Graduate School of Management, Politecnico di Milano
Master's Degree Quantitative Finance

Valutazione Prodotti Finanziari - A.Y. 2022/23

Modulo - Credit Risk and Credit Derivatives

Students:
Lorenzo Baggi 10530526
Martina Chiaventi 10929858
Daniele Fazio 10905323
Riccardo Invernizzi 10954454
Marco Lavizzari 10428222
Marco Livraghi 10953813

Professore:
NICOLA MORENI

February-March 2023

Contents

List of Figures

1 Traccia degli esercizi

Considerando come trading date il 20/12/2022, si prendano due CDS standardizzati (rolling su date 'credit IMM'): il primo con maturity 2Y (20/Dec/2024) , il secondo con maturity 4Y (20/Dec/2026). Entrambi i CDS abbiano una cedola contrattuale pari all'1% del nozionale e si ipotizzi una recovery convenzionale pari al 40%. Si ipotizzi inoltre che la curva dei tassi di interesse sia deterministica con tassi zero-coupon pari a 3% p.a..
Sapendo che il premio Upfront quotato dal mercato vale 1.49% del nozionale per il primo CDS e 3.78% del nozionale per il secondo:

- si determinino mediante *bootstrap* due livelli di default intensity λ_1 e λ_2, valide sugli intervalli [0,2y] e [2y,4y] rispettivamente, compatibili con tali quotazioni (per tale scopo si potrà assumere una regola *end-of-period* per i pagamenti della default leg);

- si calcolino le survival probabilities a 1y, 2y, 3y, 4y.

2 Risoluzione Esercizio 1

Al fine di risolvere l'esercizio occorre innanzitutto definire un generico processo di *Poisson time-inhomogeneous*, dove:

- la probabilità di default nell'intervallo [t, t + dt], se non si è ancora falliti al tempo t è data da:

$$Prob_t\big\{\tau \in [t, t+dt]\big\} \approx \lambda(t)dt; \tag{1}$$

- la probabilità di sopravvivenza nel periodo [t,T], se non ancora falliti al tempo t:

$$Surv_t(T) = Prob_t(\tau > T) = e^{-\int_t^T \lambda(s)\,ds}; \tag{2}$$

- la probabilità di default nell'intervallo [u, u + du] se non ancora falliti al tempo t:

$$Prob_t\big\{\tau \in [u, u+du]\big\} = Prob_t(\tau > u) \cdot Prob_u\big\{\tau \in [u, u+du]\big\} = \lambda(u)du \cdot e^{-\int_t^u \lambda(s)\,ds}. \tag{3}$$

Modellare eventi di default mediante un processo di *Poisson* permette di calibrare le λ, ovvero le probabilità di default istantaneo, relativamente alla struttura a termine dei *Credit Default Swap* (CDS). Un CDS è composto da due gambe: una è detta *premium leg* e paga un Upfront e/o una serie di coupon al fine di ricevere una protezione; l'altra è detta *default leg* e, in caso di default, paga la *loss given default* (LGD) (vale: $Rec = 1 - LGD$).
Per un generico CDS si ha che la gamba *premium leg* eguaglia la *default leg* e, di conseguenza, vale la seguente equazione:

$$\begin{aligned} &P(t, t_{settl}) \cdot UF + s\sum_{i=1}^{n}\Big\{\alpha_i \mathbb{E}_t\big[D(t,t_i)\mathbb{1}_{\tau_C > t_i}\big] + \\ &+ \int_{t_{i-1}}^{t_i} \mathbb{E}_t\big[D(t,u)\alpha_i(\frac{u - t_{i-1}}{t_i - t_{i-1}})\mathbb{1}_{\tau \in [u, u+du]}\big]\Big\} \\ &= \sum_{i=1}^{n}\int_{t_{i-1}}^{t_i} \mathbb{E}_t\big[D(t,u) \cdot (1 - Rec)\mathbb{1}_{\tau \in [u,u+du]}\big]. \end{aligned} \tag{4}$$

Da cui, se si assume l'indipendenza tra evento di default e tassi di interesse, è possibile ottenere la seguente equazione:

$$\begin{aligned} &P(t, t_{settl}) \cdot UF + s\sum_{i=1}^{n}\Big\{\alpha_i P(t,t_i) Surv_t(t_i) + \\ &+ \int_{t_{i-1}}^{t_i} P(t,u)\alpha_i(\frac{u - t_{i-1}}{t_i - t_{i-1}})(-1) \cdot dSurv_t(u)\Big\} \\ &= \sum_{i=1}^{n}\int_{t_{i-1}}^{t_i} P(t,u) \cdot (1 - Rec)(-1) \cdot dSurv_t(u). \end{aligned} \tag{5}$$

Inoltre, assumendo come da traccia una regola *end of period* per i pagamenti della default leg, si ottiene la seguente equazione semplificata, di più facile lettura ed implementazione:

$$P(t, t_{settl}) \cdot UF + s\sum_{i=1}^{n}\alpha_i P(t,t_i) Surv_t(t_i) = LGD \cdot \sum_{i=1}^{n} P(t,t_i)\big[Surv_t(t_{i-1}) - Surv_t(t_i)\big]. \tag{6}$$

Da cui, isolando l'Upfront, si ottiene la seguente formula:

$$UF = LGD\sum_{i=1}^{n} P(t_{settl}, t_i)[Surv_t(t_{i-1}) - Surv_t(t_i)] - s\sum_{i=1}^{n}\alpha_i P(t_{settl}, t_i) Surv_t(t_i) \tag{7}$$

Avendo modellato il tempo di default come un processo di *Poisson*, si ha che valgono le seguenti relazioni, ipotizzando che $l'hazard\,rate$ sia definito come una funzione costante a tratti (si indica con t^* il tempo di transizione tra un valore di λ ed il successivo):

$$\mathrm{Surv}_t(T) = e^{-\lambda_1(T-t)}, \text{se } T < t^*,$$

$$\mathrm{Surv}_t(T) = e^{-\lambda_1(t^*-t)}e^{-\lambda_2(T-t^*)}, \text{se } T \geq t^*.$$

Risulta dunque possibile rimaneggiare l'Upfront, calcolandolo in funzione dell'*hazard rate* ed esplicitando l'Upfront UF^{4Y} relativo al secondo CDS:

$$\begin{aligned} UF^{(4Y)} &= LGD \sum_{i=1}^{n2Y} P(t,t_i)(e^{-\lambda_1(t_{i-1}-t)} - e^{-\lambda_1(t_i-t)}) - s\sum_{i=1}^{n2Y} \alpha_i P(t,t_i)e^{-\lambda_1(t_i-t)} + \\ &+ LGD \sum_{i>n2Y}^{n4Y} P(t,t_i)(e^{-\lambda_1(t^*-t)}e^{-\lambda_2(t_{i-1}-t^*)} - e^{-\lambda_2(t_i-t^*)}e^{-\lambda_1(t^*-t)}) + \\ &- s\sum_{i>n2Y}^{n4Y} \alpha_i P(t,t_i)e^{-\lambda_2(t_i-t^*)}e^{-\lambda_1(t^*-t)}. \end{aligned} \tag{8}$$

Come si nota da questa equazione, vi son due variabili incognite: λ_1 e λ_2. Occorre dunque *in primis* ricavare λ_1, risolvendo numericamente la seguente equazione:

$$UF^{(2Y)} = LGD \sum_{i=1}^{n2Y} P(t,t_i)(e^{-\lambda_1(t_{i-1}-t)} - e^{-\lambda_1(t_i-t)}) - s\sum_{i=1}^{n2Y} \alpha_i P(t,t_i)e^{-\lambda_1(t_i-t)}. \tag{9}$$

L'equazione sopra riportata è stata risolta tramite la libreria `Scipy.optimize` presente in *Python*. Per completezza, l'intero codice è riportato in Appendice A. Ottenuto così λ_{2Y} pari a 0.0296, è ora possibile utilizzare tale valore nell'equazione (8) e risolverla numericamente per trovare $\lambda_{4Y} = 0.0395$. Si riportano in figura 1 i risultati ottenuti.

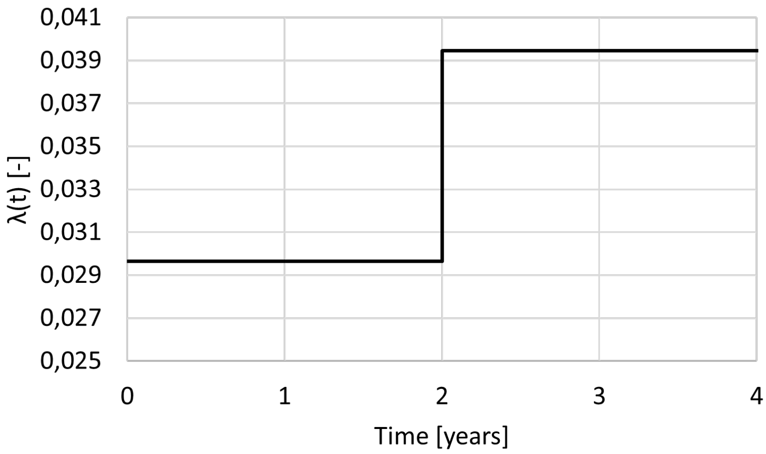

Figure 1: Piecewise constant intensity structure

3 Risoluzione Esercizio 2

Al fine di calcolare le probabilità di sopravvivenza, nota la funzione $\lambda(t)$, occorre semplicemente usare l'equazione (2) riportata in precedenza. Essendo $\lambda(t)$ costante a tratti, la risoluzione dell'integrale risulta essere immediata e di conseguenza si ha che:

- probabilità di sopravvivenza ad un anno [0,1], se non ancora defaultato al tempo 0:

$$Surv_t(1Y) = Prob_t(\tau > 1) = e^{-\int_0^1 \lambda(t)\,dt} = e^{-\lambda_1 \cdot 1Y} = 97\%;$$

- probabilità di sopravvivenza a due anni [0,2], se non ancora defaultato al tempo 0:

$$Surv_t(2Y) = Prob_t(\tau > 2) = e^{-\int_0^2 \lambda(t)\,dt} = e^{-\lambda_1 \cdot 2Y} = 94\%;$$

- probabilità di sopravvivenza a tre anni [0,3], se non ancora defaultato al tempo 0:

$$Surv_t(3Y) = Prob_t(\tau > 3) = e^{-\int_0^3 \lambda(t)\,dt} = e^{-\lambda_1 \cdot 2Y} \cdot e^{-\lambda_2 \cdot (3Y-2Y)} = 90\%;$$

- probabilità di sopravvivenza a quattro anni [0,4], se non ancora defaultato al tempo 0:

$$Surv_t(4Y) = Prob_t(\tau > 4) = e^{-\int_0^4 \lambda(t)\,dt} = e^{-\lambda_1 \cdot 2Y} \cdot e^{-\lambda_2 \cdot (4Y-2Y)} = 87\%.$$

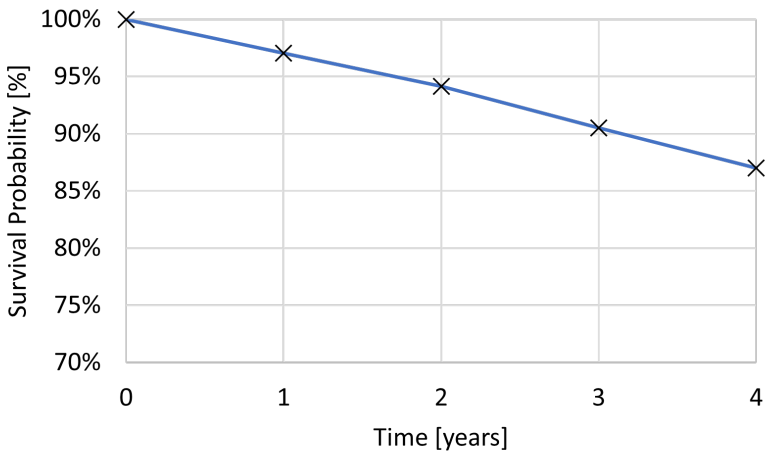

Figure 2: Probabilità di sopravvivenza in funzione del tempo

A Appendix

```
# Importing the needed libraries

import numpy as np
import scipy as sc
from datetime import date
import matplotlib.pyplot as plt
from scipy.optimize import least_squares

# Adding the needed data

rec = 0.4
lgd = 1 - rec
zcrate = 0.03
UF2 = 0.0149
UF4 = 0.0378
s = 0.01

# Creating a calendar

t = []
t.append(0.0)
for i in range(1,5):
    f_date = date(2022, 12, 20)
    l_date = date(2023, 3*i, 20)
    delta = l_date - f_date
    t.append(delta.days/360)
for i in range(1,5):
    f_date = date(2022, 12, 20)
    l_date = date(2024, 3*i, 20)
    delta = l_date - f_date
    t.append(delta.days/360)
for i in range(1,5):
    f_date = date(2022, 12, 20)
    l_date = date(2025, 3*i, 20)
    delta = l_date - f_date
    t.append(delta.days/360)
for i in range(1,5):
    f_date = date(2022, 12, 20)
    l_date = date(2026, 3*i, 20)
    delta = l_date - f_date
    t.append(delta.days/360)


# creating the zero-coupon bond discount factor

p = lambda t: np.exp(-zcrate*t)

# defining the function for the upfront

def uf2(l):
    '''
    Calculate the Upfront given a certain lambda
    '''
```

```
    buyer = [p(t[i])*(np.exp(-l*t[i-1]) - np.exp(-l*t[i]))
                for i in range(1,4*2 + 1)]
    b = sum(buyer)*lgd
    seller = [(t[i]- t[i-1])*p(t[i])*np.exp(-l*t[i])
                for i in range(1,4*2 + 1)]
    ss = s*sum(seller)
    return b - ss

# finding the optimal lambda
lambda2y = sc.optimize.least_squares(lambda x: uf2(x) - UF2,2,
                                      bounds=(0,10)).x[0]
print("Intensity 2y: ", lambda2y)


def uf4(l):
    '''
    Calculate the Upfront given a certain lambda
    '''
    buyer = [p(t[i])*(np.exp(-l*(t[i-1]-t[8]))*np.exp(-lambda2y*t[8])
                  - np.exp(-lambda2y*t[8])*np.exp(-l*(t[i]-t[8])))
            for i in range(9,16+1)]
    b = sum(buyer) * lgd
    seller = [(t[i] - t[i-1])*p(t[i])*np.exp(-lambda2y*t[8])
             *np.exp(-l*(t[i] - t[8]))
             for i in range(9,16+1)]
    ss = s*sum(seller)
    return uf2(lambda2y) + b - ss

# finding the optimal lambda
lambda4y = sc.optimize.least_squares(lambda x: uf4(x) - UF4,0.5,
                                      bounds=(0,10)).x[0]

# Printing some useful values
print("Intensity 4y: ", lambda4y)
print("surv 1y: ", np.exp(-lambda2y*t[4]))
print("surv 2y: ", np.exp(-lambda2y*t[8]))
print("surv 3y: ", np.exp(-lambda4y*(t[12] - t[8]))*np.exp(-lambda2y*t[8]))
print("surv 4y: ", np.exp(-lambda4y*(t[16] - t[8]))*np.exp(-lambda2y*t[8]))
```