



## Credit Risk Analytics - I

Dr Richard Diamond (edited) & Kannan Singaravelu

### Credit Default Swap

Credit Default Swap (CDS) is a derivative of a bond or a securitized instruments that involves transfer of a credit risk between two parties. CDS = Credit + Default + Swap, where the 'credit' is linked to bond or loan, 'default' is linked to sensitivity of default event or bankruptcy and 'swap' involves exchange of cashflows. Here, the seller takes the credit risk that the buyer does not wish to shoulder in exchange for a premium over a certain period. The buyer of a CDS contract is seeking insurance against a default of an underlying bond, while the seller is offering a guarantee that the bond will be repaid. The spread of a CDS is the annual amount of the protection buyer must pay the protection seller over the length of the contract, expressed as a percentage of the notional amount.

For example, consider a CDS contract be between two parties say party A and party B. Let Party A be the protection buyer and Party B be the protection seller. The following is the cash flow payment patterns for the above example CDS trade:

#### Party A

1. No Credit Event: Party A will continue to make premium payment at pre-defined points of time until contract maturity.
2. Credit Event Occurs: Party A will not pay premium payments due in the future anymore.

#### Party B

1. No Credit Event: Party B will continue to receive premium payment at pre-defined points of time until contract maturity.
2. Credit Event Occurs: Party B will have to make good the loss to Party A owing to the credit event occurring.

Generally, the credit event could be any of the four occurrences:

1. Complete default on payments
2. Partial default on payments
3. Credit rating downgrade
4. Widening credit spreads

However, as these trades are executed on Over the Counter (OTC), both parties can define credit events as per their requirements in the contract at the time of entering into the trade. CDS are generally quoted as spreads. All thing being equal, at any given time, if the maturity of two CDS is the same, then the CDS with a company with a higher (CDS) spread is considered more likely to default by the market.

Application of CDS includes Hedging, Speculation and Capital Structure Arbitrage (CSA).

```
In [28]: # Import libraries
import pandas as pd
import numpy as np
from IPython.display import HTML

# Plotting library
import cufflinks as cf
cf.set_config_file(offline=True)

# Handle Warnings
import warnings
warnings.filterwarnings('ignore')
```

### Default Modeling

To model the *arrival of a credit event*, we need to model an unknow random point in time  $\tau \in \mathbb{R}_+$ . For default risk modeling, we use the *default indicator* function and *survival indicator* function which is one minus the default indicator function. The survival indicator function is thus represented as,

$$I(t) = 1_{\tau > T} = \begin{cases} 1, & \text{if } \tau > nT \\ 0, & \text{if } \tau \leq T \end{cases}$$

where  $\tau$  is the time of default.

### Poisson Process

For analysis of credit events we use Poisson processes. Poisson processes are usually used to model rare events and discretely countable events such as defaults. It is a discrete time process which gives the probability of default between two time intervals conditional on surviving until the initial time point and these inter-arrival times of Poisson process are exponentially distributed.

We can model the arrival of default as the first jump of a Poisson process with intensity  $\lambda$ .

The associated survival probability  $P(t, T)$  can be obtained as,

$$P(t, T) = e^{-\lambda(T-t)}$$

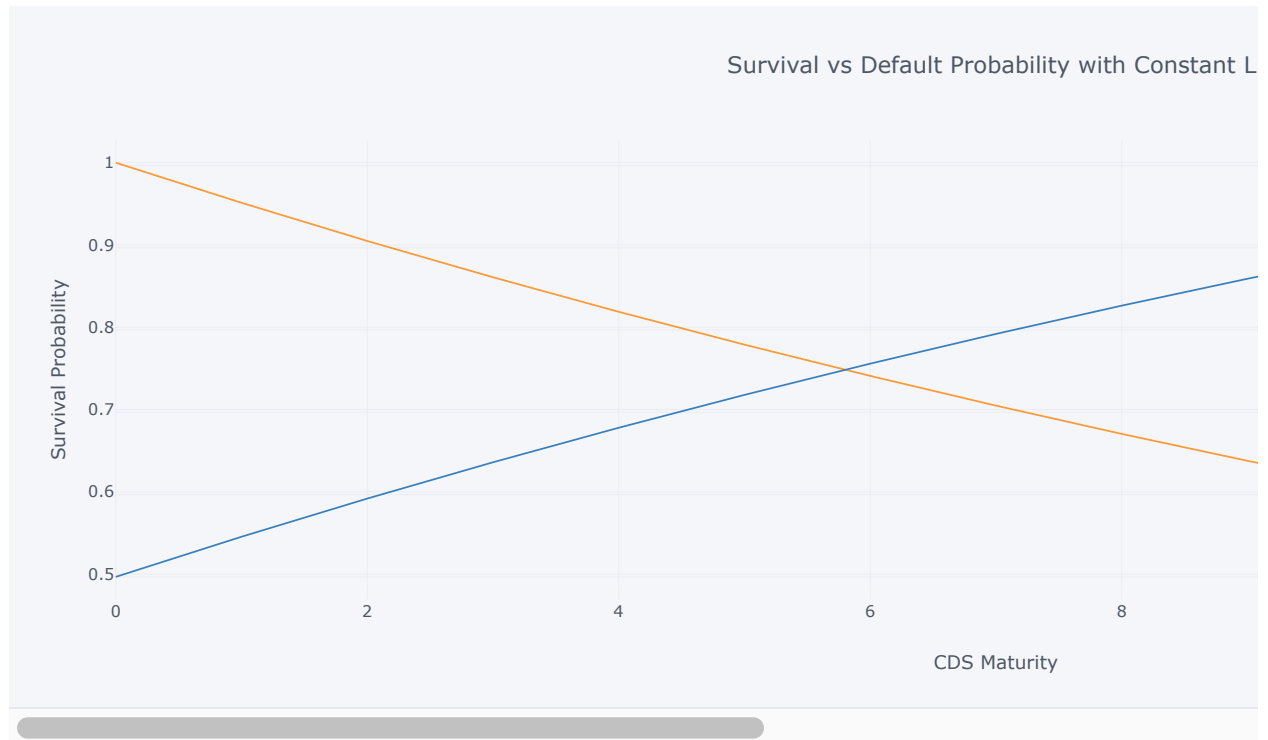
Here, the hazard rate  $\lambda$  is assumed to be constant and does not depend on time. This is an important assumption.

```
In [32]: # specify lambda
lambda_ = 0.05

# specify tenors
tenor = np.arange(15)

# calculate survival probability
data = pd.DataFrame({'Survival': np.exp(-lambda_*tenor)})
data['Default'] = 1.-data['Survival']

In [8]: # visualize the plot
data.iplot(title='Survival vs Default Probability with Constant Lambda',
           xTitle='CDS Maturity',
           yTitle='Survival Probability',
           secondary_y = 'Default',
           secondary_y_title='Default Probability')
```



If the hazard rate  $\lambda$  is known, we can have the entire term structure of probabilities and use these to estimate the likelihood of the various cashflows in the CDS. Premium payments are estimated using survival probability and default payments using probability of default.

```
In [36]: # specify lambdas
lambda1 = 0.01
lambda2 = 0.05
lambda3 = 0.10

# subsume into dataframe
data_ = pd.DataFrame({'$lambda = 1%': np.exp(-lambda1*tenor)})
data_['$lambda = 5%'] = np.exp(-lambda2*tenor)
data_['$lambda = 10%'] = np.exp(-lambda3*tenor)
data_.iplot(title='Theoretical Survival Probs',
           xTitle='CDS Maturity',
           yTitle='Survival Prob')
```



## Inhomogeneous Poisson Process (IHP)

For an Inhomogeneous Poisson Process, intensity  $\lambda(t)$  is made time-dependent. The survival probability becomes

$$P(t, T) = e^{-\int_0^T \lambda(s) ds}$$

The non-constant hazard rate opens way to stochastic modeling (see CQF Extras with Prof Wim Schouten).

However in practice, the time-dependent is assumed to be constant for each time horizon  $T_1, T_2, \dots$  leading to the term structure of hazard rates, which is not flat.

Calibration: the hazard rates term structure matches the market-implied survival probabilities, given by market credit spreads which are the market-quoted quantities.

We will explore more on the term structure of hazard rates in CDS Analytics II lab, however the quick experiment with  $\lambda_1=0.01$ ,  $\lambda_2=0.05$ ,  $\lambda_3=0.10$  give a good indication about the impact of the hazard rate.

## Survival Probability

The survival probability is a key implied quantity.

It balances the computation of cash flows for the Premium Leg and Default Leg of a Credit Default Swap. Therefore, it is the key quantity in the valuation. Occurrence of a credit event is related to the concept of probability of default, which is complement of probability of survival.

### Probability of Default (PD) = 1 - Probability of Survival

The survival probability is not directly observable in the market and has to be implied from traded credit spreads in the market. A bootstrapping algorithm to calculate the survival probability from the market credit spreads has been known as 'JPM Formulation' or JPM CDS Model.

## CDS Pricing: JPM Formulation

In CDS pricing, we bootstrap hazard rates given the fair spread -- market CDS quote.

The spread assumed to be 'fair' because it balances the premium leg and default leg.

CDS pricing model is called the Standard Model by ISDA, and popularly known as JP Morgan Approach.

## Mathematical Setup

1. Discretize the maturity into number of  $\Delta t$  so that the corresponding end of period maturities are then expressed as

$$T_n = n\Delta t$$

The objective is to calculate the cashflow associated with CDS in terms of payment that will be made at these time points.

2. Discount factors are written as functions of forward rates  $r_n$

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

3. Probability of survival, given discretized time and hazard rates (piecewise constant) then be represented as

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

Assuming the obligor is solvent at time zero, we have  $P(T_0) = 1$ .

4. 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,  $\Delta t_n$  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**.

Premium legs are known as Coupon legs (eg QuantLib) and present the PAYER side of the credit swap.

5. 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  $n$ .

6. 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, i.e.

$$PL_N = DL_N$$

Thus, we obtain

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

where,  $P(T_{n-1}) - P(T_n)$  is the marginal probability of survival.

## CDS Bootstrapping

We assume that we have a CDS market spreads for increasing maturities  $S_1, S_2, \dots, S_N$ , we can then determine their associated survival probabilities  $P(T_1), P(T_2), \dots, P(T_N)$ .

First period  $N=1$ ,

$$P(T_1) = \frac{L}{L + \Delta t_1 S_1}$$

Second period  $N=2$ ,

$$P(T_2) = \frac{D(0, T_1) [L(1 - (L + \Delta t_1 S_2) P(T_1))]}{D(0, T_2) (L + \Delta t_2 S_2)} + \frac{P(T_1) L}{L + \Delta t_2 S_2}$$

The general solution for  $P(T_N)$  can be inducted as,

$$P(T_N) = \frac{\sum_{n=1}^{N-1} D(0, T_n) [L P(T_{n-1}) - (L + \Delta t_n S_N) P(T_n)]}{D(0, T_N) (L + \Delta t_N S_N)} + \frac{P(T_{N-1}) L}{L + \Delta t_N S_N}$$

where,

$L = 1 - R$  is our Loss Given Default and  $R$  refers to recovery rate.

## Coding and Survival Prob Output

The bootstrapping algorithm to determine the survival probabilities is coded below.

The implementation is easy to follow because it matches our mathematical setup -- the terms of  $P(T_N)$  are easy to identify in variable names.

**Please Note** For credit spreads, you don't need a data file. The data rows are printed out below: credit spreads for 1Y, ..., 5Y. Such 'credit curves' corporate or sovereign usually require a professional subscription but the order of things (eg whether the 5Y spread is 20bps or 150bps) can be reasonably inferred from the credit rating/IG/HY and the regional classification.

```
In [4]: # Credit Spreads Data (snapshot on a given day)

#df = pd.read_csv('cfs_spreads_1.txt', sep='|t')
df
```

Out[4]:

	Maturity	Df	Spread
0	0	0.00	0.0
1	1	0.97	50.0
2	2	0.94	77.0
3	3	0.92	94.0
4	4	0.89	109.5
5	5	0.86	125.0

```
In [7]: # convert bps to decimal
df['Spread'] = df['Spread']/10000

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

# specify recovery rate
RR = 0.40

# loss rate
L = 1.0 - RR

df
```

Out[7]:

	Maturity	Df	Spread	Dt
0	0	0.00	0.00000	0.0
1	1	0.97	0.00500	1.0
2	2	0.94	0.00770	1.0
3	3	0.92	0.00940	1.0
4	4	0.89	0.01095	1.0
5	5	0.86	0.01250	1.0

```
In [8]: # initialize the variables
term = term1 = term2 = divider = 0
```

```
In [9]: for i in range(0, len(df.index)):
        if i == 0: df.loc[i, 'Survival'] = 1
        if i == 1: df.loc[i, 'Survival'] = L / (L + df.loc[i, 'Dt'] * df.loc[i, 'Spread'])
        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[i, 'Spread']) * \
                                          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'] * df.loc[i, 'Spread']))

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

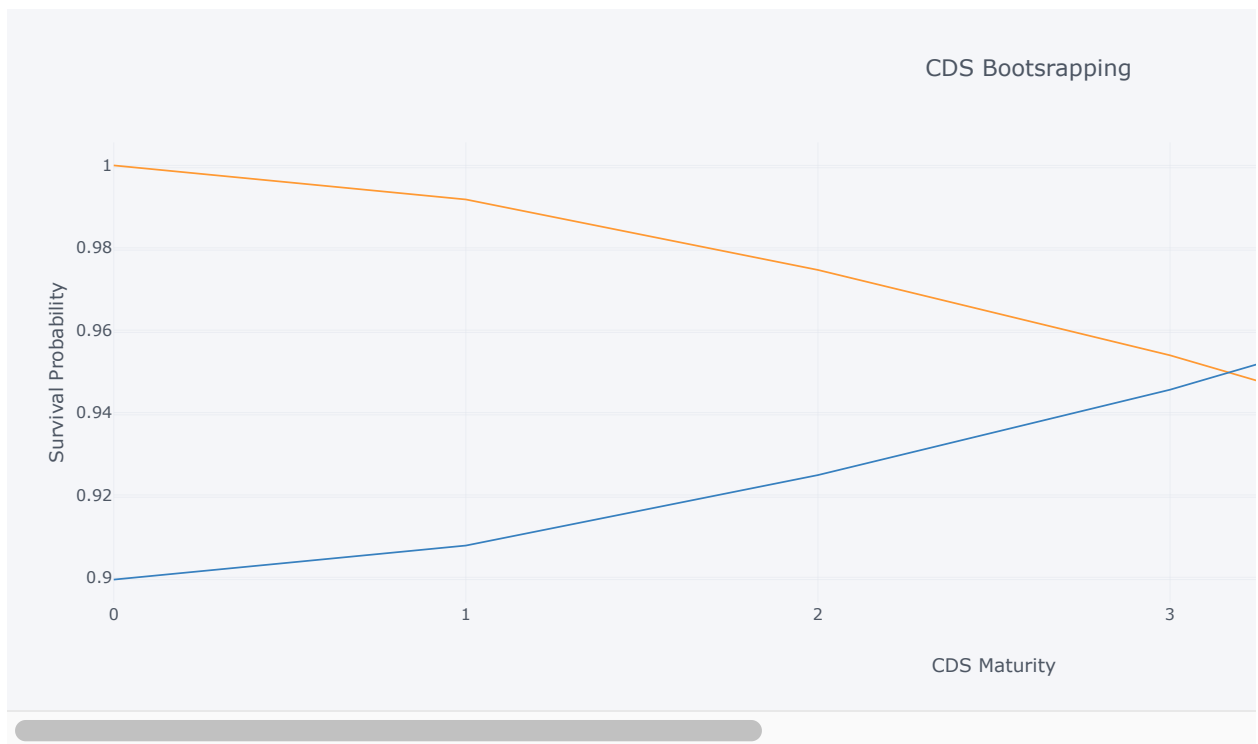
```
In [10]: # derive probability of default
df['Default'] = 1 - df['Survival']

# output the results
df
```

Out[10]:

	Maturity	Df	Spread	Dt	Survival	Default
0	0	0.00	0.00000	0.0	1.000000	0.000000
1	1	0.97	0.00500	1.0	0.991736	0.008264
2	2	0.94	0.00770	1.0	0.974623	0.025377
3	3	0.92	0.00940	1.0	0.953894	0.046106
4	4	0.89	0.01095	1.0	0.928942	0.071058
5	5	0.86	0.01250	1.0	0.899443	0.100557

```
In [11]: # plot survival probability
df[['Survival', 'Default']].plot(title='CDS Bootstrapping',
                                xTitle='CDS Maturity', yTitle='Survival Probability',
                                secondary_y = 'Default', secondary_y_title='Default Probability')
```



## User-Defined Function

```
In [12]: def survival_probability(maturity, discountfactor, spread, recovery, is_plot=False):

    # subsume list of inputs into a dataframe
    df = pd.DataFrame({'Maturity': maturity, 'Df': discountfactor, 'Spread': spread})

    # 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
        if i == 1: df.loc[i, 'Survival'] = L / (L + df.loc[i, 'Dt'] * df.loc[i, 'Spread'])
        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[i, 'Spread']) * \
                                          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'] * df.loc[i, 'Spread']))

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

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

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

    return df
```

```
In [13]: # inputs
maturity = np.arange(6)
discountfactor = [0, 0.97, 0.94, 0.92, 0.89, 0.86]
```

```
cds_spread = [0, 50, 77, 94, 109.5, 125]
recovery = 0.40
```

```
In [14]: # call the user-defined function
sp = survival_probability(maturity, discountfactor, cds_spread, recovery, is_plot=False)
sp

# sp.Survival now stores our Probability of Survival. This will be re-used to check CDS Pricing below.
```

```
Out[14]:
```

	Maturity	Df	Spread	Dt	Survival	Default
0	0	0.00	0.00000	0.0	1.000000	0.000000
1	1	0.97	0.00500	1.0	0.991736	0.008264
2	2	0.94	0.00770	1.0	0.974623	0.025377
3	3	0.92	0.00940	1.0	0.953894	0.046106
4	4	0.89	0.01095	1.0	0.928942	0.071058
5	5	0.86	0.01250	1.0	0.899443	0.100557

## Empirical Snapshot: Bank CDS in 2012-13

```
In [17]: # HSBC
maturity = np.arange(6)
discountfactor = [0, 0.9972, 0.9916, 0.9775, 0.9619, 0.9426]
cds_spread = [0, 11.2, 27.7, 36.9, 57.1, 67.8]
recovery = 0.40
```

```
In [18]: hsbc = survival_probability(maturity, discountfactor, cds_spread, recovery, is_plot=False)
hsbc
```

```
Out[18]:
```

	Maturity	Df	Spread	Dt	Survival	Default
0	0	0.0000	0.00000	0.0	1.000000	0.000000
1	1	0.9972	0.00112	1.0	0.998137	0.001863
2	2	0.9916	0.00277	1.0	0.990802	0.009198
3	3	0.9775	0.00369	1.0	0.981663	0.018337
4	4	0.9619	0.00571	1.0	0.962224	0.037776
5	5	0.9426	0.00678	1.0	0.944246	0.055754

```
In [19]: # Barclays
maturity = np.arange(6)
discountfactor = [0, 0.9972, 0.9916, 0.9775, 0.9619, 0.9426]
cds_spread = [0, 17.7, 44.6, 54.8, 83.5, 96.2]
recovery = 0.40
```

```
In [20]: barclays = survival_probability(maturity, discountfactor, cds_spread, recovery, is_plot=False)
barclays
```

```
Out[20]:
```

	Maturity	Df	Spread	Dt	Survival	Default
0	0	0.0000	0.00000	0.0	1.000000	0.000000
1	1	0.9972	0.00177	1.0	0.997059	0.002941
2	2	0.9916	0.00446	1.0	0.985240	0.014760
3	3	0.9775	0.00548	1.0	0.972925	0.027075
4	4	0.9619	0.00835	1.0	0.945239	0.054761
5	5	0.9426	0.00962	1.0	0.921855	0.078145

```
In [21]: # create dataframe of banks cds
bank = pd.DataFrame({'Barclays CDS': barclays['Spread']*10000,
                    'HSBC CDS': hsbc['Spread']*10000,
                    'Barclays PD': barclays['Default']*100,
                    'HSBC PD': hsbc['Default']*100})

# plot bank cds & probability of default
bank.iplot(title='Spread vs Default Probability',
           xTitle='CDS Maturity',
           yTitle='CDS Spread (bps)',
           secondary_y = ['Barclays PD', 'HSBC PD'],
           secondary_y_title='Default Probability (%)')
```



CDS Pricing

Return to our initial example, some generic credit curve for which we have bootstrapped Survival Probabilities

Maturity	Surv Prob
0Y	1.000000
1Y	0.991736
2Y	0.974623
3Y	0.953894
4Y	0.928942
5Y	0.899443

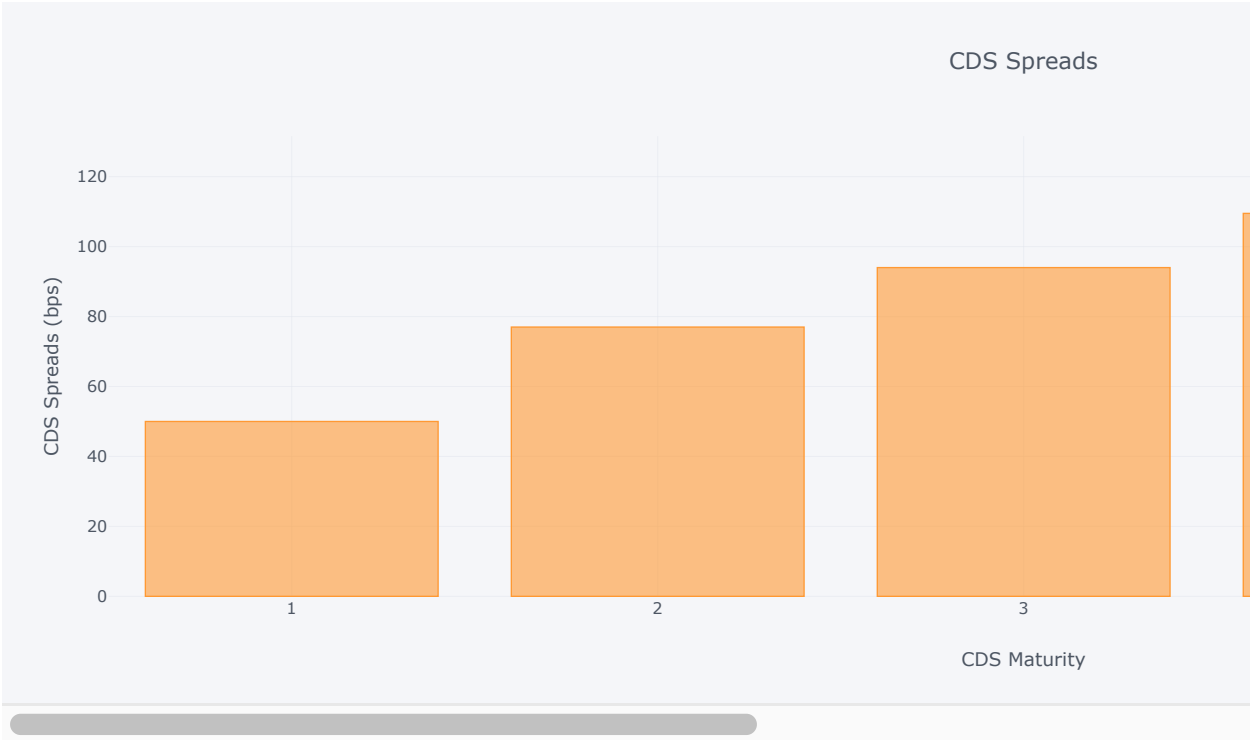
If we know these implied quantities and matching discounting factors, then we can use the formula for  $S_N$  to recompute the credit spreads.

```
In [22]: def get_cds_spreads(maturity, discountfactor, probability, recovery, is_plot=False):  
    # subsume list of inputs into a dataframe  
    df = pd.DataFrame({'Maturity': maturity, 'Df': discountfactor, 'Survival': probability})  
  
    # specify delta_t  
    df['Dt'] = df['Maturity'].diff().fillna(0)  
  
    # loss rate  
    L = 1.0 - recovery  
  
    # initialize the variables  
    nterm = dterm = 0  
  
    for i in range(0, len(df.index)):  
        if i == 0: df.loc[i, 'Spread'] = 0  
        else:  
            nterms = 0; dterms = 0  
            for j in range(1, i+1):  
                nterm = L * df.loc[j, 'Df'] * (df.loc[j-1, 'Survival'] - df.loc[j, 'Survival'])  
                nterms = nterms + nterm  
  
                dterm = df.loc[j, 'Df'] * df.loc[j, 'Survival'] * df.loc[j, 'Dt']  
                dterms = dterms + dterm  
  
            df.loc[i, 'Spread'] = nterms / dterms * 10000  
  
    if is_plot:  
        # plot spreads  
        df['Spread'][1:].plot(kind='bar',  
                               title='CDS Spreads',  
                               xTitle='CDS Maturity',  
                               yTitle='CDS Spreads (bps)')  
  
    return df
```



```
In [23]: # sp.Survival is the INPUT
maturity = np.arange(6)
discountfactor = [0, 0.97, 0.94, 0.92, 0.89, 0.86]
recovery = 0.40

In [24]: get_cds_spreads(maturity, discountfactor, sp.Survival, recovery, is_plot=True)
```



Out [24]:

	Maturity	Df	Survival	Dt	Spread
0	0	0.00	1.000000	0.0	0.0
1	1	0.97	0.991736	1.0	50.0
2	2	0.94	0.974623	1.0	77.0
3	3	0.92	0.953894	1.0	94.0
4	4	0.89	0.928942	1.0	109.5
5	5	0.86	0.899443	1.0	125.0

# References

- Single-name CDS data (brochure) <https://www.spglobal.com/marketintelligence/en/mi/products/single-name-pricing-data.html>
- Cufflinks documentation <https://github.com/santosjorge/cufflinks> and <https://plotly.com/python/cufflinks/>