0.5, delta\*x1Tilde.^0.5];

drift = regressors\dx; % OLS regressors coefficients estimates

res = regressors\*drift - dx;

alpha = -drift(2);

mu = -drift(1)/drift(2);

sigma = sqrt(var(res, 1)/delta);

end

## VASICEK CALIBRATION

function [theta,sigma,k] = CalibrateVasicek(S,delta)

n = length(S)-1;

Sx = sum( S(1:end-1) );

Sy = sum( S(2:end) );

Sxx = sum( S(1:end-1).^2 );

Sxy = sum( S(1:end-1).\*S(2:end) );

Syy = sum( S(2:end).^2 );

theta = (Sy\*Sxx - Sx\*Sxy) / ( n\*(Sxx - Sxy) - (Sx^2 - Sx\*Sy) );

k = -log( (Sxy - theta\*Sx - theta\*Sy + n\*theta^2) / (Sxx -2\*theta\*Sx + n\*theta^2) ) / delta;

a = exp(-k\*delta);

sigmah2 = (Syy - 2\*a\*Sxy + a^2\*Sxx - 2\*theta\*(1-a)\*(Sy - a\*Sx) + n\*theta^2\*(1-a)^2)/n;

sigma = sqrt(sigmah2\*2\*k/(1-a^2));

end

## DATA CLEANING

function input = cleanData(input)

t = linspace(1, size(input,1), size(input,1));

% indices to NaN values in x

% (assumes there are no NaNs in t)

nans = isnan(input);

nans = input<=0;

% replace all NaNs in x with linearly interpolated values

for i=1:size(input,2)

temp = input(:,i);

input(nans(:,i),i) = interp1(t(~nans(:,i)), temp(~nans(:,i)), t(nans(:,i)));

end

end

## CORRELATION BETWEEN MARKET AND CREDIT RISK

% Considering the correlation between Market and Credit Risk

CreditVaR = CVaRUpper;

MarketVaR = 8.684251925440446e+07;

rho = 0:0.001:1;

TotalVaR = sqrt(MarketVaR^2 + CreditVaR^2 + 2 .\* rho .\* MarketVaR .\* CreditVaR);

plot(rho, TotalVaR);

title('TotalVaR: Showing the correlation relationship')

xlabel('\rho')

ylabel('TotalVaR')

TVaRUpper = TotalVaR(end);

TVaRLower = TotalVaR(1);

CVaR\_CDS = 2.004451627069122e+07;

CVaR\_BONDS = 2.817924122034073e+06;

rho = 0:0.001:1;

TotalVaR = sqrt(CVaR\_CDS^2 + CVaR\_BONDS^2 + 2 .\* rho .\* CVaR\_CDS .\* CVaR\_BONDS);

plot(rho, TotalVaR);

title('CreditVaR\_{tot}: Showing the correlation relationship')

xlabel('\rho')

ylabel('CVaR')

CVaRUpper = TotalVaR(end);

CVaRLower = TotalVaR(1);

## HO-LEE MODEL

% Ho-Lee Model for Lambda's

%cdsGE = cleanData((xlsread('CDS HIstoric.xlsx', 'SPREAD', 'C7:J650')))./100./100;

%lambdaGE = cdsGE ./ (1-0.4);

plot(lambdaGE)

plot(lambdaGE(end,:))

lambdaThreeYear = lambdaGE(:, 4); % 3 year curve

forward = lambdaGE(end, 4); % 3 year spot rate at maturity

lambdaReturns = lambdaThreeYear(2:end)./lambdaThreeYear(1:end-1) - 1;

sigma = std(lambdaReturns) %\* sqrt(252);

T = 252;

delta = 1/252;

lambdaSim = zeros(T,1);

lambdaSim(1) = lambdaGE(end, 4); % starting point of the simulation

for i=2:T

delta = 1/252;

lambdaSim(i) = lambdaSim(i-1) + (forward + sigma^2 \* i/252) \* delta + sigma\* normrnd(0,1);

end

plot([lambdaGE(:,4); lambdaSim])

## CIR SIMULATION

function simulatedFX = simCIR(fx0, alpha, mu, sigma, z, T, delta)

% This function simulates rates in the future using the CIR model

% Inputs:

% fx0 = Current FX rates

% T = The number of years in future to simulate

% delta = The number of years for discretization

N = T / delta;

simulatedFX = zeros(N,length(fx0));

simulatedFX(1,:) = fx0 + alpha .\* (mu - fx0) .\* delta + sigma .\* sqrt(fx0.\*delta).\*z(1,:)';

for i=2:length(simulatedFX)

simulatedFX(i,:) = simulatedFX(i-1,:)' + alpha .\* (mu - simulatedFX(i-1)) .\* delta + sigma .\* sqrt(simulatedFX(i-1)\*delta).\*z(i,:)';

end

end

## GBM SIMULATION

function [S] = simGBM(S0, mu, sigma, z, delta, T)

%simGBM: Generates a path of correlated GBM's

%Ouput = S = Simulated Asset Paths

%S0 = vector of intial prices

%mu = vector of expected returns

%covMat = covariance matrix of returns

%delta = delta t

%T = Number of years to simulate

N = T / delta;

S = zeros(N,length(S0));

S(1,:) = S0 .\* exp((mu-sigma.^2/2)\*delta + sigma.\*sqrt(delta) .\*z(1,:)');

for i=2:N

S(i,:) = S(i-1,:)' .\* exp((mu-sigma.^2/2)\*delta + sigma.\*sqrt(delta) .\*z(i,:)');

end

end

## VASICEK SIMULATION

function newX = simVas(x, T, delta, z, theta, sigma, k, riskNeutral)

% This function simulates rates in the future using the Vasicek model

% Inputs:

% r = historic term structure of rates

% T = The number of years in future to simulate

% delta = The number of years for discretization

if nargin < 8

riskNeutral = 0;

end

% Simulating in risk neutral measure

lambda = -1.2; % The market price of risk for interest rates

if (riskNeutral ==1)

theta = theta - lambda \* sigma / k; % Adjust the drift of the process

end

% The current known value of x

x1 = x(:,1);

x\_current = x(end,:);

x1\_current = x1(end);

N = T / delta;

simulatedx1 = zeros(N,1);

simulatedx1(1) = x1\_current \* exp(-k\*delta) + theta \* (1 - exp(-k \* delta)) + sigma \* sqrt((1-exp(-2\*k\*delta))/2/k) \* z(1);

for i=2:length(simulatedx1)

simulatedx1(i) = simulatedx1(i-1) \* exp(-k\*delta) + theta \* (1 - exp(-k \* delta)) + sigma \* sqrt((1-exp(-2\*k\*delta))/2/k) \*z(i);

end

temp = zeros(N,size(x,2)-1);

for i=1:size(x,2)-1

temp(:,i) = repmat(x\_current(i+1), N,1);

end

newX = [simulatedx1, temp];

end

## STRESSED VALUE AT RISK

%% Stressed VaR Computation

riskFactors = xlsread('RiskFactors\_UnderStress2.xlsx','Sheet2','B2:DV751');

riskFactorsToday = riskFactors(end,:);

riskFactorReturns = zeros(size(riskFactors)- [1 0]);

for y=1:size(riskFactorReturns,2);

riskFactorReturns(:,y) = riskFactors(2:end,y)./riskFactors(1:end-1,y) - 1;

end

priceToday = zeros(250,1);

scenarios = zeros(size(riskFactorReturns));

for i=1:size(scenarios,1)

scenarios(i,:) = (1+riskFactorReturns(i,:)) .\* riskFactorsToday;

end

equitiesScen = scenarios(:,1:3);

underlyingScen = scenarios(:,4:6);

fxUSDCADScen = scenarios(:,7);

fxEURCADScen = scenarios(:,8);

zeroUSDScen = scenarios(:,9:23);

zeroCADScen = scenarios(:,24:38);

zeroEURScen = scenarios(:,39:53);

lambda = scenarios(:,54:125);

bondSpreadScen = scenarios(

for i=1:250

% Compute the portfolio value today using pricers

c = PortfolioConstants;

spot\_at\_expiration = [];

z = PricingInput(0,equitiesScen(499+i,:)',zeroUSDScen(499+i,1), underlyingScen(499+i,:)',spot\_at\_expiration, impVol, zeroUSDScen(499+i,:), zeroCADScen(499+i,:), zeroEURScen(499+i,:), zeroCurveTimes,...

fxUSDCADScen(499+i), fxEURCADScen(499+i), currentRatings, lambda);

z.BONDS = 1;

z.CDS = 1;

z.OPTIONS = 1;

z.STOCKS = 1;

priceToday(i) = price(z);

end

VaR99 = zeros(250,1);

n = 250;

for j=1:5:250 % moving window

equitiesScen = scenarios(0+j:499+j,1:3);

underlyingScen = scenarios(0+j:499+j,4:6);

fxUSDCADScen = scenarios(0+j:499+j,7);

fxEURCADScen = scenarios(0+j:499+j,8);

zeroUSDScen = scenarios(0+j:499+j,9:23);

zeroCADScen = scenarios(0+j:499+j,24:38);

zeroEURScen = scenarios(0+j:499+j,39:53);

lambda = scenarios(0+j:499+j,54:125);

c = PortfolioConstants;

prices = zeros(size(equitiesScen,1),1);

for i=1:size(equitiesScen,1)

x = PricingInput(1,equitiesScen(i,:)',zeroUSDScen(i,1), underlyingScen(i,:)', [], impVol, zeroUSDScen(i,:), zeroCADScen(i,:),...

zeroEURScen(i,:), zeroCurveTimes, fxUSDCADScen(i), fxEURCADScen(i), currentRatings, reshape(lambda(i,:),8,9)');

prices(i) = price(x);

end

disp(j);

deltaP = prices - priceToday(j+1);

VaR99(j) = prctile(deltaP, 1);

end

hist(deltaP)

% Find the index corresponding to the minimum value in the array

[minVal, index] = min(VaR99);

plot(VaR99(1:5:250))

title('Value At Risk from 03/01/07 - 12/31/09 Weekly Movement');

xlabel('Starting 500 Day Window Measured in Weeks Past 03/01/07')