



Credit Risk Analytics II (Hazard Rates)

Dr Richard Diamond & Kannan Singaravelu

Credit Curve

The table below shows the term structure of credit spreads (in bps) for two reference entities: Wells Fargo (WFC) - a well-rated institution and Clear Channel Communication (CCMO) - a highly leveraged entity with oncoming problems with refinancing in 2011, which were known to the markets.

Make your own observation about CCMO spreads. While you would prefer to buy cheaper 1Y CDS, that might not be available, or allowed.

Maturity	WFC	CCMO	$Z(0;T)$
1Y	50	751	0.97
2Y	77	1164	0.94
3Y	94	1874	0.92
5Y	125	4156	0.86
7Y	133	6083	0.81

The objective here is to bootstrap implied survival probabilities along with the term structure of hazard rate for

- WFC bank with recovery rate of 50%
- CCMO corporation with recovery rate of 10%

Extending CDS Analytics - I, we will study an effect of increase in recovery rate on implied survival probabilities. We also will explore what happens with hazard rates and offer the intuition (or counter-intuition) behind the Credit Triangle.

```
In [3]: # Import libraries
import pandas as pd
import numpy as np
from numpy import *

# Plotting library
import matplotlib
import matplotlib.pyplot as plt
```

```
# Plot settings
matplotlib.rcParams['figure.figsize'] = [14.0, 8.0]
matplotlib.rcParams['font.size'] = 18
matplotlib.rcParams['lines.linewidth'] = 2.0

# Interactive plotting
## import cufflinks as cf
## cf.set_config_file(offline=True)

#cf.set_config_file(theme = 'pearl')
# cf.getThemes() ['ggplot', 'pearl', 'solar', 'space', 'white', 'polar',
```

```
In [4]: # subsume list of inputs into a dataframe
cds = pd.DataFrame({'Maturity': [1, 2, 3, 5, 7],
                    'WFC': [50, 77, 94, 125, 133],
                    'CCMO': [751, 1164, 1874, 4156, 6083],
                    'Df': [0.97, 0.94, 0.92, 0.86, 0.81]})

# output
cds
```

```
Out[4]:
```

	Maturity	WFC	CCMO	Df
0	1	50	751	0.97
1	2	77	1164	0.94
2	3	94	1874	0.92
3	5	125	4156	0.86
4	7	133	6083	0.81

Interpolation

Given the missing data points in the CDS table, we'll have to perform interpolation to derive the missing tenor and spreads before we proceed with our analysis.

Discount Factors

For discounting factors, we'll do a log-linear interpolation for $\tau_i < \tau < \tau_{i+1}$,

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

We'll now create a User Defined Function (UDF) in Python to derive the discount factors.

```
In [3]: def get_discount_factor(maturity, discountfactor, tenor):

    max_time_index = len(maturity) - 1

    if tenor == 0: Df = 1.
    if tenor > 0 and tenor < maturity[0]: Df = discountfactor[0]
    if tenor >= maturity[max_time_index]: Df = discountfactor[max_time_in

    for i in range(0, max_time_index):
```

```

        if tenor >= maturity[i] and tenor < maturity[i+1]:
            term1 = ((tenor-maturity[i])/(maturity[i+1] - maturity[i]))*l
            term2 = ((maturity[i+1]-tenor)/(maturity[i+1] - maturity[i]))
            lnDf = term1 + term2
            Df = exp(lnDf)

    return Df

```

```

In [4]: t = np.arange(8)
        Df = np.zeros(len(t))
        for i in range(0, len(t)):
            Df[i] = get_discount_factor(cds.Maturity, cds.Df, t[i])

```

```

In [5]: print(f'Discount Factors: {Df}')

```

```

Discount Factors: [1.          0.97         0.94         0.92         0.88949424
0.86
0.83462566 0.81         ]

```

Credit Spreads

For credit spreads (or equally hazard rates), a linear interpolation is done either using the below formulation or by applying the method of your choice.

$$CDS(\tau) = \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i} CDS_{i+1} + \frac{\tau_{i+1} - \tau}{\tau_{i+1} - \tau_i} CDS_i$$

In this case, we'll leverage the power of NumPy by directly applying the `interp` methods for interpolation of spread values.

```

In [6]: # interpolation WFC spreads
        wfc = np.interp(t, cds.Maturity, cds.WFC)

        # set spreads to zero at t=0
        wfc[0] = 0

        # interpolation CCMO spreads
        ccmo = np.interp(t, cds.Maturity, cds.CCMO)

        # set spreads to zero at t=0
        ccmo[0] = 0

```

```

In [7]: # output the results
        print(f'WFC Spreads: \t {wfc}')
        print(f'CCMO Spreads: \t {ccmo}')

```

```

WFC Spreads:    [ 0.   50.   77.   94.  109.5 125.  129.  133. ]
CCMO Spreads:   [ 0.   751. 1164. 1874. 3015. 4156. 5119.5 6083. ]

```

```

In [8]: # subsume list of inputs into a dataframe
        df = pd.DataFrame({'Maturity': t,
                           'WFC': wfc,
                           'CCMO': ccmo,
                           'Df': Df})

        # output
        df

```

Out [8]:

	Maturity	WFC	CCMO	Df
0	0	0.0	0.0	1.000000
1	1	50.0	751.0	0.970000
2	2	77.0	1164.0	0.940000
3	3	94.0	1874.0	0.920000
4	4	109.5	3015.0	0.889494
5	5	125.0	4156.0	0.860000
6	6	129.0	5119.5	0.834626
7	7	133.0	6083.0	0.810000

Piecewise Constant Hazard Rate

We'll now modify the survival probability function that we built in the last lab to accomodate the calculation of intensity or hazard rates. The piecewise constant hazard rate is given by

$$\lambda_m = -\frac{1}{\Delta t} \log \frac{P(0, t_m)}{P(0, t_{m-1})}$$

```
In [9]: def get_survival_probability(maturity, discountfactor, spread, recovery,
# subsume list of inputs into a dataframe
df = pd.DataFrame({'Maturity': maturity, 'Df': discountfactor, 'Sprea

# convert bps to decimal
df['Spread'] = df['Spread']/10000

# specify delta_t
df['Dt'] = df['Maturity'].diff().fillna(0)

# loss rate
L = 1.0 - recovery

# initialize the variables
term = term1 = term2 = divider = 0

for i in range(0, len(df.index)):
    if i == 0: df.loc[i, 'Survival'] = 1; df.loc[i, 'Hazard'] = 0
    if i == 1: df.loc[i, 'Survival'] = L / (L+df.loc[i, 'Dt']*df.loc[i,
        df.loc[i, 'Hazard'] = -log(df.loc[i, 'Survival']/df.loc[i-1, 'S
    if i > 1:
        terms = 0
        for j in range(1, i):
            term = df.loc[j, 'Df']*(L*df.loc[j-1, 'Survival'] - \
                (L + df.loc[j, 'Dt']*df.loc[
                    df.loc[j, 'Survival'])

            terms = terms + term

        divider = df.loc[i, 'Df']*(L+df.loc[i, 'Dt']*df.loc[i, 'Spread']
        term1 = terms/divider
```

```

        term2 = (L*df.loc[i-1,'Survival']) / (L + (df.loc[i,'Dt'] * d

df.loc[i,'Survival'] = term1 + term2

        if (df.loc[i,'Survival'] >= 0 and df.loc[i-1,'Survival'] >= 0
            df.loc[i, 'Hazard'] = -log(df.loc[i,'Survival']/df.loc[i-

# derive probability of default
df['Default'] = 1. - df['Survival']

# derive marginal probability of default
df['Marginal'] = df['Survival'].diff().fillna(0)

if plot_prob:
    # plot survival probability
    df[['Survival', 'Default']].iplot(title='Survival vs Default Prob
                                     xTitle='CDS Maturity',
                                     yTitle='Survival Probability',
                                     secondary_y = 'Default',
                                     secondary_y_title='Default Prob

if plot_hazard:
    # plot Hazard Rates
    df['Hazard'].iplot(kind='bar', title='Term Structure of Hazard Ra
                       xTitle='CDS Maturity',
                       yTitle='Hazard Rates')

return df

```

```

In [10]: WFC = get_survival_probability(df.Maturity,df.Df,df.WFC,0.50,plot_prob=Tr
WFC

```


Out[10]:

	Maturity	Df	Spread	Dt	Survival	Hazard	Default	Marginal
0	0	1.000000	0.00000	0.0	1.000000	0.000000	0.000000	0.000000
1	1	0.970000	0.00500	1.0	0.990099	0.009950	0.009901	-0.009901
2	2	0.940000	0.00770	1.0	0.969649	0.020871	0.030351	-0.020450
3	3	0.920000	0.00940	1.0	0.944966	0.025785	0.055034	-0.024683
4	4	0.889494	0.01095	1.0	0.915366	0.031825	0.084634	-0.029600
5	5	0.860000	0.01250	1.0	0.880536	0.038793	0.119464	-0.034830
6	6	0.834626	0.01290	1.0	0.854360	0.030178	0.145640	-0.026176
7	7	0.810000	0.01330	1.0	0.827388	0.032079	0.172612	-0.026972

Survival Probability & Recovery Rate

```
In [11]: # plot probabilities for various recovery rate
sp = pd.DataFrame()

for i in arange(0.1, 1, 0.1):
    sp[i] = get_survival_probability(df.Maturity, df.Df, df.WFC, i) ['Survival']

sp.columns = around(sp.columns*100)
sp.iplot(title='Survival Probability for various Recovery Rate',
```

```
xTitle='CDS Maturity',  
yTitle='Survival Probability',  
labels = sp.columns)
```

The recovery rate has to be set within the reasonable range of 10 - 60%, so that the shape of the survival probability curve remain feasible as well. High levels of recovery rate induce a skew.

The relationship between credit spread (short term), hazard rate, and recovery rate is approximated using the Credit Triangle rule, applies over small timescales.

$$\text{CDS}_{1Y} = \lambda \times (1 - R)$$

```
In [12]: # surface plot  
sp.ipplot(kind='surface', title='Relationship of Survival Probability, Mat
```

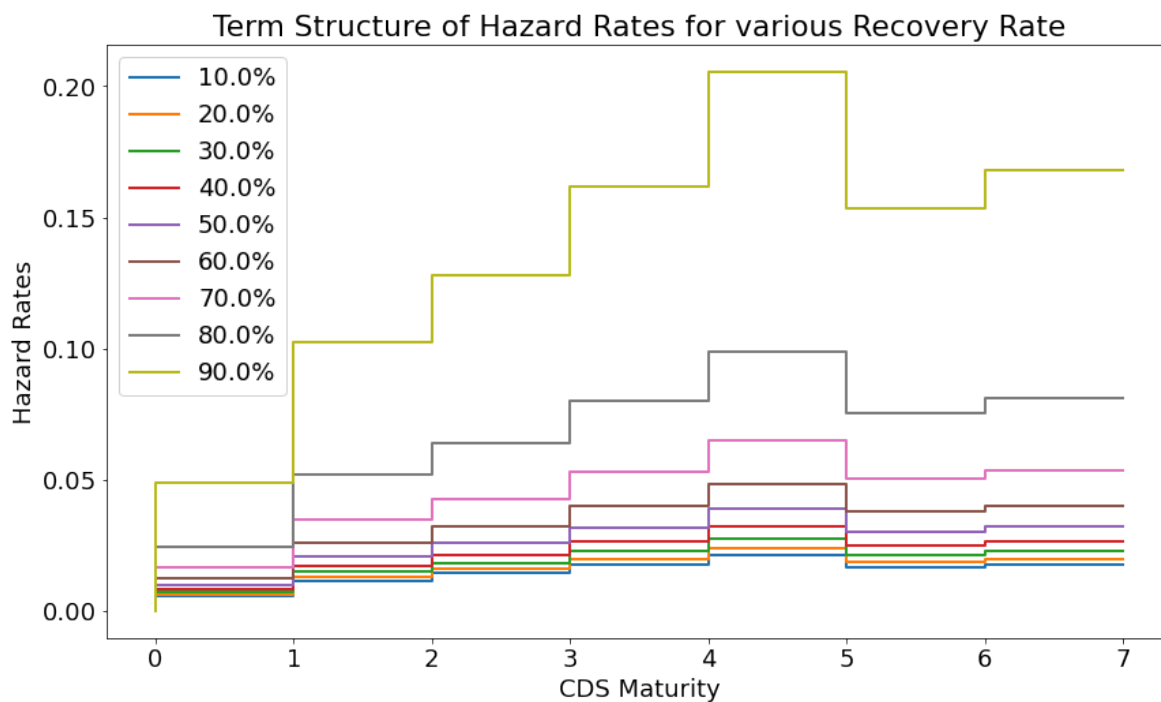

Hazard Rate Visualization

Let's now plot the hazard rates for various recovery rates for WFC.

```
In [13]: hz = pd.DataFrame()

for i in arange(0.1, 1, 0.1):
    hz[i] = get_survival_probability(df.Maturity, df.Df, df.WFC, i) ['Hazard']
    plt.step(hz.index, hz[i], label = f'{i*100:0.4}%')

plt.title('Term Structure of Hazard Rates for various Recovery Rate')
plt.xlabel('CDS Maturity')
plt.ylabel('Hazard Rates')
plt.legend();
```



CCMO Observation

First we remind ourselves the credit spread quotes (bps) and particularly expensive 5Y CDS.

Let's plot the survival probability of CCMO and study its readings.

Maturity	CCMO
1Y	751
2Y	1164
3Y	1874
4Y	3015
5Y	4156
6Y	5119
7Y	6083

```
In [14]: CCMO = get_survival_probability(df.Maturity,df.Df,df.CCMO,0.10,plot_prob=
```

Pricing with QuantLib Library

Given the maturity and market quoted spreads, we'll now derive the hazard rates and survival probability using QuantLib-Python and compare the result with the one derived above.

For this exercise, we use the Trans-European Automated Real-time Gross settlement Express Transfer (TARGET) calendar.

Further, we will compute cash values for Default Leg and Premium Leg in order to evaluate likely P&Ls from a credit swap.

Maturity	Spread	$Z(0; T)$
0Y	0	1
1Y	50	0.97
2Y	77	0.94
3Y	94	0.92
4Y	109.5	0.89
5Y	125	0.86

```
In [15]: from QuantLib import *
```

```
In [16]: maturity = np.arange(6)
discountfactor = [1, 0.97, 0.94, 0.92, 0.89, 0.86] # updated DF to 1 at
cds_spread = [0, 50, 77, 94, 109.5, 125] # assume reasonable spreads fro
recovery = 0.40
```

```
In [17]: sp = get_survival_probability(maturity, discountfactor, cds_spread, recov
sp
```

```
Out[17]:
```

	Maturity	Df	Spread	Dt	Survival	Hazard	Default	Marginal
0	0	0.00	0.00000	0.0	1.000000	0.000000	0.000000	0.000000
1	1	0.97	0.00500	1.0	0.991736	0.008299	0.008264	-0.008264
2	2	0.94	0.00770	1.0	0.974623	0.017406	0.025377	-0.017113
3	3	0.92	0.00940	1.0	0.953894	0.021498	0.046106	-0.020728
4	4	0.89	0.01095	1.0	0.928942	0.026506	0.071058	-0.024952
5	5	0.86	0.01250	1.0	0.899443	0.032271	0.100557	-0.029500

```
In [18]: # set up
calendar = TARGET()

todaysDate = Date(14, December, 2020)
Settings.instance().evaluationDate = todaysDate

risk_free_rate = YieldTermStructureHandle(FlatForward(todaysDate, 0.0, Ac
```

Hazard Rates

```
In [19]: # CDS parameters
recovery_rate = 0.4
quoted_spreads = [0.005, 0.0077, 0.0094, 0.01095, 0.0125]
tenors = [Period(1, Years),
           Period(2, Years),
           Period(3, Years),
           Period(4, Years),
           Period(5, Years)]
maturities = [calendar.adjust(todaysDate + x, Following) for x in tenors]

instruments = [
    SpreadCdsHelper(
        QuoteHandle(SimpleQuote(s)),
        tenor,
        0,
        calendar,
        Annual,
        Following,
        DateGeneration.TwentiethIMM,
        Actual365Fixed(),
        recovery_rate,
        risk_free_rate,
    )
]
```

```

    for s, tenor in zip(quoted_spreads, tenors)
]

# calculate hazard rate
hazard_curve = PiecewiseFlatHazardRate(todaysDate, instruments, Actual365
print("----"*10)
print("Calibrated hazard rate values: ")
print("----"*10)
for x in hazard_curve.nodes():
    print("hazard rate on %s is %.7f" % x)

survival_probability = list(sp['Survival'])[1:]
temp = list(zip(maturities, survival_probability))

# calculate survival probability
print("----"*10)
print("Survival probability values: ")
print("----"*10)
counter = 1
for each in temp:
    print(f"{counter}Y probability QuantLib:{hazard_curve.survivalProbabi
    counter = counter + 1

```

Calibrated hazard rate values:

hazard rate on December 14th, 2020 is 0.0083334
hazard rate on December 20th, 2021 is 0.0083334
hazard rate on December 20th, 2022 is 0.0174670
hazard rate on December 20th, 2023 is 0.0215308
hazard rate on December 20th, 2024 is 0.0263512
hazard rate on December 22nd, 2025 is 0.0318601

Survival probability values:

1Y survival probability:0.991701,
expected 0.991736
2Y survival probability:0.974676,
expected 0.974623
3Y survival probability:0.953978,
expected 0.953894
4Y survival probability:0.929041,
expected 0.928942
5Y survival probability:0.900041,
expected 0.899443

Reprice Instruments

```

In [20]: # reprice instruments
nominal = 1000000.0 # Notional Principal
probability = DefaultProbabilityTermStructureHandle(hazard_curve)

# create a cds for every maturity:
all_cds = []
for maturity, s in zip(maturities, quoted_spreads):
    schedule = Schedule(
        todaysDate,
        maturity,
        Period(Annual),

```

```

        calendar,
        Following,
        Unadjusted,
        DateGeneration.TwentiethIMM,
        False,
    )
    cds = CreditDefaultSwap(Protection.Seller, nominal, s, schedule, Foll
    engine = MidPointCdsEngine(probability, recovery_rate, risk_free_rate
    cds.setPricingEngine(engine) # Computation of cashflows, PL and DL o
    all_cds.append(cds)

print("----"*20)
print("Repricing of quoted CDS spreads employed for calibration: ")
print("----"*20)
for cds, tenor in zip(all_cds, tenors):
    print(f"{tenor} Fair spread : {cds.fairSpread():.6}")
    print(f"    NPV          : {cds.NPV():.2}")
    print(f"    default leg : {cds.defaultLegNPV():.6}") # Default Leg x N
    print(f"    coupon leg  : {cds.couponLegNPV():.6}") # Premium Leg x N
    print("")

```

Repricing of quoted CDS spreads employed for calibration:

```

1Y Fair spread : 0.005
   NPV          : 1.39416e-08
   default leg  : -5060.76
   coupon leg   : 5060.76

2Y Fair spread : 0.0077
   NPV          : 5.45697e-12
   default leg  : -15362.3
   coupon leg   : 15362.3

3Y Fair spread : 0.0094
   NPV          : -4.36557e-11
   default leg  : -27815.5
   coupon leg   : 27815.5

4Y Fair spread : 0.01095
   NPV          : 2.91038e-11
   default leg  : -42736.6
   coupon leg   : 42736.6

5Y Fair spread : 0.0125
   NPV          : -4.29281e-10
   default leg  : -60211.2
   coupon leg   : 60211.2

```

SCENARIO

We have purchased a 5Y CDS at 125 bps, so we are the PAYER of this fixed premium (coupon).

Market information came in about a potential trouble with the company's refinancing, and survival probabilities decreased. The value of default leg appreciated say to 200,000.

In dollar terms, PV-ed, we have committed to pay 60,211.2 only. Therefore, our scenario P&L is as follows:

$$200,000 - 60,211.2 = 139,788.8$$

The fair quote of the spread S_N is such that the expected present value of the payments made by buyer and seller are equal **under a risk-neutral expectation**, i.e.

$$PL_N = DL_N$$

REMINDER CONTENT from CDS Analytics - I

Premium legs are computed by multiplying with the corresponding discount factors and probability of survival at each payment dates (time points).

$$PL_N = S_N \sum_{n=1}^N D(0, T_n) P(T_n) (\Delta t_n)$$

where, Δt_n is the year fraction corresponding to $T_{n-1} - T_n$ and $P(T_n)$ is the survival probability up to time T_n . This accounts for the expected present value of payments made from the buyer **A** to the seller **B**.

Default legs are conditional on the occurrence of default event. The expected present value of this payment depends on the recovery rate R in the event of default. The loss payment on default is then equal to $(1 - R)$ for every \$1 of notional principal. Thus, the expected present value of loss payments is given as

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

The probability is given by the probability of surviving until period $n - 1$ and then defaulting in period n .

References

- CDS Historical Data <http://www.worldgovernmentbonds.com/cds-historical-data/> (careful of adverts)
- ESRB Risk Dashboard / Macro Risk <https://sdw.ecb.europa.eu/reports.do?node=1000003315> (2.8 Sovereign debt CDS premia plotted)
- QuantLib-Python Documentation <https://quantlib-python-docs.readthedocs.io/en/latest/index.html> and QuantLib-SWIG <https://github.com/lballabio>
- Cufflinks Documentation <https://github.com/santosjorge/cufflinks> and <https://plotly.com/python/cufflinks/>

